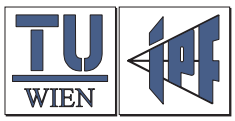
GEOWISSENSCHAFTLICHE MITTEILUNGEN

Heft Nr. 56, 2001

SEMI-AUTOMATIC EXTRACTION OF BUILDINGS BASED ON HYBRID ADJUSTMENT USING 3D SURFACE MODELS AND MANAGEMENT OF BUILDING DATA IN A TIS

von
Franz Rottensteiner



Veröffentlichung des Institutes für
Photogrammetrie und Fernerkundung
122

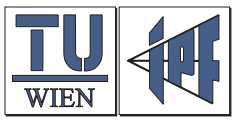
**Schriftenreihe der Studienrichtung VERMESSUNG UND GEOINFORMATION
TECHNISCHE UNIVERSITÄT WIEN**

GEOWISSENSCHAFTLICHE MITTEILUNGEN

Heft Nr. 56, 2001

SEMI-AUTOMATIC EXTRACTION OF BUILDINGS BASED ON HYBRID ADJUSTMENT USING 3D SURFACE MODELS AND MANAGEMENT OF BUILDING DATA IN A TIS

von
Franz Rottensteiner



Veröffentlichung des Institutes für
Photogrammetrie und Fernerkundung
122

**Schriftenreihe der Studienrichtung VERMESSUNG UND GEOINFORMATION
TECHNISCHE UNIVERSITÄT WIEN**

Herausgeber und Verleger: o. Prof. Dr.-Ing. Karl Kraus
Vorstand des Institutes für Photogrammetrie und Fernerkundung
der Technischen Universität Wien
A-1040 Wien, Gußhausstraße 27-29

Die Kosten für den Druck wurden aus eigenen Einnahmen des Institutes für Photogrammetrie und Fernerkundung der Technischen Universität Wien getragen.

Diese Arbeit wurde an der Fakultät für Technische Naturwissenschaften und Informatik der Technischen Universität Wien (Karlsplatz 13, A-1040 Wien, Österreich) zur Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften eingereicht.

Begutachter:

Ao. Univ.-Prof. Dipl.-Ing. Dr. Josef Jansa
Institut für Photogrammetrie und Fernerkundung der Technischen Universität Wien
A-1040 Wien, Gußhausstraße 27-29, Österreich

Ao. Univ.-Prof. Dipl.-Ing. Dr. Werner Schneider
Institut für Vermessung, Fernerkundung und Landinformation
der Universität für Bodenkultur Wien
A-1190 Wien, Peter Jordan-Straße 82, Österreich

Tag der mündlichen Prüfung: 15. 3. 2001

Druck: die kopie, A-1040 Wien
Auflage: 250 Stück
ISBN 3-9500791-3-0

Acknowledgement

This work was funded by the Austrian Science Fund (FWF) and the Austrian National Bank in the framework of the Research Program S7004-MAT on Pattern Recognition and Digital Image Processing, Project IV: Stereovideometry and Spatial Object Recognition.

I also want to thank Fischer Luftbild in Graz for providing the aerial images of the test block in Stoitzendorf to the Institute of Photogrammetry and Remote Sensing free of charge.

Personal thanks

I want to thank my wife Andrea for the patience she had with me especially in the phases of intense work with this thesis. I also want to thank my parents Maria and Franz Rottensteiner for giving me the possibility to study under circumstances which were sometimes not so easy for them.

At the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology I want to thank my colleagues who helped me completing this work. My supervisor, Prof. Josef Jansa, always had time for fruitful discussions. Helmut Kager, Norbert Pfeifer and Gottfried Mandlburger read parts of the manuscript and gave many valuable hints for its improvement. Martin Kerschner read the whole manuscript very carefully and helped me to detect some hidden errors in it. Philipp Stadler implemented parts of the program *ORPHEUS*, especially the object oriented interface to the mapping functions. Lionel Dorffner implemented a part of the object oriented interface for 3D modelling techniques, especially the tools for making 3D models persistent and the VRML export.

Last, but not least I want to thank Prof. Karl Kraus for always encouraging me to go on with this work, and Prof. Werner Schneider from the Institute of Surveying, Remote Sensing and Land Information University of Agricultural Sciences Vienna for being willing to act as my 2^{nd} supervisor.

Abstract

In this work, a new method for semi-automatic building extraction together with a concept for storing building models alongside with terrain and other topographic data in a topographical information system (TIS) is presented. The new approach is based on the integration of object parameter estimation into the photogrammetric process. Its main features are:

- A *hybrid modelling scheme* is applied for building extraction:
  - A user interface based on the principles of Constructive Solid Geometry (CSG) is provided. Each building can be de-composed into a set of simple primitives which are reconstructed individually. After reconstruction, these primitives are combined by Boolean set operators.
  - The internal data structure of both the primitives and the compound buildings is based on boundary representation.
- The primitives are provided to the user in a data base of common building shapes.
- The work flow for reconstructing one primitive consists of four steps:
  1. Selection of the primitive from the data base
  2. Interactive modification of the primitive parameters
  3. Automatic fine measurement
  4. Visual inspection of the results of the automated tools and interactive post-editing of the primitive parameters if required.
- In all these phases, the integration of robust parameter estimation and object modelling takes over a key role: Internally, both the whole building and the individual primitives are modelled by boundary representation. The specific properties of these boundary models are directly connected to parameter estimation in the photogrammetric process: the parameters of the building faces and the co-ordinates of the building vertices are determined simultaneously in a hybrid adjustment of both camera co-ordinate observations and *surface observations*. By the term "surface observation", we mean the observation that a certain point is situated on a surface corresponding to a building face.
- By a specific way of integrating the estimation of the surface parameters, a minimum set of such parameters for each building primitive has to be determined only, even though boundary representation is used for modelling. In addition, the data base of known building shapes can be expanded easily. The new technique is flexible enough to handle all building shapes which can be described as polyhedra.
- The automated modules for fine measurement of building primitives are an example for the application of a general framework for object surface reconstruction using hierarchical feature based object space matching. The role of the integration of object space into the matching process is again taken over by the modelling technique based on the concept of surface observations.
- As soon as a building has been reconstructed, it is inserted into a TIS. Management of both building and terrain (and, if available, other types of) data is based on the same principle even though different geometrical modelling techniques are used: the meta data of buildings and terrain models are managed in a relational data base with special features for topographical data. The actual data are contained in the data base as *binary large objects*. An object oriented interface is used for handling hybrid topographic data in visualization programs.
- The new method is integrated into the software developed at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology, especially *ORPHEUS* and *ORIENT* for monoscopic interactive measurement in multiple digital images and photogrammetric adjustment of hybrid observations. The part of the TIS is taken over by the program *SCOP.TDM*.

This thesis starts with an overview on existing systems for (semi-automatic and automatic) building extraction. After that, the theoretical background for the new technique in the fields of object modelling, data management in TIS, and parameter estimation is presented, and the new method as well as all of its components are described. In this context, special emphasis is laid on the description of our general framework for the automatic reconstruction of object surfaces which is also embedded in previous work in the field of image matching. Finally, the new method is evaluated in a test project in the Lower Austrian village of Stoitzendorf (image scale: 1:4500, focal length: 15 cm, 70% overlap, 50% side lap). In this test project, where most buildings are visible in six images, the automatic tool is shown to give results with an accuracy of $\pm$2-5 cm in the planimetric position and $\pm$5-10 cm in height if the roof edges are well-defined. The influence of the most important control parameters as well as critical configurations are discussed. The buildings of a part of the village are reconstructed to show the applicability of the whole process in an exemplary way.

Kurzfassung

In dieser Arbeit wird eine neue Methode zur halbautomatischen Gebäudeextraktion ebenso vorgestellt wie ein Konzept für die gemeinsame Speicherung und Verwaltung von Gebäudemodellen, Gelände- und anderen topographischen Daten in einem topographischen Informationssystem (TIS). Die neue Methode zur Gebäudeextraktion basiert auf der Integration der Schätzung von Objektparametern in den photogrammetrischen Auswerteprozeß. Ihre wichtigsten Merkmale umfassen:

- Anwendung eines *hybriden Modellierungsschemas* für die Gebäudeextraktion:
  - Es gibt eine Benutzerschnittstelle, die auf dem Prinzip der Constructive Solid Geometry (CSG) basiert. Diese Benutzerschnittstelle erlaubt es, jedes Gebäude zuerst in eine Menge von einfachen Primitiven zu zerlegen, die dann mit Hilfe Boolscher Operatoren kombiniert werden.
  - Die interne Datenstruktur sowohl der Primitive als auch der zusammengesetzten Gebäude basiert auf dem Prinzip der Modellierung durch Begrenzungsflächen (*boundary representation*).
- Die Gebäudeprimitive werden in einer Datenbank von häufig auftretenden Gebäudeformen zur Verfügung gestellt.
- Der Arbeitsablauf zur Rekonstruktion eines einzelnen Primitivs umfaßt vier Schritte:
  1. Auswahl eines Primitivs aus der Datenbank
  2. Interaktive Modifikation der Parameter des Primitivs
  3. Automatische Feinmessung
  4. Visuelle Inspektion und gegebenenfalls interaktive Nachbearbeitung der Parameter des Primitivs.
- In allen Phasen dieses Auswerteprozesses übernimmt die Integration von robuster Parameterschätzung und Objektmodellierung eine Schlüsselrolle: Intern werden sowohl das gesamte Gebäude als auch die einzelnen Primitive als Flächenmodelle dargestellt. Die spezifischen Eigenschaften dieser Flächenmodelle ergeben sich unmittelbar aus den Erfordernissen der Parameterschätzung im photogrammetrischen Prozeß. Die Parameter der Gebäudeflächen und die Objektkoordinaten der Gebäudeecken werden simultan durch eine hybride Ausgleichung sowohl von beobachteten Bildkoordinaten als auch von *Flächenbeobachtungen* bestimmt.
- Durch die spezifische Art der Integration der Schätzung der Flächenparameter muß nur eine minimale Anzahl von Parametern für jedes Primitiv verwendet werden, obwohl das sehr allgemeine Konzept der Modellierung durch Begrenzungsflächen angewendet wird. Zusätzlich kann mit Hilfe unserer Modellierungstechnik die Datenbank der Gebäudeprimitive sehr leicht erweitert werden. Die neue Methode ist flexibel genug, um alle Gebäude rekonstruieren zu können, die durch Polyeder beschreibbar sind.
- Das Modul für die automatisierte Feinmessung der Gebäudeprimitive stellt ein Beispiel für die Anwendung eines allgemeinen Konzepts zur automatisierten Rekonstruktion von Objektoberflächen dar, das auf der hierarchischen Anwendung von merkmalsbasierten Zuordnungsverfahren im Objektraum basiert. Wiederum wird die Integration des Objektraumes in den Zuordnungsprozeß mit Hilfe der oben beschriebenen Modellierungstechnik auf Basis des Prinzips der Flächenbeobachtungen erreicht. Unter “Flächenbeobachtung” wird dabei eine Beobachtung der Art verstanden, daß ein Punkt auf einer Fläche im Objektraum liegt, die ihrerseits einer Fläche des zu rekonstruierenden Objektes zugeordnet werden kann.
- Sobald ein Gebäude rekonstruiert worden ist, wird es in einem TIS verspeichert. Die Verwaltung von Gebäude-, Gelände- und, falls vorhanden, anderer topographischer Daten beruht auf einem einheitlichen Prinzip, auch wenn unterschiedliche Methoden zur geometrischen Modellierung verwendet werden: die

Metadaten werden in einer relationalen Datenbank mit spezifischen Erweiterungen zur Verwaltung topographischer Daten verwaltet. Die tatsächlichen Daten werden in der Datenbank als binäre große Objekte (*binary large objects*) behandelt. Für den Zugriff auf diese hybriden topographischen Daten z.B. in Visualisierungsprogrammen wurde eine objektorientierte Schnittstelle erstellt.

- Die neue Methode ist in die am Institut für Photogrammetrie und Fernerkundung der TU Wien entwikkelten Programme integriert, insbesonders in *ORPHEUS* und *ORIENT* für die monoskopische interaktive Messung in digitalen Bildern bzw. für die Ausgleichung hybrider Beobachtungen. Die Rolle des TIS wird von *SCOP.TDM* übernommen.

Diese Arbeit beginnt mit einem Überblick über bestehende Systeme auf dem Gebiet der (voll- und halbautomatischen) Gebäudeextraktion. Danach wird der theoretische Hintergrund der neuen Methode bezüglich der geometrischen Modellierung topographischer Objekte, der Datenverwaltung in TIS und der verwendeten Verfahren zur Parameterschätzung dargestellt. Es folgt eine detaillierte Beschreibung des neuen Verfahrens und aller seiner Komponenten, wobei besonders auf unser allgemeines Konzept zur automatisierten Rekonstruktion von Objektoberflächen Wert gelegt wird, das seinerseits wieder auf bestehenden Arbeiten auf dem Gebiet der Bildzuordnung beruht. Schließlich wird das neue Verfahren an Hand eines Testprojekts im niederösterreichischen Ort Stoitzendorf (Bildmaßstab: 1:4500, Kammerkonstante: 15 cm, 70% Längs- und 50% Querüberdeckung) evaluiert. In diesem Testprojekt, in dem die meisten Gebäude in sechs Bildern sichtbar sind, zeigte sich, daß das automatisierte Modul bei gut definierten Dachkanten Ergebnisse mit einer Genauigkeit von $\pm$2-5 cm in der Lage und $\pm$5-10 cm in der Höhe gibt. Der Einfluß der wichtigsten Steuerparameter des Verfahrens auf die Ergebnisse wird ebenso untersucht wie die Bedingungen, unter denen es scheitert. Weiters wurden für einen Teil des Ortes alle Gebäude unter Verwendung des hier vorgestellten Verfahrens rekonstruiert, um auf exemplarische Weise seine Anwendbarkeit zu demonstrieren.

Contents

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

In the past, topographical information systems (TIS) and geographical information systems (GIS) only contained 2D or 2.5D data even though photogrammetric data acquisition delivered 3D co-ordinates of points measured in a stereoscopical plotting device. With the enormous decrease of costs of both computational power and data storage capacities, fully 3D representations of real-world objects become applicable from an economical point of view. There is a great demand for 3D building descriptions, especially for 3D city models, 3D GIS and virtual reality models. These data are required for many applications, among others:

- Architecture: visualizations of planned objects for competitions
- City planning: providing visualization tools for proposed projects as a basis of decision-making
- Environmental planning: simulations of air distribution and air pollution
- Telecommunication: planning transmitter placement
- Tourism: 3D visualizations for public relations or as souvenirs.

However, the great demand for 3D data at a considerably high level of detail collides with the enormous costs of data acquisition for these purposes. That is why automation of these tasks is a currently very desirable and challenging yet equally difficult task because of the great number of possible building forms, even in a rather homogeneous cultural region such as Central Europe, and due to problems such as occlusions in densely built-up areas and noise in the primary data. Automation of photogrammetric data acquisition has become possible due to the application of digital image processing techniques in digital photogrammetry. Besides of the drawback that visualization tools for digital images still lack resolution in comparison with analogue photographs, there are several advantages connected with using digital images:

- For working with digital images, digital softcopy workstations running on standard computers can be used rather than expensive plotting devices requiring special hardware.
- Data transfer is simpler with regard to plotting results because the data can be post-processed on the same computer.
- Digital image processing techniques can be used for image enhancement.
- Digital image processing techniques render possible the automation of photogrammetric measurement tasks.

There is a significant difference in the potential for automation depending on the task to be solved. According to [Gülch, 2000] four levels of automation can be achieved by systems for digital photogrammetric plotting of cartographic features:

1. *Interactive systems:* There is no automation at all, all measurements are performed manually.
2. *Semi-automatic systems:* Automatic modules are integrated in a more or less interactive work flow. Interaction is required for scene interpretation and for providing approximate values for the automated modules.
3. *Automated systems:* The main tasks are performed automatically, user interaction is focused on project setup before the automatic phase and on visual inspection, post-editing and correcting errors of the results of the automated modules. Scene interpretation is either a part of the automatic process, or it can be neglected if only one object is expected to be visible in the images or if the influence of other visible objects is small (e.g. generation of digital terrain models in small scale topographic mapping).
4. *Automomous systems:* Fully automatic systems without a need for post-editing, working as a "black box". Such systems do not yet exist for photogrammetric applications.

Photogrammetric plotting can be split roughly into the following major steps:

1. Image orientation (section 4.5.2) to describe the geometry of the sensing devices.
2. Reconstruction of the object, which is usually split into three sub-tasks:
   (a) Object recognition: The scene visible in the images has to be interpreted and classified.
   (b) Measurement of corresponding points in two or more images of the same scene.
   (c) Estimation of the object parameters from the corresponding points.

Considerable efforts have been made in the recent years to automate all tasks in photogrammetric plotting on the basis of *matching* techniques. In most applications in the field of vision-based 3D reconstruction, two strategies can be applied:

- Data driven or bottom-up strategies: These algorithms start with low-level feature extraction from all images, and then typically search for homologous features in different images. There is no semantic meaning assigned to the features: Data driven processes do not care about which features are found as long as they really come from identical object features. Knowledge about the object depicted in the images is used for matching. However, it is not very detailed and often given implicitly in the algorithms, e.g. by certain smoothness assumptions about object surface.
- Model driven or top-down strategies: Features are extracted from the images, too, but in this case, a data base containing explicit model knowledge about the object(s) is provided, and the object model is to be adjusted to the data, i.e. model features have to be matched with image features.

Of course, data driven and model driven strategies can be combined in order to automate a certain task. The degree of automation achieved depends on the complexity of the task which is to be solved:

- **Automation of inner orientation:** The term "automation of inner orientation" is a synonym for the automation of fiducial mark measurement in metric cameras. This problem is a typical example of a model driven process. On the one hand, a camera model is required containing the positions of the fiducial marks and the borders of the camera body, and on the other hand, models of the fiducial marks are required. The positions of the fiducial marks are usually symmetric with respect to the axes of the image co-ordinate system. Automatic inner orientation basically consists of three stages [Schickler and Poth, 1996, Rottensteiner, 1993]:

1. Coarse location of the fiducial marks: Image pyramids (section 5.3) can be used for that purpose. If only one photograph per image data file is permitted, the search can start at standard positions (e.g. the image corners) [Schickler and Poth, 1996]. Another possibility is the detection of the camera borders by Hough transformation [Rottensteiner, 1993].
2. Fine location of the fiducial marks: At this stage, any target location algorithm can be applied. Usually, the task is solved by raster based matching techniques (cf. section 5.2.1) using a raster image depicting the fiducial shape as a template.
3. Determination of the camera pose: An analogue photograph can be put into a scanner in eight different ways (four different rotation states, geometrically positive or negative). A non-symmetric feature on the camera body has to be located for that purpose. From the results of the previous step, eight sets of transformation parameters can be derived, each giving an approximate position for the asymmetric feature. This feature is searched for in all those areas; the area with the best fit corresponds to the correct set of transformation parameters [Schickler and Poth, 1996].

- **Automation of outer orientation:** The automation of outer orientation comprises two steps which have to be treated in different ways [Heipke, 1997]:

  1. Automatic measurement of tie points: Being a typical data-driven process, this task is solved by multi-image feature based matching techniques (section 5.2.2): There is no emphasis on which features are detected as long as the same features are found in different images. Current procedures work well in near-normal case configurations as they appear in aerotriangulation (orientation of aerial images) [Tang et al., 1996]. In other situations, e.g. in the presence of great scale differences of the images or convergent viewing directions as they appear in terrestrial photogrammetry, they might fail. Automatic measurement of tie points is often referred to as "automatic relative orientation" [Heipke, 1997].
  2. Automatic measurement of control points: This is a model-driven process. Automatic measurement of targeted control points can be performed using similar techniques as in automatic location of fiducial marks, the difference being that the orientation of the target (i.e., its rotation) in object space is also unknown [Rottensteiner and Prinz, 1996]. Automatic measurement of non-targeted control points is very difficult because the structures of the models involved are different depending on the object which is used as a control point. Most algorithms aim at using one specific class of objects. [Schickler, 1992] gives an example for the automation of control point measurement using a country-wide control point database containing wire-frame models of houses which are matched to imaged data. Another example is given in [Drewniok and Rohr, 1996], where the control points are manhole-covers. In this case, the model of the control points is given implicitly as a radiometric model which is the basis for a search for candidates in the images. The candidates are matched to a GIS database containing all manhole covers of the given area. Automatic measurement of control points has been solved for certain classes of objects, but not yet in as general a manner as the measurement of tie points [Heipke, 1997].

- **Automation of object reconstruction for topographic mapping:** This problem has been tackled for various classes of objects, but a general solution in the sense that a semantic description of the scene can be derived automatically from multiple views has not yet been (and will probably never be) found. An incomplete list of tasks which have already been or are about to be solved with varying degrees of automation and robustness contains [Gülch, 2000]:

  – Automatic generation of digital elevation models (DEMs) for topographic mapping. The task has been solved for 2.5D grids by feature based matching techniques [Gülch, 1994, Krzystek, 1995]. Post-editing is still necessary, especially in built-up areas and forested regions. There are methods for automatic break line detection which are necessary for high-quality Digital Terrain Models (DTMs), but they are not yet very robust [Wild and Krzystek, 1996, Rieger et al., 1999].

  - Automatic and semi-automatic building extraction from aerial images: an overview about the state of the art in the field of automation of building extraction will be given in section 1.2.
  - Automatic road extraction from aerial or satellite imagery: This task is tackled by applying image segmentation techniques exploiting the specific reflectance properties of road surfaces for an initial determination of road candidate regions in geo-coded images. Depending on the resolutions of these images, roads are either considered to be narrow image lines, i.e., ridges and ravines in the grey level images, or narrow image regions being bordered by parallel lines. The road candidate regions are further evaluated using domain-specific knowledge about roads, both in order to eliminate false candidates and to connect road candidates which are separated due to segmentation errors so that, finally, a consistent network of roads is generated, e.g. [Baumgartner et al., 1999, Wiedemann and Hinz, 1999, Steger, 2000].

- **Automation of object reconstruction in close-range application:** Close-range photogrammetry is characterized by the fact that there is a wide variety of possible image configurations, and full automation can only be expected in special cases.
  - Automatic generation of models of industrial surfaces: This problem has been solved for smooth surfaces in a similar way as for DEMs, e.g. for quality checks in the car industry [Krzystek, 1995].
  - Automatic plotting of building facades: As building facades are often smooth surfaces, again the modules for DEM generation can be applied for automatically deriving 2.5D facade models.
  - Some efforts have been spent on the automation of the reconstruction of more complex object surfaces. The problem is hard to be solved in a general manner, but some automation has been achieved in semi-automatic systems integrating CAD techniques for object modelling (cf. section 1.2.3) or by specific structured light approaches, e.g. [Krattenthaler et al., 1993].

In section 1.2, an overview on the state of the art in the field of automatic and semi-automatic building extraction shall be given. This section will be followed by the statement of the problem we want to solve in this work (section 1.3), including an overview on the chapters that are to follow.

1.2 Building extraction: State of the art

According to [Brunn, 1998], automatic building extraction consists of two steps:

1. *Building detection* comprises methods to detect regions of interest for subsequent building reconstruction
2. *Building reconstruction* is the determination of the geometrical parameters of a building located in a given region of interest.

These steps cannot always be clearly distinguished because the first step may already make use of implicit geometric model knowledge and thus will already perform some tasks which could more or less belong to the second one. Various data sources can be used for these purposes. The most important ones comprise [Brenner, 1999]:

- Aerial images: they offer a high accuracy potential, but due to radiometric problems and due to occlusions, the automation of tasks related to building extraction turns out to be difficult, especially in densely built-up areas.
- Digital Surface Models (DSM): DSMs can be derived indirectly by image matching techniques from aerial or RADAR images or directly from laser scanner data. Data from laser scanner systems explicitly offer a dense (up to 9 points/m^2) 3D data source with less occlusions due to the small opening angles,

and apparently it is easier to separate the task-relevant information from the rest of the data than it is with aerial images. However, high resolution laser scanner data are still expensive, and the accuracy potential is not as high as it is with high-resolution aerial images.

- 2D GIS or map data: These data provide extremely accurate information for building detection and also give some hints for the reconstruction step. However, additional information is required to reconstruct the third dimension, and buildings not contained in the data base cannot be extracted from these data.

In all stages of building extraction, knowledge about buildings has to be used. Knowledge can be represented implicitly, e.g. by applying certain rules in order to detect buildings, or explicitly by providing a data base of explicit building models.

1.2.1 Building detection

All data sources mentioned above can be used for building detection purposes:

1. Digital images: Grouping of extracted image features in order to find rectangular structures can be applied as well as colour-based segmentation methods, e.g. [Baltsavias and Mason, 1997].

2. High-quality Digital Surface Models (DSM) can be analyzed using various techniques (see below).

3. 2D GIS information can be used to locate existing buildings, e.g. [Haala et al., 1997], [Haala et al., 1998].

4. In addition, interactive determination of the regions of interest for building reconstruction can be used as long as automatic building detection is not yet operational.

The most promising methods seem to be those working with high-resolution DSM. It is the idea of these techniques to classify all laser measurements into ground points versus off-terrain points reflected by roof tops, vegetation, cars and other objects. After that, a digital terrain model (DTM) is computed using only the points classified as terrain points. Finally, the DTM is subtracted from the DSM, which results in another DSM just containing the off-terrain points. In a second step, the off-terrain points have to be classified once more in order to separate building regions from vegetation and other objects. Two different groups of algorithms can be distinguished in this context: Mathematical morphology and DTM generation by robust parameter estimation.

Mathematical morphology: In [Weidner, 1997] and [Fritsch and Ameri, 1998], a morphological “opening” operation is applied for the classification of DSM points. The structural element is considered to be a cubic prism or a sphere. It is moved beneath the DSM so that it touches at least one (the lowest) point of the DSM. The opening of the DSM is the surface consisting of the highest points reached by any part of the structural element as it traverses the DSM. All points on this surface are classified as terrain points, the others (local height variations smaller in extent than the structural element) are considered potential building points. The size of the structural element has to be adapted to the minimum building extent which can be expected. [Brunn and Weidner, 1997] improve the results of morphological opening by providing a framework for fusing various data sources in a Bayesian network for building detection. They use different resolutions of the DSM to iteratively apply morphological classification, the classification results of subsequent resolution levels being combined by Bayesian rules, and the final results of the iterative morphological classification are again combined with existing 2D map data. In this way they overcome errors due to larger regions covered by vegetation.

Parameter estimation: In [Pfeifer et al., 1999b], a method for linear prediction is proposed for the interpolation of DTMs from laser scanner data. The error distribution of laser scanner heights with reference to the ground surface is no longer a assumed to be a normal distribution, but a skew distribution with a strong bias towards off-terrain elevations. In the first interpolation step, a rough surface approximation is determined. All points have the same influence. Thus, the surface obtained runs in an averaging way between the ground points and the off-terrain points. After that, robust iterative estimation by modulating weights according to a weight function of the residuals is performed. The weight function takes the skew error distribution into account. Due to the special form of the weight function, the interpolating surface runs nearer and nearer to the ground after each iteration. Iteration is stopped as soon as a threshold for the r.m.s. error of the terrain points is reached. After that, the classification is performed by thresholding the residuals. For a detailed description, see [Kraus and Pfeifer, 1998] and [Pfeifer et al., 1999a]. This algorithm was originally developed in order to separate vegetation from terrain points in wooded regions. In [Rieger et al., 1999], an example for the way this algorithm can be improved to derive a building mask from DSM in a resolution of about 1-2 m is given. Small areas corresponding to larger groups of conifer trees are eliminated by applying a despeckle filter. In order to use this algorithm for building detection in high-resolution DSM data, it has to be applied to the original data several times in a coarse-to-fine strategy because otherwise the error distribution of the points on the roofs would correspond to the error distribution on a steep hill, which would prevent the algorithm from successfully classifying these points as “non-terrain-points”.

1.2.2 Building reconstruction

The fact that there is a building in the region of interest is assumed to be answered by a previous detection step. In order to automate building reconstruction, the computer has to learn what a building is. Thus, a data base representing model knowledge about what a building is has to be made available. The models can be provided explicitly by giving a set of building primitives or it can be provided implicitly by declaring rules for extracting features from the original data and for grouping these features in different aggregation stages. According to [Brenner, 1999], building reconstruction can be subdivided into two phases which, however, can often not be clearly separated:

1. Structuring of the input data: This step comprises the extraction of relevant structures or features such as lines in digital images, surface discontinuities and/or co-planar regions in the DSM as well as applying certain grouping rules, e.g. searching for parallel and/or orthogonal lines.
2. Geometrical reconstruction: The structures extracted in the first step have to be combined to form consistent building models.

The first important question with respect to building reconstruction is connected with the internal representation of the buildings. Two types of modelling are commonly used [Englert, 1998, Brenner, 1999, Müller, 1998]:

1. Boundary representation (B-rep) of the buildings: The buildings are represented by their bounding surfaces together with their intersections and neighbourhood relations [Mäntylä, 1988, Koehl, 1997]
2. Constructive solid geometry (CSG): The buildings are represented by a set of (volumetric) primitives, each of them described by a small set of parameters such as position, orientation, length, width, height, etc. [Mäntylä, 1988].

Another central issue is the representation of knowledge for building extraction. There are three possibilities for providing domain specific model knowledge for that purpose [Brunn, 1998]:

- *Parameterized models*: Basic primitives such as hip roof or saddle back roof buildings. The topology of these primitives is provided by a data base containing all available primitive types. Only the geometrical parameters have to be adjusted.

- *Generic models*: These models do not define explicitly a building shape but rather offer the possibility to formulate consistent object models using one of the modelling types given above. Depending on the level of detail which can be described by generic models, several examples can be found [Brenner, 1999]:
  - Prismatic model: a prismatic building is characterized by two horizontal planes (roof and bottom) and a set of n vertical planes (walls).
  - Polyhedral model: a very general model which can be used to represent all types of building bordered by flat surfaces. Usually this type of model is given as a B-rep.
- Combinations of simple primitives in order to model more complex buildings. This means that "glueing tools" and/or Boolean operators for building primitives are required [Englert, 1998]. The concept of combining primitives by Boolean operators corresponds to modelling the buildings by CSG.

Building reconstruction from single data sources: In automatic building reconstruction, data driven and model driven techniques have to be combined. Different techniques can be distinguished by the data they use as an input for structuring, by the type of features they extract in the structuring process and by the way they represent the model knowledge about buildings (implicitly or explicitly).

In [Lang, 1999], the feature aggregates are 3D corners with neighbouring edges which are combined by applying a model data base containing typical building shapes. [Henricsson et al., 1996] aggregate the bounding edges of homogeneous image patches and derive 3D planar surface blobs which are combined by a consistency check considering the neighbourhood relations of adjoining blobs.

In [Baillard et al., 1999], 3D straight line segments are reconstructed from multiple views. For each straight line segment, a hypothesis for a plane containing that line is generated, and the tilt of the plane is computed using a correlation method based on homographies between two images and the plane. Planes without sufficient support are eliminated, and from the rest, roofs are constructed using rules for grouping of planes. In [Fritsch and Ameri, 1998], DSMs are analyzed using the Gaussian surface theory to perform classification of DSM pixels according to their mean and Gaussian curvature signs. Pixels classified to be "flat" are connected by region growing. The flat regions thus extracted give hypotheses for planar surfaces which have to be combined by a generic technique [Ameri and Fritsch, 1999].

Building reconstruction and data fusion: In most recent approaches to the problem of building extraction, *data fusion* becomes more and more important. Data from various sources can be combined in order to overcome the drawbacks of specific sensor types. [Haala et al., 1998] and [Brenner, 2000] fuse existing 2D map data with DSM: The map data provide very precise information about the building outlines (which is badly defined even in high-resolution laser scanner DSMs), but the bounding polygons may have a very complicated shape. That is why the shape of each polygon has to be analyzed in order to either partition its enclosed area into more simply shaped regions which can be used as base polygons for simple building primitives or in order to create hypotheses for possible roof planes which can be estimated from the DSM data.

2D map data are also combined with digital images, e.g. [Paško and Gruber, 1996, Suveg and Vosselman, 2000]. In [Suveg and Vosselman, 2000], a method for partitioning the 2D map data into rectangles is applied, too. Each rectangle is then used as the basis of a building primitive, and the primitive is fitted to the image data. As several primitive types are available, the type having either the best fit or the greatest support in the images has to be chosen, or a priori information about the primitive type can be used.

DSM and image data can be combined, too, e.g. [Ameri, 2000]. Based on the 3D information derived from the DSM, a building model can be created. After that, these models are back-projected to the images, where their edges can be matched with image edges, and the parameters of the building models can be derived using the image information, which considerably increases their accuracy, especially with respect to the building outlines.

1.2.3 Semi-automatic building extraction

Due to the complexity of the task, fully automatic building extraction is not yet operational. That is why semi-automatic systems are currently being developed [Englert, 1998, Müller, 1998, Gruen and Wang, 1998, Veldhuis, 1998, Rottensteiner, 2000] which offer a compromise between the great demand for automation in data extraction and the fact that the problem has not yet been completely solved. In semi-automatic systems, recognition and interpretation tasks are performed by the human operator, whereas modelling and / or precise measurement is supported by automated tools. In a more general context, the term *CAD based photogrammetry* is used for such types of systems [van den Heuvel, 2000], which emphasizes the fact that the principle can also be applied to other types of objects than buildings, e.g. in architectural photogrammetry [Streilein, 1999] or for the reconstruction of pipes in industrial environments [van den Heuvel, 2000]. Historically, photogrammetry was considered to be a tool for precise measurement of 3D co-ordinates of points, which were, in many applications, considered to be an end product. Nowadays, especially in the context of building extraction, the creation of a consistent 3D model of the object is aimed at. CAD is a powerful tool for creating such 3D models. If 3D models of existing objects shall be created, photogrammetry and CAD can be combined in two ways [van den Heuvel, 2000]:

1. Photogrammetry interfaced with CAD (figure 1.1): In this context, photogrammetry is considered to deliver a structured point cloud which is then transformed to a CAD system, where the 3D description of the object is created using CAD tools. The degree of automation in such systems can vary, and so does the complexity of the "interface" in figure 1.1. As photogrammetric measurement is usually performed on standard equipment such as analytic plotters or digital stereo workstations, stereoscopic view can be employed which considerably simplifies scene interpretation by the human operator. There are several examples for such systems employing different degrees of automation:

   - The *CyberCity modeler* [Wang and Gruen, 1998, Gruen and Wang, 1998] is a very advanced tool for the generation of 3D city models. It is based on interactive measurement of 3D points in some stereo measurement device. In the measuring process, the points are given labels in order to distinguish points on the 2D boundary of the building roofs from other object points. From these data, the *CyberCity modeler* can automatically generate the full topology of the buildings by a graph-based consistent labelling technique. The 3D building models can be visualized and edited in a post-processing step. After that, they are transformed into a specific internal data format for 3D vector data based on B-rep. The data are stored in a relational data base.
   - The technique for the generation of 3D city models described in [Koehl, 1997] and [Koehl and Grussenmeyer, 1998] is also based on 3D points measured in some measuring device. After that, building edges describing a wire frame model of the roof are defined in the CAD system. The edges are labelled as belonging to the eaves, to the roof top, to other edges of the roof outline or to none of these classes. From that information, the full 3D B-rep model is created automatically. The walls are automatically added to the model as vertical planes, and they are intersected with an existing DTM. The data can also be stored in a relational data base.

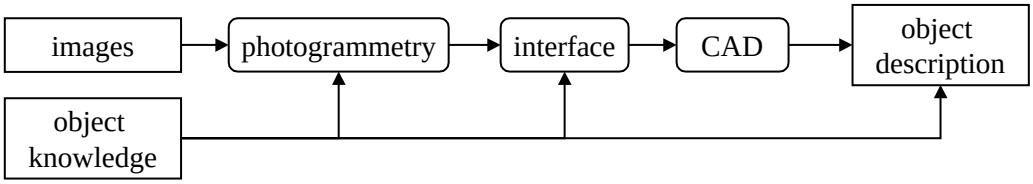


Figure 1.1: Photogrammetry interfaced with CAD; taken from [van den Heuvel, 2000]

2. Integration of photogrammetry and CAD (figure 1.2): The CAD model is directly integrated in the measurement process, and the mathematical model of the photogrammetric process directly contains

the relations between the image measurements and the parameters of the CAD model. Again, different degrees of automation appear to be possible. One way of integrating photogrammetry and CAD is given by offering tools for the generation of topology in an interactive environment. This is done in PC based photogrammetric systems such as the *PhotoModeler* [PhotoModeler, 2000] or *ORPHEUS* [Kager et al., 2000]. Object parameters can be determined after the photogrammetric process, e.g. by estimating surface parameters from 3D points (*PhotoModeler*), or their determination can be integrated into photogrammetric adjustment in order to support image orientation or to enforce object constraints (*ORPHEUS*). More sophisticated systems provide domain-specific a priori object information and automated tools for precise measurement which can be used to speed up the data acquisition process.

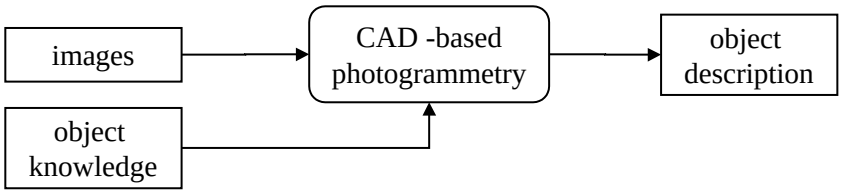


Figure 1.2: Photogrammetry integrated with CAD; taken from [van den Heuvel, 2000]

Semi-automatic techniques for building extraction offered by the latter group of CAD-based photogrammetric systems provide a data base of known building primitive types typical for a certain cultural region, and they offer tools for the automation of precise determination of the parameters of these primitives as well as for efficient interactive adaptations. "Complicated" building shapes can be created by combining primitives, as suggested in section 1.2.2. The work flow for the reconstruction of a certain building primitive looks as follows, e.g. [Müller, 1998]:

1. Interactive selection of an appropriate building primitive from the knowledge base by the human operator.
2. Interactive determination of approximations for the building parameters.
3. Automatic fine measurement and adjustment of the building parameters to the image data.
4. Visual inspection of the matching results and interactive editing in case the automation tools failed to achieve correct results.

In this group of systems for semi-automatic building extraction, a top-down strategy is applied: model knowledge is provided by the user who is responsible for a correct scene interpretation, whereas tedious measurement tasks are performed automatically. As model knowledge is given by the primitives which have to be adjusted to the data, the interactive part of the work flow is similar to working in a CAD system. Existing systems for semi-automatic building extraction based on that principle can be characterized according to several items:

- Representation of domain specific model knowledge: CSG is generally used in this type of semi-automatic systems. However, the systems differ by the primitive types they can use. In [Veldhuis, 1998], only flat rectangular and saddle back roof buildings are used. The system developed at the University of Bonn offers several basic primitives such as pyramids, tetrahedrons and boxes as well as two "combined primitives" representing a saddle back and a hip roof building, respectively [Englert, 1998, Müller, 1998]. In addition to the primitives, simple generic building types such as vertical prisms can be offered as well.
- Internal representation of building models: Besides the CSG interface for the user, the internal structure of the building models can either be based on CSG or B-rep. In addition, both modelling techniques can be used for the internal representation of the primitives. [Englert, 1998] uses CSG as the central data structure, the CSG tree of a building being converted to B-rep as soon as the building is declared to be completed by the user. B-rep conversion is required especially for visualization purposes.

- Formulation of the relations between observable entities and the building parameters: This question is closely linked to the one of the internal representation. The user can determine camera co-ordinates in digital images, from which the parameters of the object have to be determined. Just the same, feature extraction algorithms can determine image entities by their camera co-ordinates. If B-rep is used as the internal structure of the building primitives, there has to be a connection between the parameters of the building faces and the camera co-ordinates, whereas in the CSG case, the primitive parameters can be directly linked to the camera co-ordinates. There are also differences with respect to what is considered to be the observed quantity. With respect to building vertices, their camera co-ordinates are generally assumed to be observed, which is used for estimating the building parameters in the interactive phase. With respect to image edges corresponding to edges of the building, either the camera co-ordinates of the vertices and end-points of the image edges or their orthogonal distance from the approximate building edge projected to the image can be considered to be observed. [Veldhuis, 1998] gives a comparison between the application of two B-rep and a CSG primitive for automated fine measurement. As the number of parameters involved in the latter case is much smaller than the one in the first case, he observes a better convergence behaviour of the CSG solution. However, in that work, object constraints on the B-rep model are considered by additional constraint equations in adjustment whereas they could be considered explicitly by the formulation of the surface equations [Rottensteiner, 2000].

- Presentation of the relations between observable entities and the building parameters to the user: This is a question of providing a good graphical user interface. In order to reduce the number of user interactions required and in order to avoid tedious searching in menus which might be difficult to survey, simple rules for assigning changes of building parameters to user interactions have to be found.

- Techniques for automatic fine measurement: This is the item by which existing systems for semi-automatic building extraction differ most. In all systems, object-to-image matching techniques are applied: the edges of the building primitives are matched to extracted image edges. The way this is done differs considerably as well as the way the image edges are extracted and represented in the reconstruction process does. [Veldhuis, 1998] projects the building edges back to the images where they are fitted to edge pixels (i.e., pixels having a great magnitude of the grey level gradient) in a least squares adjustment. The orthogonal distance of these pixels from the approximate edges are considered to be the observations in the least squares fit, the unknowns being the building parameters which are directly related to the observations. In [Müller, 1998], raster based image matching techniques are used for the determination of the roof top heights whereas feature based matching based on clustering and RANSAC techniques are applied for a robust determination of the shape parameters of the building.

According to [Gülch, 2000], a system for semi-automatic extraction of topographic objects should provide the following functionality in order to make it applicable in a digital photogrammetric production line:

- Functionality in common with image processing systems:
  - Management of large amounts of image data and multi-image processing tools.
  - Management of auxiliary data such as orientation parameters of the sensors involved.
  - Display management: Multi-image display at arbitrary scales is required. Extracted objects as well as intermediate results of object extraction have to be super-imposed to the digital images. 3D visualization is helpful for data analysis.

- Functionality in common with GIS systems:
  - Management of data structure: A connection to standard GIS or CAD systems has to be made available.
  - Interactive tools for data capture are required for on-screen digitization for providing approximate values and for the acquisition of objects which cannot be captured automatically.

  - CAD tools for interactive editing such as object destruction, displacement, and connection to existing objects should be available.

- Specific functionality:

  - Automated extraction tools have to be available. A real-time or at least a near real-time performance is required. Automatic extraction can start from the approximate positions provided by the user, and it can stop according to some quality criteria or on user demand.
  - It should be easy for the user to switch between automatic and interactive modes of operation.
  - New objects should be connected to existing objects automatically.
  - An "Undo"-function is important to go back to initial values if automatic extraction has failed.
  - Multiple resolutions of images should be used both for viewing and, e.g., feature extraction.
  - Self-diagnosis of the automated tools. Quality measures should be offered to the user to inform the operator about critical zones.

The applicability of such systems has been evaluated in several test projects, e.g. [Englert and Gülch, 1996].

1.3 Problem statement

In this work, we will describe a new method for semi-automatic building extraction from digital aerial images. It is the goal of the new method to create a sound basis for an operational tool for the acquisition of buildings for topographic information systems. A prototype for such a system will also be described in this work. The method has to be based on a rigorous mathematical background with respect to the integration of object reconstruction and the photogrammetric process as well as with respect to the statistical techniques applied for robust parameter estimation, and the automated tools shall be designed in a way general enough to be used as a plug-in to other (fully automatic) systems in the future.

The level of detail which shall be achieved by the new method corresponds to the level of detail required for large-scale topographic mapping: the roof shapes as well as larger roof structures such as dormers have to be included in it, but roof overhangs, small structures such as small chimneys and structures on the facades can be omitted. The accuracy requirements for the building parameters are in the range of $\pm$ 5 - 10 cm. The data structure should enable the country-wide management of 3D building together with terrain data in a hybrid TIS. With respect to its functionality, the system should fulfil the requirements described in section 1.2.3.

The new system is to be based on the principle of integrating CAD functionality into the photogrammetric process in the sense depicted in figure 1.2. The core of the system will be a unique way of modelling based on B-rep and the integration of the hybrid photogrammetric adjustment system *ORIENT* [Kager, 1989]. This program offers possibilities for the integration of the determination of object shape parameters and 3D point coordinates from observed image co-ordinates. This integration is based on the principle of "surface observations" in the sense that points can be observed to be situated on a (user-defined) surface in object space. 3D object modelling by B-rep on the basis of "surface observations" and the system *ORIENT* will be used in all phases of the work flow:

1. *Knowledge representation:* Domain specific model knowledge about buildings will be provided in a data base of building primitives. In addition to that data base, generic types (vertical prisms) have to be provided by the system. We will show that we can avoid the over-parameterization which is attributed to B-rep [Veldhuis, 1998] by an implicit formulation of symmetry assumptions so that in the end only a minimum set of parameters is required for each building primitive.

2. *Interactive editing of building primitives:* It should be possible to edit the parameters of the building primitives by simple graphical tools: the user can digitize building vertices in one or more digital image(s). After each user interaction, the building parameters and the co-ordinates of the building vertices will be determined by simultaneous adjustment of the image co-ordinates digitized by the user, the surface observations provided by our modelling scheme and observed building parameters, the latter having to be used for avoiding singularities. Robust estimation techniques will be applied to find out which building parameters can be determined from the information provided by the user.

3. *Automatic fine measurement:* Our modelling technique based on B-rep and the principle of surface observations will be used for the evaluation of hypotheses of correspondence between building edges and edges extracted from digital images. In order to separate false correspondence hypotheses from correct ones, robust estimation techniques have to be applied.

From the point of view of solid modelling, a *hybrid scheme* will be applied: by using building primitives which can be combined by Boolean operators, a CSG interface has to be provided for the user. However, the internal representation of both the primitives and the compound building being created from a set of primitives will be in B-rep. We will show that this hybrid approach offers several advantages which we consider to be important:

- *Simplicity:* Working with primitives (and a few generic building types) is quite common in CAD systems, so that the CSG interface provides a familiar working environment to the user.

- *Generality:* A great variety of possible building shapes can be described by B-rep even if we restrict ourselves to using planar surfaces at the moment. A modelling system on the basis of CSG cannot be better than the data base of primitives, but our method will handle simple generic building types, too, an approach which might be expanded in the future.

- *Extensibility:* In our system, the building primitives have to be considered to be data. This means that it will be very simple to introduce new building primitives without any programming effort, and by the properties of the way the surfaces are mathematically formulated, these primitives can be formulated using a minimum set of parameters. It must also be possible to declare a certain compound building shape created in the course of the work flow to be a new primitive.

- *Applicablility in another context:* We want to emphasize the fact that our method must in no way be restricted to being applied in the context of a semi-automatic work flow. If an approximate building model is available in object space, the automated modules can be applied to it in order to adjust the building parameters to the image data, for instance, to improve the accuracy of a model created from another data sets such as a DSM.

With respect to automatic fine measurement, a general framework for object surface reconstruction based on the same principle of surface observations as our modelling technique for buildings has to be applied. The automated tools for building extraction will be shown to be just one specialization of a general principle of surface reconstruction by hierarchical feature based matching in object space. In the process of object reconstruction, the hypotheses generation and verification paradigm should be applied, hypotheses verification being performed by robust estimation techniques as they are offered by *ORIENT*. At the image scales we are interested in, occlusions and shadows influence the performance of common matching techniques. That is why we consider the application of more than two images for matching to be of crucial importance. We also want to emphasize the importance of applying coarse-to-fine techniques on the basis of digital image pyramids in order to make the performance of the automated tools more robust.

The prototype of our system for semi-automatic building extraction has to be integrated into the program *ORPHEUS* for digital photogrammetric plotting. *ORPHEUS* provides a graphical user interface (GUI) for *ORIENT* as well as modules for visualization of and monoscopic interactive measurement in multiple digital

images. It is capable of handling large amounts of image data and thus fulfils all the required functionality with respect to image processing and visualization as discussed in section 1.2.3.

Buildings are constructed on the earth surface. At the level of detail we are interested in, roof overhangs are not a part of the model even though this limitation is not caused by our modelling technique. The floor points of the buildings cannot be measured directly. We want to integrate existing DTM data in our work flow in the way that the vertical walls are intersected with the terrain. We consider both terrain and buildings to be an integral part of a description of the real world in a region of interest which are, however, represented by different modelling techniques. This has to be reflected in a possible solution for handling both building and elevation data of a whole country in a TIS. The efficient management of country-wide elevation data has been solved in the program *SCOP.TDM* [Hochstöger, 1996]. In this program, DTMs are treated as binary large objects (BLOBs), the meta data of each DTM being managed by a relational data base with additional topological data types. In this work, we will expand that principle to other types of topographic objects, in our special case to buildings, in order to be able to manage 3D building data at a national scale. Further types of topographic objects such as roads can be included into that concept in the future. The integration of *SCOP.TDM* should provide the TIS functionality required for a system of building extraction according to section 1.2.3.

In order to evaluate our prototype system for semi-automatic building extraction, a test project was carried out in the Lower Austrian village of Stoitzendorf. The buildings of a part of that village had to be reconstructed using our new system. With respect to the performance of our automated tools, we want to evaluate both their reliability and the accuracy of their results as well as the influence of some important control parameters on the results. In addition to that, the applicability of the whole system shall be evaluated in order to obtain an estimation of the efforts required for the generation of 3D building models for TIS using our system.

1.3.1 Comparison to related work in the field of automated building extraction

Our work has to be seen in the context of CAD-based semi-automatic building extraction techniques as they were described in section 1.2.3. With respect to the application of a CSG interface using building primitives and with respect to the work flow for the reconstruction of a building primitive, it can be compared to the works of [Veldhuis, 1998] and [Englert, 1998]. However, it differs from these systems and other work in the field of semi-automatic building reconstruction in various ways:

- *Internal representation of the building models:* We will use a hybrid modelling technique: a CSG user interface will be provided, but the internal representation has to be completely based on B-rep. In other systems for semi-automatic building extraction based on the integration of CAD into the photogrammetric process such as the one described in [Englert, 1998], CSG is used as the central data structure in the reconstruction process, the CSG tree being converted to B-rep in a post-processing step. In our work, B-rep is the central data structure which is updated immediately as soon as a new primitive is added to it using a Boolean operator.

- *Parameterization of boundary representation:* As stated above, [Veldhuis, 1998] claims that parametric building primitives are better suited than B-rep for the purpose of automatic reconstruction due to the over-parameterization of B-rep if general plane surface equations are used. He used additional constraint equations to enforce geometrical constraints such as orthogonality or symmetry between the faces of the building primitives. We will show a way how these geometrical constraints can be established just by a proper formulation of the plane equations, which leads to the fact that our primitives in B-rep are described by a minimum set of parameters.

- *Adapting building parameters in interactive editing:* [Müller, 1998], in his description of the system $HASE^{+}$ developed at the University of Bonn (see also [Englert, 1998]), shows how building parameters can be directly related to changes of the camera co-ordinates of a certain building vertex. For that purpose, certain rules have to be established in order to give answers to two questions:

1. Which building parameters are modified in dependence on the vertex changed by the user?
2. How can an operator declare which parameters are actually to be changed if that relation is not unique, i.e., if by changes of one vertex several parameters might be influenced?

In our approach, an operator can directly measure the vertices of the building primitive in the images, and the question which parameters are affected by a user interaction is answered by robust hybrid parameter estimation.

- *Automatic fine measurement:* [Veldhuis, 1998] uses a direct mathematical relation between the unknown object parameters and the observations, i.e., the orthogonal distances of edge candidate pixels from the projected object edge. The partial derivatives of these mathematical relations are computed numerically. In [Müller, 1998], different automation tools are applied for different parameters of the building primitives. The height of the roof tops can be determined by raster based matching techniques whereas the shape parameters of the building are determined by feature based matching on the basis of clustering and RANSAC techniques for robust parameter estimation. These robust estimation techniques are restricted to problems involving a small set of parameters only. Our technique of feature based matching of image and object edges is based on robust estimation by iterative re-weighting observations. The observations are the camera co-ordinates of the polygon vertices of image edges and surface observations for all these points in object space. There is no direct link between the surface parameters and the image co-ordinates as it is the case in [Veldhuis, 1998], but the link is given implicitly by the fact that a point appears in an image as well as in two surfaces (at their intersection). From the idea of using surface observations, our approach can be compared to the work of [Ameri, 2000] who uses a similar concept for enhancing the results of building extraction from a DSM. However, in that approach, the orthogonal distances between image edge vertex and the projection of the object edge are considered to be observed, and the edge parameters and vertex co-ordinates in image space are determined by least squares adjustment together with the object space parameters. We only determine object parameters because we consider the camera co-ordinates of the image edge vertices to be the observed values in image space. There is also a difference in the representation of the digital images in matching: [Ameri, 2000] uses edge elements which are not connected by edge tracking whereas we perform edge tracking and thus use the information of aggregated image edge segments. From the point of view of image representation, our work is related to [Lang, 1999] in her approach for reconstructing 3D building vertices by feature based matching because we adapted the feature extraction techniques described in that work and in closer detail in [Fuchs, 1998]. However, there is a difference in the way the object is modelled because in [Lang, 1999], the relevant object structures are the edges whereas in our system, an edge is considered to be the intersection of two surfaces, and that information is only used implicitly in the way parameter estimation is performed.

- *Integration of a DTM:* The system described in [Müller, 1998] offers the possibility to determine the height of a terrain point identified by the user by raster based matching, and the building is cut off at that height by a horizontal plane. The terrain is represented by a triangulated irregular network of the floor points of the buildings. [Koehl, 1997] and [Koehl and Grussenmeyer, 1998] use a DTM to project the border vertices of the roofs onto it in order to automatically complete the topology of buildings after having provided the roofs as an input. We also use a DTM, but in our system, the topology is already assumed to be known, and the DTM is used to compute the heights of the ground points only.

- *Integration of a TIS:* The results of building extraction are often given in a format readable in conventional CAD programs. This is convenient for single projects, but it can not be considered to be appropriate for handling the 3D building data of, e.g., a whole state. For providing building data on a national scale, these data have to be managed by a TIS. [Wang and Gruen, 1998] use a relational data base for managing building as well as elevation and image data. However, in standard relational data bases, queries by geometrical attributes are rather slow. [Yang et al., 2000] also applies a relational data base, but in this work, the geometrical data of the buildings are contained as BLOBs, and there is an additional indexing scheme enabling fast access by geometrical and/or topological criteria. In our system, the buildings are

also transferred to a relational data base with additional geometrical and topological data types as BLOBs. The meta data, e.g. the 3D bounding box, are managed by the relational data base which offers quick tools for data selection according to geometrical criteria. Our system is an expansion of the principle applied for managing elevation data on a national scale in the program system *SCOP.TDM*.

1.3.2 Outline of this work

There are three major goals of this work:

1. We want to give an overview about the theoretical foundations of all fields relevant for our work, especially for topographic modelling, TIS, hybrid robust parameter estimation and automatic object surface reconstruction.
2. We want to describe our system and all its components in the context of the above theoretical foundations.
3. We want to test our system in a realistic test project in order to evaluate its performance with respect to
   (a) its applicability in the overall process,
   (b) the reliability and the accuracy of the results, especially those of the automated tools.

Part II is dedicated to the theoretical background of topographic modelling, topographic data management and the acquisition of topographic data by photogrammetry. In chapter 2, different topographic modelling techniques are discussed. We do not restrict ourselves to solid modelling which is relevant for the 3D representation of buildings, but we also will describe techniques used for the representation of DTMs. We do so not only because we use DTMs for determining the heights of the ground points of buildings, but also because we want to achieve a hybrid description of the earth surface together with man-made objects situated on it, the description thus consisting of different types of topographic objects which should all be stored in a TIS.

Management of hybrid topographic data in a TIS is one of the main topics of chapter 3. That chapter begins with an overview on existing TIS technology. After that, the properties of the program system *SCOP.TDM* we use for data management are described. We also will explain our expansion of the principle for the management of elevation data on a national basis applied in *SCOP.TDM* to hybrid object representation schemes.

Chapter 4 deals with photogrammetric data acquisition. In that chapter, we will formulate photogrammetric data acquisition as a parameter estimation problem, and we will do so by presenting the properties of the hybrid photogrammetric adjustment system *ORIENT*. This implies a discussion about *ORIENT*'s mathematical and stochastic model as well as a more general discussion about least squares adjustment and robust estimation techniques for the detection of gross errors. The mathematical model of parameter estimation includes the integration of the estimation of object surface parameters into the photogrammetric process, which is one of the most important theoretical issues for our technique of modelling buildings in the reconstruction process. One of the reasons for the great flexibility of *ORIENT* is its data structure, which has to be known in order to understand the applicability of that adjustment program for our purposes. In chapter 4, we will also present the program *ORPHEUS* which is our major working environment for semi-automatic building extraction. The chapter will be closed with a consideration of some practical aspects of photogrammetric data acquisition.

Having described some theoretical foundations of acquisition, modelling and management of topographic data in the previous chapters, part III is dedicated to the automation of data acquisition in digital photogrammetry. Chapter 5 deals with the automatic reconstruction of object surfaces by means of matching techniques. It contains an overview on existing technology in the fields of feature extraction, image matching and the application of image pyramids in object reconstruction. After that, our general framework for object reconstruction is explained. This framework is based on hierarchical multi-image feature based matching in object space. In feature based matching, the paradigm of generation and evaluation of correspondence hypotheses is applied. Hypotheses evaluation is performed using *ORIENT*'s tools for robust estimation. The integration of object space is done

by using *ORIENT*'s techniques for the estimation of object parameters. Quite some space is dedicated to the way this can be achieved. The application our framework to a specific example, i.e. the generation of DEMs for small-scale topographic mapping, concludes chapter 5.

Chapter 6 describes the central issue of this work, our system for semi-automatic building extraction, on the basis of the foundations laid in the previous chapters. A very important aspect of our method is the way knowledge about buildings is represented in the extraction process, which will be explained in great detail. The overall work flow will be described as well as the work flow in the individual phases of building extraction. This comprises a description of the interactive steps in the work flow, but also the details about the matching process which is another specific application of the framework described in chapter 5. In chapter 7, we will present the results of our test project. Most emphasis will be laid on an analysis of the properties of the automated tools in the reconstruction process, but we will also demonstrate the applicability of the overall system in an exemplary manner.

Finally, in part IV consisting of chapter 8 only, we will present some conclusions of our work, and we will give an outlook on future work to be done in order not only to improve our system, but also to increase the degree of automation in building extraction by integrating other sensors.

Part II

Basic Concepts

Chapter 2

Modelling of topographic objects

It has already been stated in chapter 1 that modelling techniques are a central issue in building extraction. It has also been stated in that chapter that it is our goal to create a description of the earth surface together with other types of objects situated on the terrain, and that this description should be stored in a TIS. That is why we are not only interested in modelling schemes appropriate for buildings (and other types of man-made objects), but we also want to integrate a description of the terrain, thus, we also need modelling techniques for that type of data. So we will have a more general look on techniques for modelling topographic objects in this chapter before we will concentrate on digital terrain models and, even more relevant for the task of building extraction, on modelling techniques for solid objects.

The basic entities contained in topographic data bases are called *topographic objects*. Objects having similar properties are described by the same *object class*. Each topographic object is a specific instance of a certain object class. By the term *object definition*, the description of the attributes of an object class which are common to all instances of that class is meant. Topographic objects have different types of properties (cf. figure 2.1; [Kraus, 2000, Bill and Fritsch, 1991]):

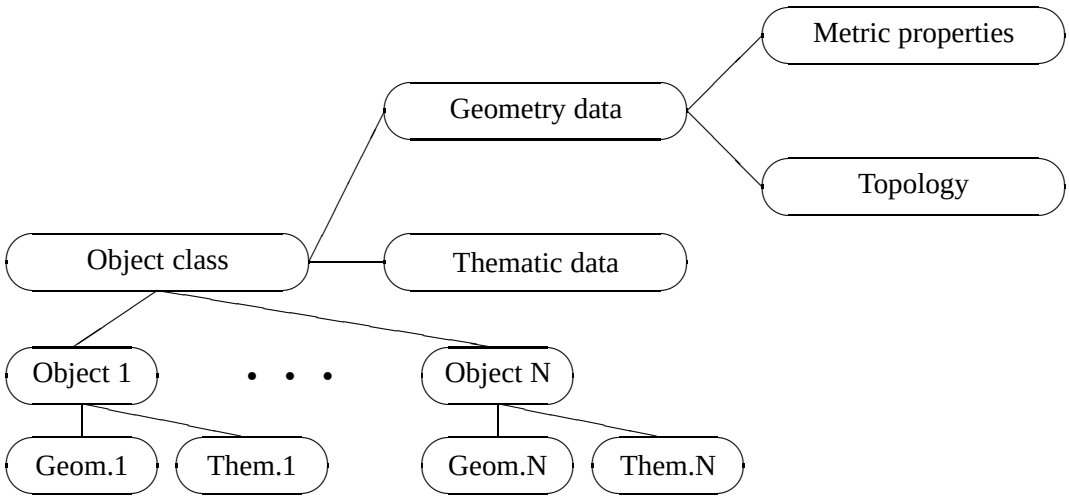


Figure 2.1: Topographic object definition according to [Kraus, 2000].

- *Thematic data* related to the object
- *Geometric data* describing both form and pose of the object in a given object co-ordinate system. The geometric data can be sub-divided into two separate data sets:
  - *Metric properties* are usually described by the co-ordinates of points and/or parameters of curves, surfaces, etc., in the object co-ordinate system.

  - *Topological properties* describe the neighbourhood relations between these points as well as those between an object and other objects.

- Each instance *Object i* ($0 < i < N$) is assigned a *unique identifier* (key).

Our interest is concentrated on the representation of object geometry so that we will keep aside the thematic aspect. The internal representation of the geometric properties of topographic objects depends on both the object type and the context in which the objects of a topographic data base are to be used, i.e., the applications of the topographic data base. In the following section, we will give an overview on representation schemes of the geometrical properties of 3D topographic objects.

2.1 Geometric modelling

According to [Bill and Fritsch, 1991], *geometric modelling* is understood as description, processing and storing of the geometrical properties of spatial objects using analytical or approximating methods. Depending on the object class and the tasks to be solved, various modelling techniques have been introduced. These modelling techniques can be classified according to several criteria as will be described in the following paragraphs. Overviews on geometrical modelling techniques can be found in [Bill and Fritsch, 1991], [Samet, 1989] or in [Mäntylä, 1988].

Analytical vs. approximating techniques: Analytical modelling techniques are based on closed analytical surface functions or volumetric primitives whereas approximating modelling techniques are based on interpolation methods or on methods of approximation by finite elements [Bill and Fritsch, 1991].

Vector vs. raster data: A basic classification scheme of geometrical modelling techniques is based on the internal representation of the data. Topographic objects can be represented by:

1. *Vector data:* The representation of the objects is based on distinct points described by their co-ordinates in the reference system and their topological relations, especially edges (connections of two points) and surfaces (e.g. represented by closed loops of edges, see below). Vector representations are very compact and thus do not require much disk space. In addition, operations such as geometrical transformations or visualization can be performed rather fast. However, in some situations, e.g. in the context of the union of thematic layers, more complex algorithms have to be applied than with raster data defined on a common grid.

2. *Raster data:* The representation of the objects is based on the elements of a (2D or 3D) matrix. The geometry of such an element (a *grid point* or *pixel*) is given by the row and column indices of that element, the offset of the first (e.g. the upper left) pixel of the matrix, and the grid interval. There are two views on the meaning of a pixel in a raster model. First, the pixel can be seen to represent a singular grid point. In this case, the rectangular area enclosed by four neighbouring grid points is called a *facet* of the raster model. From another point of view, the pixel can be seen to represent a rectangular area itself in an integral manner. The value assigned to a pixel describes one thematic attribute of that pixel. Topological information is only contained implicitly: neighbourhood relations can be described by index differences. Topographic objects can only be represented by a set of neighbouring pixels having identical attributes. Thus, the manipulation of individual objects is very difficult. However, the structure of raster data is simple, and operations requiring information on surface coverage can be performed rather easily. This is also true for data acquisition which can, for instance, be performed by classification of satellite images. These benefits are contrasted by the enormous requirements for data storage (especially for continuous tone data and in particular for 3D raster data) and the high computational costs for tasks such as geometrical transformations.

Geometrical dimensionality: Restricting ourselves to representation schemes containing the third dimension (the height component), there are several possibilities how this can be done [Kraus, 2000]:

1. *2D+1D*: The objects are basically described by their planimetric co-ordinates. A Digital Terrain Model (DTM) provides height information in an additional thematic layer so that for each 2D object point, its height can be interpolated from the DTM.
2. *2.5D*: Still, the objects are basically described by their planimetric co-ordinates. However, for each 2D point, the height is additionally stored as an attribute. As only one height can be assigned to one planimetric position, caves, bridges, etc., cannot be modelled in that way.
3. *3D*: All information is contained in three dimensions, and all co-ordinates are treated equally. Using 3D modelling (*solid modelling*) techniques is essential for modelling man-made objects such as buildings. However, the computational costs of common algorithms such as intersections are rather high.
4. *4D*: Time is contained as a fourth dimension.

Note that using 2D+1D or 2.5D techniques, closed solid objects can not be described. These techniques just provide a surface description of the object which, however, is sufficient in many cases, e.g. for a description of the terrain.

Topological dimensionality: Objects in vector data format are represented by points and their topological relations. Topological relations are described in terms of *topological simplices* [Heitzinger, 1996, Halmer et al., 1996] which are assigned a dimension:

1. Simplices of dimension 0 are called *nodes* or *vertices*; they correspond to the points of the object.
2. Simplices of dimension 1 are called *edges*: each edge connects two vertices.
3. Simplices of dimension 2 are called *faces*. In their simplest form, the faces are triangles.
4. Simplices of dimension 3: volumetric primitives, tetrahedrons in their simplest form.

Depending on the topological dimension used for modelling, four types of 3D object representations can be distinguished [Kraus, 2000, Bill and Fritsch, 1991] (cf. figure 2.2):

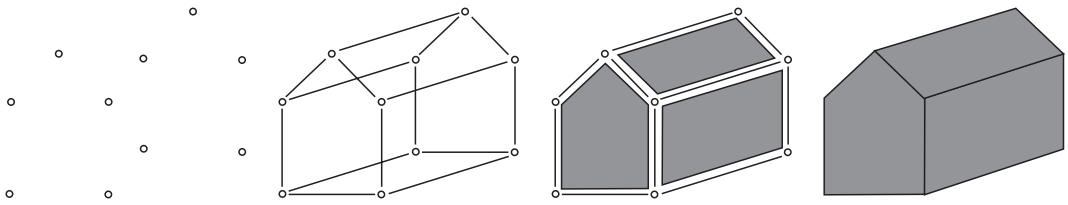


Figure 2.2: Object representation methods of different topological dimension (left to right): point cloud, wire frame model, surface model, volumetric model.

1. *Point cloud*: the object is just described by the vertices.
2. *Wire frame model*: the object is described by vertices and edges.
3. *Surface model*: the object is described by vertices, edges and faces. In a more general sense than the one described above, the faces can, for instance, be represented by closed planar polygons.
4. *Volumetric model*: the object is described by vertices, edges, faces and volumes, for example as a set of volumetric primitives.

Depending on the object class, different modelling techniques will be appropriate, each technique having its specific advantages and disadvantages. Several criteria described by the following questions can be used to analyze the properties of a certain modelling technique [Mäntylä, 1988, Bill and Fritsch, 1991, Englert, 1998]:

- *Domain*: Which objects can be described by a representation method?
- *The set of possible object descriptions*: Which semantically and syntactically correct representations can be built using that method?
- *Completeness*: Which representations describe at least one object or, in other words, are there correct representations which do not correspond to a real object?
- *Consistency*: Does a representation correspond to exactly one object or not?
- *Uniqueness*: Does each object have exactly one representation or not?
- *Efficiency*: How complex are algorithms based on that representation technique? What are the computational efforts both in time and memory space?

In the following sections, we will focus our attention to those classes of objects which are closer connected to our field of interest. Modelling techniques for the representation of the terrain in the sense of the "bald earth" will be discussed in section 2.2. After that, we will concentrate on the representation of man-made solid objects such as buildings in section 2.3.

2.2 Digital terrain models (DTMs)

Modelling the earth's surface is a central issue in topographic mapping. Digital Terrain Models (DTMs) are a valuable data source for many applications. That is why systems for the generation and visualization of DTMs were developed early [Kraus, 2000]. In this context, a clear distinction has to be made between DTMs, digital surface models (DSMs) and digital elevation models (DEMs):
DTMs describe the earth surface in the sense of the "bald earth" without human artefacts such as buildings or bridges and without vegetation. **DSM**s describe the surface in the sense of "the first point of intersection by a projecting ray", i.e., DSMs include points on buildings and vegetation as well as terrain points (in a more general sense, a DSM can represent the surface of any object in the way described above). Thus, whereas the term "DTM" also describes a semantic property of the object, "DSM" does not. The term **DEM** describes a 2.5D grid-based model that contains the elevations of points with respect to a reference surface, without any restriction on what the object is like. This term, thus, characterizes a modelling technique rather than the data which are described by an elevation model. In topographic mapping, both DSMs and DTMs can be described by DEMs (but they can also be described by other techniques, as we shall see in this section).

Most systems for DTM generation model the terrain surface in 2.5D. This means that only surfaces which can be represented as $Z = Z(X, Y)$ can be modelled by these systems. As the earth surface cannot be described by one closed analytic function in a scale appropriate for topographic mapping, it has to be divided into facets which are small enough that within them the surface can be approximated by an analytic function. Terrain modelling techniques can be classified according to several criteria:

1. Data structure: Surfaces can be described as raster models or by more sophisticated techniques using different types of facets.
2. Mathematical Model: The terrain surface has to be modelled within the facets. Thus, a functional model for the interpolation of surface points within the facets has to be provided.

3. Generation: DTMs are generated from distinct points having been measured in analytic plotters or automatically by image matching techniques. DTM generation methods can be classified according to whether they allow for filtering or not. Another classification scheme distinguishes local generation techniques from global ones.

In order to obtain a high-quality representation of the earth surface, several requirements have to be fulfilled [Kraus, 2000]:

- *Smoothing*: Random errors are contained in the original data. In order to reduce their influence on the resulting terrain model, filtering methods are required in the generation process.
- *Geomorphological correctness*: Geomorphological information such as information on break lines or mountain peaks should be considered.
- *Variable point density*: In order to reduce the amount of disk and memory space required for DTMs, methods allowing for variable point density (variable facet size) are required: in regions of high terrain curvature, a dense distribution of data points is required whereas flat and smooth regions can be modelled by a small number of large facets.
- *Robustness*: The generation process should be robust with respect to the spatial distribution of the original data points.
- *Applicability*: In the sense of the classification scheme for modelling techniques given in section 2.1, 2.5D representations are very restricted: overhanging regions and caves cannot be modelled by them.

There are two common groups of approaches for terrain modelling techniques which will be described in the following sections: *Grid based DTMs* will be discussed in section 2.2.1, and *DTMs based on triangulation* will be discussed in section 2.2.2.

2.2.1 Grid-based terrain models

Figure 2.3 shows the geometry of a grid-based DTM representation. The terrain is represented by a 2D matrix $Z_{i,j}$ of $N_X \times N_Y$ elements. The transformation between the matrix indices $(i,j), \{i,j\} \in \mathcal{N}_0$, and the planimetric co-ordinates (X,Y) is described by a shift and two scales: $X = X_0 + \Delta X \cdot i$ and $Y = Y_0 + \Delta Y \cdot j$. Inside the grid meshes, the terrain surface is described by a bi-linear polynomial function in X and Y. Thus, the height Z of a point $\mathbf{P}$ having the planimetric co-ordinates (X,Y) can be computed from [Kraus, 2000]:

$$Z = a_0 + a_1 \cdot s + a_2 \cdot t + a_3 \cdot s \cdot t \tag{2.1}$$

In equation 2.1, (i,j) are the minimum indices of the corner points of the grid mesh containing (X,Y), and (s,t) are the planimetric co-ordinates of $\mathbf{P}$ in grid units in a local co-ordinate system centered at grid point (i,j):

$$\begin{aligned} s &= \frac{X - X_0}{\Delta X} - i \\ t &= \frac{Y - Y_0}{\Delta Y} - j \end{aligned} \tag{2.2}$$

The coefficients a_k of the bi-linear function are computed from the corner points of the grid mesh:

$$\begin{aligned} a_0 &= Z_{i,j} & a_1 &= Z_{i+1,j} - Z_{i,j} \\ a_2 &= Z_{i,j+1} - Z_{i,j} & a_3 &= Z_{i+1,j+1} + Z_{i,j} - Z_{i+1,j} - Z_{i,j+1} \end{aligned} \tag{2.3}$$

The heights of the DTM raster points have to be computed from the arbitrarily distributed object points. The standard methods for that purpose are:

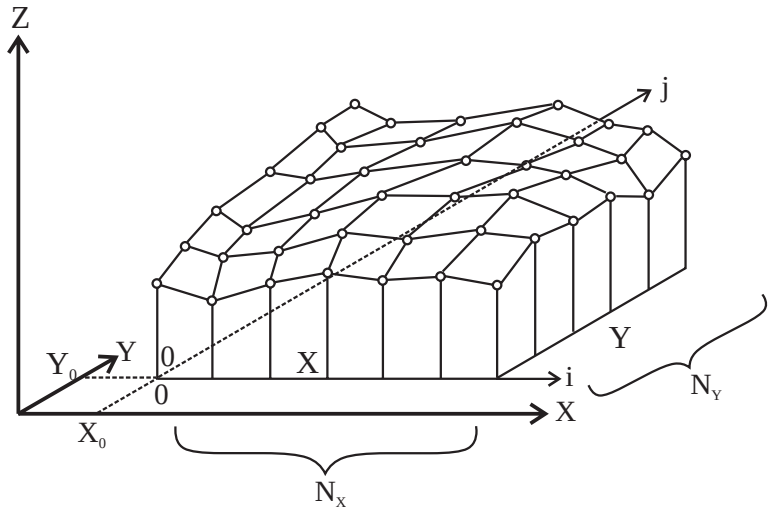


Figure 2.3: Geometry of grid-based DTMs [Kraus, 2000].

- *Surface approximation by finite elements*: This method is based on equations 2.1 to 2.3. For each measured point $\mathbf{P}$, one equation 2.1 can be formulated linking the (measured) height Z_P via the coefficients a_k with the (unknown) grid heights $Z(i,j)$ which are computed from least squares adjustment. Least squares adjustment has the requested smoothing effects. In order to obtain a smooth surface, both the curvatures and the torsions in the grid points can be assumed to be zero, which leads to additional observation equations being introduced to adjustment. This is especially necessary for regularization in grid meshes without observed data points [Ebner and Reiß, 1978, Wild and Krzystek, 1996].

- *Linear Prediction*: The grid heights can be computed from linear prediction from the heights of the data points $\mathbf{P}$. Assuming the terrain to be smooth, the heights are correlated, the degree of correlation (of smoothness) being described by a co-variance function shaped like a Gaussian bell curve. By certain assumptions on that co-variance function, the smoothing effects of the method are controlled [Wild, 1983].

For high-quality DTMs, the effect of filtering is, sometimes, too rigid. Considering for instance rough terrain, there are regions with abrupt changes of terrain smoothness. In order to model such terrain edges, a hybrid DTM data structure containing *break lines* is required: The break lines are introduced in the generation process in both models: along these lines, no smoothness assumptions are introduced to adjustment, and the intersection points of the break lines with the grid lines have to be determined as additional unknowns. In the finite element method, curvature equations are introduced along the break lines whereas across the break lines, no such equations are used. In linear prediction, the co-variance of heights from different sides of the break line are assumed to be 0. Note that the functional model from equation 2.1 only holds for grid meshes not being crossed by break lines. In grid meshes being crossed by break lines, the grid points and the break line points have to be triangulated so that the break line points are forced to be connected by triangulation edges (cf. section 2.2.2), and the surface is represented by the 3D triangles in these regions [Kraus, 2000]. The break lines have to be added to the DTM data structure which will then no longer be a pure raster representation [Köstli and Sigle, 1986, Kraus, 2000]. Figure 2.4 shows a high-quality DTM from large scale topographic mapping: the edges of road embankments are introduced as break lines. The DTM has been created by linear prediction using the program system *SCOP* [SCOP, 1994].

Note that with the advent of 3D laser scanners, robust estimation techniques have gained even more importance because they deliver a digital surface model (DSM) rather than a DTM. For these kind of data, it is important to separate the points representing the terrain from the points on buildings and vegetation. This can be achieved by a robust estimation technique based on a skew error distribution function applied to the original data in several resolutions, thus filtering away the points not belonging to the terrain

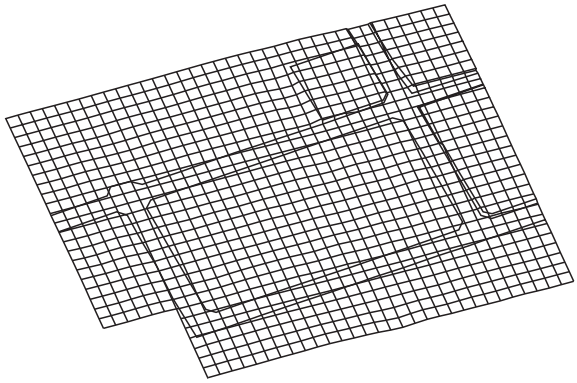


Figure 2.4: A high-quality 2.5D DTM of a topographical surface containing break lines.

[Pfeifer et al., 1999a, Kraus and Pfeifer, 1998, Rieger et al., 1999]. A coarse-to-fine strategy is especially necessary with high-resolution laser scanner data in densely built-up areas because in this case, the buildings cause the same error characteristics as small table mountains with steep edges (figure 2.5; cf. also section 1.2.1). Figure 2.5 also shows that buildings can (within certain limits) be contained in grid-based models and that nice visualizations can be created in this way. However, the buildings are not really addressable as topographic objects by this representation technique, which is the reason why a subset of a grid-based DSM can hardly be called a building model. The fact that the terrain points can to a certain extent be separated from the other points is one of the reasons why laser scanner data are used for automatic building detection [Weidner, 1997, Fritsch and Ameri, 1998, Haala et al., 1998, Rieger et al., 1999].

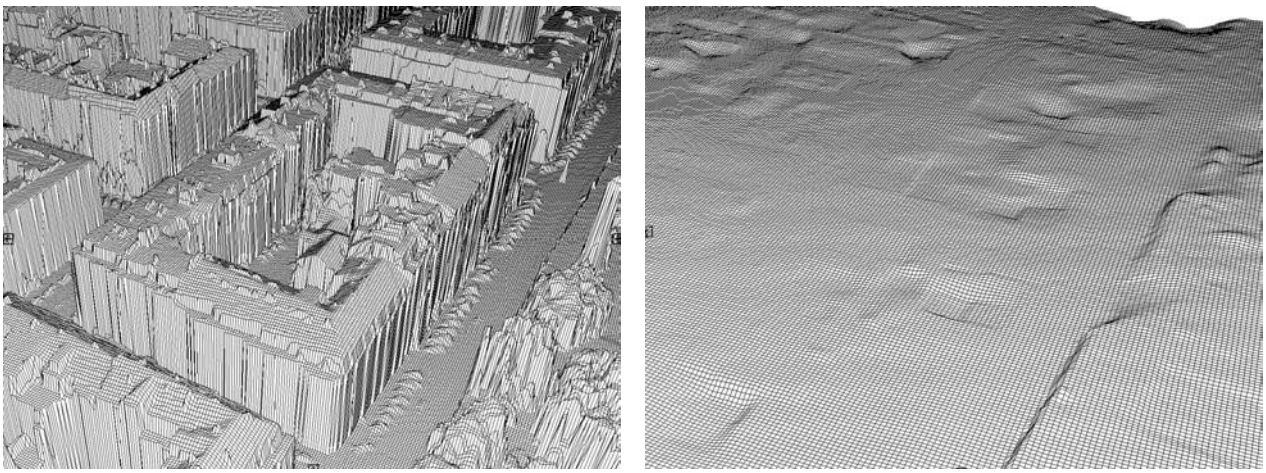


Figure 2.5: DSM and DTM from high-resolution laser scanner data of a test site in the city of Vienna. Left: A DSM going through the original data points. Right: The filtered DTM. Buildings, vegetation and other objects have been eliminated.

2.2.2 Terrain models based on triangulation

By this group of methods, the terrain is represented by a set of vertices v, a set of edges e, and a set of triangular faces f. The 3D co-ordinates of the original data points are assigned to the vertices. Each edge e connects two vertices, and it is the intersection of exactly two faces. Each triangular face f, on the other hand, is bordered by exactly three edges. The edges e and the faces f describe the neighbourhood relations of the original data points. This triangulation network is called *triangulated irregular network* (TIN). The terrain is approximated by the polyhedron consisting of the triangles f. Inside the triangles, the surface is assumed to be planar [Kraus, 2000].

In order to create a TIN, the original data points have to be connected by edges so that triangles are formed. This

is usually done incrementally by inserting one point after the other. As usually only the planimetric co-ordinates of the vertices are used for that purpose, most TIN approaches have more or less 2.5D characteristics. There is no unique way to generate triangles from arbitrarily distributed points, so that a solution has to be found which is optimal in a certain sense. The most common optimization criterion for 2D triangulation is the *Delaunay criterion*: the points are to be connected by edges to form triangles so that for each triangle no fourth point of the triangulation is within the circumcircle [Preparata and Shamos, 1990, Kropatsch et al., 2000b]. This criterion is equivalent to the maximization of minimum angles of the triangles [Heitzinger, 1996]. The Delaunay triangulation criterion yields triangles which are closest to equilateral ones.

In order to make the representation independent from the orientation of the co-ordinate system, the Delaunay criterion can be expanded to 3D triangulation by considering the surface normals of the vertices v, too. These surface normals are either one of the results of data acquisition, or they have to be determined from the data in a pre-processing step, approximating the surface locally by its tangential planes. A 3D triangulation is then a triangulation of 3D points fulfilling two criteria [Heitzinger, 1996]:

- The minimum angles of the triangles are maximized (Delaunay criterion)
- The angle between two neighbouring triangle surfaces is maximized, i.e. the angles between the surface normals are minimized. This criterion will result in a smooth approximation of the surface.

In order to insert a point into the triangulation, the triangle that the point belongs to has to be found. For that purpose, the surface normals are used: in 2D, a point is inside a triangle if it is on the same side of all edges if the edges are oriented in the same way. In 3D space it cannot be decided whether a point is on the left or right hand side of an oriented edge because a straight line does not divide 3D space into two half-spaces. However, a plane defined by the edge and the normal vector of the triangle does, and this plane can be used for the decision on which side of an edge a point is situated. Thus, a point is considered to be "inside" a triangle if it is inside a triangular prism defined by the triangle and the normal vector of that triangle. One can see the problems connected with that definition: closed surfaces will be intersected by that prism twice. Thus, the decision is not unique, and the results of 3D triangulation are no longer independent of the order of the data points [Heitzinger, 1996].

In some cases, the approximation of the terrain by planar surfaces is not precise enough. This problem can be overcome by modelling the surface inside the triangles by Bézier triangles with certain smoothness assumptions at the borders [Pfeifer and Pottmann, 1996].

As with grid-based DTMs, break lines are necessary to be considered in order to obtain a high-quality representation of the surface. At break lines, there is a discontinuity of the first derivatives, thus, for points at a break line, there exist two surface normals; the smoothness criterion described above is not to be used. In addition, the break lines have to coincide with edges in the triangulation. This can be achieved by making them constraints in the triangulation: two vertices connected by a break line in object space have to be connected by an edge in the triangulation even if the shape of the triangles will then no longer be optimal in the sense of the Delaunay criterion [Halmer et al., 1996]. Figure 2.6 shows a high-quality DTM based on 3D triangulation considering constraints.

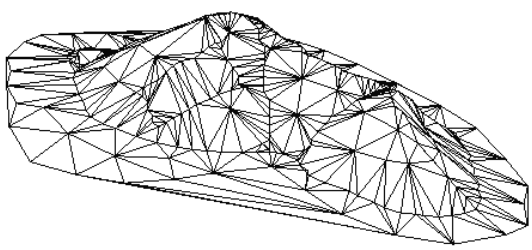


Figure 2.6: A perspective view of a 3D triangulation with constraints at the ridge.

2.2.3 Comparison of grid based and triangulation based DTMs

In table 2.1, grid based and triangulation based terrain modelling techniques are compared with respect to the criteria defined in the first part of section 2.2. Note that even though triangulation is based on 3D algorithms, there are restrictions in the applicability due to the problems with the definition of the ordering criterion described in section 2.2.2. With respect to modelling of buildings, both methods can be applied with certain drawbacks. Modelling buildings (without overhangs!) by grid based models is possible if two break lines of different heights are introduced at the building outlines (one corresponding to the building outlines on the floor and the other one to the roofs; the latter one has to be displaced slightly from the first one so that it is "inside" the outline on the floor: the walls have to be slightly tilted so that no overhangs occur) and if break lines are used to model internal ridges. However, using grid based modelling, a building cannot be addressed as an individual object in a data base. Modelling buildings by triangulation techniques is a special case of boundary representation (cf. section 2.3.1). It can be accomplished, but not automatically, just based on a point cloud as input.

	Grid DTM	Triangulation
Smoothing	Due to the generation algorithm based on least squares adjustment, grid based methods perform smoothing.	Difficult to achieve because the original data points are used.
Geomorphology	Break lines can be considered.	Break lines can be considered.
Point density	Fixed due to the matrix structure. Usually, a trade-off between storage capacities and precision is sought for.	Variable as the original data points are used.
Robustness	Robust estimation procedures can be applied. Problems appear with inhomogeneous point distributions.	Problems due to the non-uniqueness of the ordering criterion.
Applicability	Restrictions due to the 2.5D characteristics. Simple algorithms can be found for many tasks.	More general than grid based methods, but also restricted. More difficult algorithms are required.

Table 2.1: A comparison of grid based and triangulation based DTMs.

2.3 Modelling of man-made solid objects

Solid modelling is the branch of geometric modelling concerned with the representation of 3D solid, especially man-made, objects. On the contrary to the modelling techniques described in section 2.2, solid modelling divide 3D space into a part that is "inside" the object and a part that is "outside". Some paradigms regarding the topological dimensionality of 3D vector based descriptions have already been described in section 2.1. Of the four classes of representations, only surface and volumetric models can be used for solid modelling techniques. Wire frame models (cf. figure 2.2) do not contain the faces of the object and thus are not unique [Samet, 1989, Mäntylä, 1988]. However, they are often used for visualization because they can be interpreted easily. According to [Samet, 1989], there are six common representations in solid modelling:

1. *Spatial enumeration*: This is the simplest form of a 3D volumetric raster model. A section of 3D space is described by a 3D matrix of evenly spaced cubic volume elements (*voxels*). All voxels being "inside" the solid object are flagged. Voxel models contain highly redundant information, and thus require much disk/memory space. In addition, topological information is not contained in such models, and operations such as geometric transformations are rather costly. The spatial resolution is restricted by the grid size,

i.e. the voxel dimensions. However, Boolean operators such as the union of two objects, can be performed quite easily [Samet, 1989, Streilein, 1999].

2. *Cell decomposition*: a hierarchical adaptation of spatial enumeration. 3D space is sub-divided into cells which, however, are of different size. A regular variant of this representation based on recursive decomposition of 3D space by cubes is the *region octree*. More general decomposition methods make use of cells of different shapes, positions, and sizes. These simple cells are glued together to describe the solid object.

3. *Boundary model* or *Boundary representation (B-rep)*: The object is represented by its boundary which consists of a set of faces, a set of edges and a set of vertices as well as their mutual topological relations.

4. *Sweep methods*: A planar shape is moved along a curve. According to the type of movement, translational and rotational sweep techniques are possible. These methods are well-suited for representing prismatic and rotationally symmetric solid objects.

5. *Primitive instancing*: These modelling schemes provide a set of all possible object shapes which are described by a set of parameters. Instances of any object shape can be created by varying these parameters.

6. *Constructive solid geometry (CSG)*: In CSG, primitive instances are combined to form more complex objects by using geometric transformations and Boolean set operations.

In the next sections we will describe in more detail the last four of the above modelling techniques because they are relevant for modelling buildings. After that, the applicability of these representation schemes for building extraction and representation will be compared in section 2.3.5.

2.3.1 Boundary models

Boundary models offer a very flexible tool for modelling man-made objects. They are based on a surface-oriented view of solid objects: an object is considered to be represented completely by its bounding faces. In order to represent the object correctly, boundary models additionally consist of edges and vertices as well as the topological relations of all features. The faces, the edges, and the vertices are the (labelled) nodes of a graph, and the direct neighbourhood relations are described by the edges of the graph (figure 2.7). B-rep models consist of geometrical data and topological data. The geometrical data are assigned to the nodes of the graph in figure 2.7:

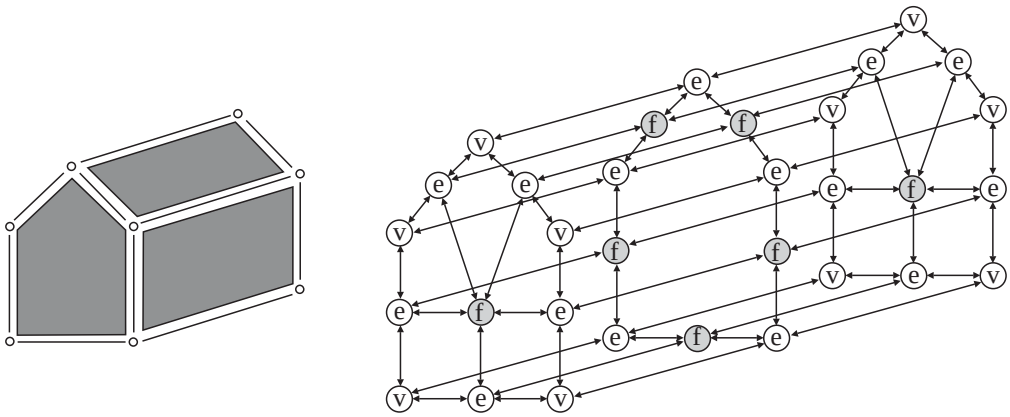


Figure 2.7: Boundary model of a solid object: a graph with nodes of type f (faces), e (edges) and v (vertices) and their topological relations.

- The faces f are described by surface equations. As we want to restrict ourselves to polyhedral objects, faces can be assumed to be plane surfaces described by equation 2.4:

$$a \cdot (X - X_0) + b \cdot (Y - Y_0) + c \cdot (Z - Z_0) + d = 0 \tag{2.4}$$

  In equation 2.4, (a, b, c) are the components of the normal vector of the surface and $\mathbf{P}_0 = (X_0, Y_0, Z_0)^T$ is some reduction point not necessarily situated on the surface.

- Edges are given by the intersection of two faces. If the faces are planar, the edges are straight lines described by two equations 2.4.

- Vertices are geometrically described by the point co-ordinates (X, Y, Z) which, however, depend on the equations 2.4 of the neighbouring faces.

The topological relations are provided by the edges in the graph in figure 2.7 [Mäntylä, 1988]:

- The solid consists of a set of faces.

- Each face is bordered by a set of edges. These bordering edges have to be ordered so that they form a closed curve (a *loop*). In order to separate the "inside" of a solid object from its "outside", the edge neighbour list of each face has to be ordered, e.g. mathematically positive (figure 2.8).

- Edges have neighbouring faces intersecting at the edge.

- Edges are limited by neighbouring vertices.

- Vertices have a set of neighbouring edges which intersect at them.

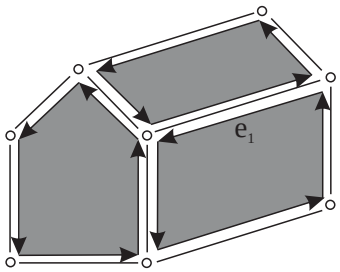


Figure 2.8: Edges being neighbours of faces have to be ordered to form loops in order to distinguish the "inside" of the solid from its "outside". Each edge (e.g. e_1) is contained in two loops with opposite orientations.

The above definitions do not yet guarantee a boundary model to describe a valid solid object. For that purpose, the solid has to be a closed orientable 2-manifold [Mäntylä, 1988]. This means that the boundary of a solid is completely covered by faces so that three conditions are fulfilled:

1. Each edge connects exactly two vertices.

2. Each edge has exactly two neighbouring faces (loops). Within the two loops, the edge is contained in opposite orientations (e.g. edge e_1 in figure 2.8).

3. Every vertex is surrounded by a single cycle of edges and faces.

Up to now, we have only considered "simple" solids without holes. In order to describe more complex objects such as the one depicted in figure 2.9, the boundary of faces has to be de-composed into two or more closed loops. The *outer loop* (l_1 in figure 2.9) is oriented mathematically positive. The other loops (e.g. l_2 in figure 2.9) are called *inner loops*; they are oriented mathematically negative. Finally, solids may consist of one or more

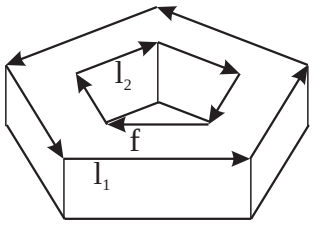


Figure 2.9: A solid object with a hole: the boundary of the upper and lower surfaces are de-composed into two closed loops. Adapted from [Samet, 1989].

shells. A shell is a maximal connected set of faces of a solid (e.g. a cube inside a cube). For closed orientable 2-manifold polyhedral solids, the Euler-Poincaré formula relates the number of vertices n_v, the number of edges n_e, the number of faces n_f, the number of loops n_l, the number of holes n_h, and the number of shells n_s [Streilein, 1999]:

$$n_v - n_e + (n_f - n_l) = 2 \cdot (n_s - n_h) \tag{2.5}$$

In order to manipulate solids fulfilling equation 2.5, *Euler operators* can be used. Euler operators change the numbers of topological elements in equation 2.5 without changing its validity. It can be shown that using these operators, only realizable models can be built. There are basic Euler operators which are pairwise inverse [Mäntylä, 1988]:

1. *Vertex splitting*: A vertex is split into two vertices connected by a new edge. Care has to be taken that the the neighbouring loops are modified consistently, too.
2. *Vertex joining*: Two neighbouring vertices are merged into one and the edge between them is deleted. This operator is the inverse of vertex splitting.
3. *Face splitting*: A face is split into two faces by inserting a new edge between two vertices of its outer loop.
4. *Face joining*: Two neighbouring faces are merged by deleting an edge between them. This operator is the inverse of face splitting.
5. *Loop splitting*: A loop is split into two loops, one of them being a new inner boundary of the loop's face, by deleting an edge. This operator will thus create a new inner loop.
6. *Loop joining*: Two loops, at least one of them being an inner loop, are joined to form one loop. Thus, an inner loop is deleted. This operator is the inverse of loop splitting.

Using these operators, all topologically consistent boundary models can be built. However, the geometrical integrity of a boundary model cannot be enforced that easily. Invalid models can be built by assigning inappropriate geometric information to consistent topological entities [Mäntylä, 1988].

2.3.1.1 Internal representation of boundary models

The internal representation of boundary models can be based on faces, on vertices, or on edges. The most common representation technique is the *winged-edge* [Baumgart, 1975] or, slightly modified, the *full winged-edge* data structure [Mäntylä, 1988], both representing the neighbourhood relationships in the structure describing the edges. In this representation, a solid consists of a list of faces, a list of edges and a list of vertices. Each face consists of a list of loops, one of which being marked as outer loop of the face. Each loop contains a reference to the face it belongs to and to the first edge of the loop. Each vertex contains its co-ordinates and a reference to its first neighbouring edge. The rest of the neighbourhood relations is contained in the edges, enforcing by structural means two of the consistency rules for 2-manifolds (figure 2.10):

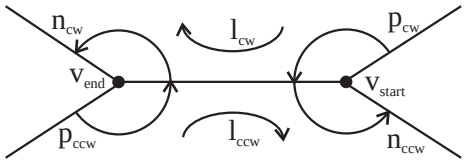


Figure 2.10: An edge in the full winged-edge data structure. Adapted from [Mäntylä, 1988].

- Each edge contains references to its two neighbouring vertices v_{start} and v_{end}.
- Each edge contains references to its two neighbouring loops l_{cw} and l_{ccw}. As the edge is oriented in opposite directions in its neighbouring loops, we can distinguish the loop where it is contained in its positive ("clockwise") direction l_{cw} from loop l_{ccw}, where it is contained in its negative ("counter-clockwise") orientation.
- Each edge contains references to the next edges in both loops: n_{cw} is the next edge in l_{cw}, and n_{ccw} is the next edge in l_{ccw}. Thus, each loop can be traversed using the reference to the first edge in the loop and then following n_{ccw}. The edges intersecting at a vertex can be traversed using the reference to the first edge of the vertex and then following the references n_{ccw}, n_{cw}, p_{ccw}, or p_{cw}, depending on whether the vertex is v_{end} or v_{start} of the edge and changing from l_{ccw} to l_{cw} and vice versa at each step.
- Each edge contains references to the previous edges in both loops: p_{cw} is the next edge in l_{cw}, and p_{ccw} is the next edge in l_{ccw}.

An adaptation of the full winged-edge is the *half-edge* data structure [Mäntylä, 1988]: a half-edge describes one segment of a loop. It consists of a reference l to its parent loop, a reference v to its starting vertex in direction of the loop, a reference edg to its edge and two references to the next and the previous half-edge in the loop. The edge data structure is modified so that it just combines the two half-edges he_1 and he_2. The half-edge data structure expresses even more clearly the property of 2-manifolds that an edge occurs exactly in two loops, in each having a different direction (figure 2.11).

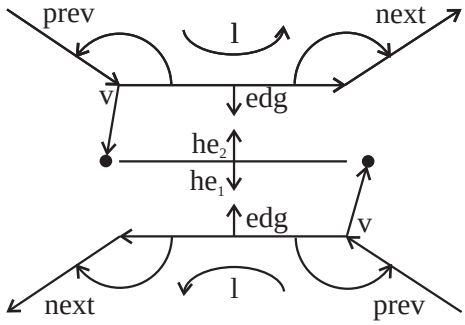


Figure 2.11: An edge with its two half-edges in the half-edge data structure. Adapted from [Mäntylä, 1988].

2.3.1.2 Properties of boundary models

Boundary models are well-suited for visualization tasks because they readily include all data required for that purpose. This is one of the reasons why they are used very often for 3D solid modelling systems. They offer a very flexible tool for modelling man-made objects even though it is hard for a human operator to create a valid boundary model from scratch without further guidance by a user interface. Boundary models are unambiguous, i.e. there is exactly one object corresponding to a given boundary model, and they are unique if neighbouring coplanar faces are merged so that the faces are forced to have maximum extent. There are also some drawbacks of

boundary models: whereas topological integrity can be enforced relatively easily, geometric correctness cannot or only at a very high computational cost. In addition, boundary models are not closed under set operations, e.g. the union of two boundary models does not necessarily result in a new valid boundary model (e.g. if a vertex or an edge of one solid touches any element of the other), even though this drawback can be overcome by inserting vertices and/or edges twice into the resulting model. It is especially these Boolean set operations which become very difficult using boundary models [Mäntylä, 1988]. As the combination of solids is very important for many modelling systems (cf. section 2.3.4), we will now have a closer look at it.

2.3.1.3 Combinations of boundary models

The simplest possibility for combining boundary models is glueing two solids over one common co-incident face (figure 2.12). According to [Mäntylä, 1988], this problem can be solved in three steps:

1. Merge the two solids into one solid having two shells.
2. Join the two shells. This can be accomplished by merging the glued faces into one by making the outer loop of one of them an inner loop of the other one and deleting the first face.
3. Eliminate co-incident vertices and edges and join co-planar faces.

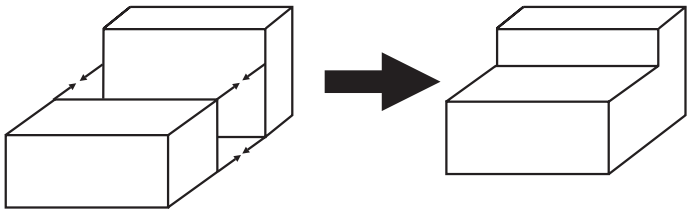


Figure 2.12: Glueing two solids over a common face.

Note that in applications such as building extraction, the geometric condition that the glued faces are co-incident will hardly ever occur. The same statement holds true for vertices: two vertices will hardly ever really be co-incident with an arbitrary precision. That is why any comparisons of geometrical entities, especially of the vertex co-ordinates, have to be made using a user-defined threshold ε: Two vertices are considered to be co-incident if their Euclidean distance $d \leq \varepsilon$ [Englert, 1998].

Glueing two solids is not a very general operation because it only works if certain geometrical conditions are fulfilled, which is not generally the case with real-world objects. More general operations are performed using *Boolean set operations* [Mäntylä, 1988]: union ($\cup$), intersection ($\cap$), and difference ($\setminus$). Figure 2.13 shows the effect of these operators applied to two (2D) point sets A and B. Figure 2.14 shows the results of the Boolean union of two box-shaped solid objects. In order to perform these set operations, all possible geometrical intersections between vertices, edges, and faces have to be computed, and various tests for overlap, co-planarity, and intersection have to be performed, again taking into account numerical errors by an error threshold ε.

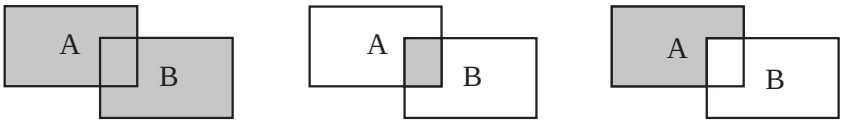


Figure 2.13: The three Boolean set operations, from left to right: union ($A \cup B$), intersection ($A \cap B$), difference ($A \setminus B$), for reasons of simplicity in 2D. The result is shown in grey colour. Taken from [Englert, 1998].

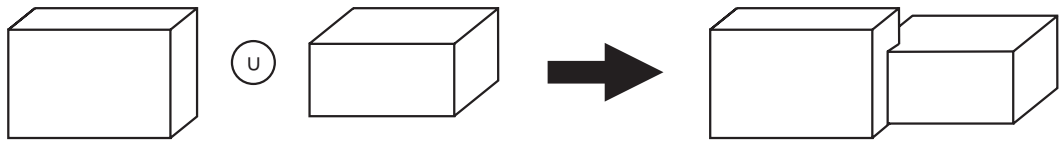


Figure 2.14: An example for the union of two solids.

[Mäntylä, 1988] suggests to break the complicated analysis into several steps of a complexity which can be handled. It is the idea of his algorithm to first split both solids into a part being "outside", one being "inside" and a third one being co-incident with ("on") the other solid. Thus, if the two solids are denoted by A and B, the boundaries of these solids are classified into eight classes ($AinB, AoutB, AonB^+, AonB^-, BinA, BoutA, BonA^+, BonA^-$), where $AinB$ means "the part of the boundary of A that is inside B", $AoutB$ means "the part of the boundary of A that is outside B", $AonB^+$ means "the part of the boundary of A that is co-incident with the boundary of B where the face normals are equal", $AonB^-$ means "the part of the boundary of A that is co-incident with the boundary of B where the face normals are opposite", and the other components are defined analogously. After that, these parts are merged to generate the resulting solid according to which operator is to be applied, following equation 2.6 [Mäntylä, 1988]:

$$\begin{aligned} A \cup B &= AoutB \oplus BoutA \oplus AonB^+ \\ A \cap B &= AinB \oplus BinA \oplus AonB^+ \\ A \setminus B &= AoutB \oplus (BinA)^{-1} \oplus AonB^- \end{aligned} \tag{2.6}$$

where $\oplus$ denotes the glueing operation and $(BinA)^{-1}$ denotes $BinA$ with the orientations of all faces reversed. The algorithm consists of four steps:

1. Compute all possible intersections between edges from the two solids as well as those between edges and faces from the two solids, and search for vertices being co-incident with an edge or a face. Edges should be split at a vertex which is found to be co-incident with the edge. Note that new intersecting vertices are added to both solids.

2. Vertex neighbourhood classification: The cycle of edges and faces around each intersection vertex has to be classified as being "inside", "outside" or "on" the other solid, and it has to be split into two parts "inside" and "outside", in order to replace the eight-way classificiation required for equation 2.6 by a four-way classification by assigning the co-incident parts of both solids to the other parts, this assignment being dependent on the Boolean operator currently applied. After that, both input solids are split by inserting edges of zero length at the intersection vertices. Vertex neighbourhood classification is very complicated because various cases of intersections or co-incidence of face sectors have to be handled.

3. According to the results of classification, new edges are inserted to split the faces of both solids along the intersection. After this step, both input solids are split into three parts as described above.

4. Finally, the resulting solid has to be generated according to the operator applied. First, new faces have to be generated at the intersections for all parts of the two solids. After that, all faces belonging to the resulting solid have to be merged according to equation 2.6, and the vertex and edge lists of the resulting solid have to be created. Finally, the individual parts coming from the two input solids have to be glued together at the intersection faces.

A similar algorithm applying a different technique of vertex classification is given by [Englert, 1998].

2.3.2 Sweep methods

Sweep-representations of a 3D object are created by moving a planar (2D) shape (e.g. a closed polygon) according to a pre-defined rule [Samet, 1989, Streilein, 1999]. Depending on the rule by which the 2D shape is moved, two types of sweep representations can be distinguished (figure 2.15):

1. *Translational sweep*: The shape is translated along a pre-defined translational vector.
2. *Rotational sweep*: The shape is rotated around a pre-defined rotational axis.

Taking into consideration the way these representations are generated, it can be concluded that they are well suited for prismatic and rotationally symmetric objects. The concept of translational sweeps can be enlarged to sweeping two shapes along each other [Mäntylä, 1988]. Sweep representations are widely-used in computer vision. However, the generation of arbitrary objects becomes rather difficult using this technique [Streilein, 1999].

Figure 2.15: Sweeping a planar rectangular shape. Left: a translational sweep creates a vertical prism. Right: a rotational sweep creates a cylindrical object.

2.3.3 Primitive instancing

It is the idea of *primitive instancing* to provide a set of pre-defined object types. Each object type is represented by a small set of parameters, and individual instances are created by selecting a model type, providing the new instance with a parameter tuple and applying a rigid motion (a translation and a rotation in 3D space) to the primitive. Figure 2.16 shows a simple building primitive described by its length l, its width w, the gutter height h_1, and the roof height h_2. Conditions can be imposed to the parameters in order to achieve validity of all models which can be created by variation of the parameters. For instance, none of the parameters of the primitive in figure 2.16 may become 0 or negative. Of course, by using this modelling technique, one is restricted by the pre-defined "library" (the *primitive data base*) of object types: only objects corresponding to one of the pre-defined types can be modelled because no operations for the combinations of instances are possible. On the other hand, providing a large number of object types in the primitive data base results in an enormous programming effort as tailored code is required for each object type. However, for the object types contained in the primitive data base, modelling becomes very easy and efficient. That is why primitive instancing is supported by many CAD and modelling systems as an auxiliary technique for the representation of often-needed parts, even though the modelling system itself is based on other representations (cf. section 2.3.5) [Mäntylä, 1988, Streilein, 1999].

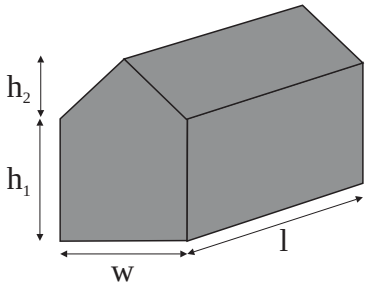


Figure 2.16: A simple primitive as it could be used for modelling buildings. The primitive shape is described by the parameter vector $\mathbf{P} = (l, w, h_1, h_2)^T$.

2.3.4 Constructive Solid Geometry (CSG)

It is the concept of CSG to provide solid 3D primitives which are described a set of parameters reflecting the object dimensions (Figure 2.17). The CSG primitives are simple objects such as cubes, boxes, tetrahedrons or quadratic pyramids. They are considered to be bounded point sets in 3D space, and they can easily be combined using Boolean set operations (*union* $\cup$, *intersection* $\cap$ and *difference* $\setminus$) in order to represent more complex objects consisting of more than one primitive (cf. section 2.3.1.3). In theory, the (bounded) primitives themselves can be considered to be the intersections of half spaces containing all points $\mathbf{P}$ for which the inequality $f(\mathbf{P}) \leq 0$ is fulfilled, where $f(\mathbf{P})$ is a characteristic function of the point $\mathbf{P}$ and $f(\mathbf{P}) = 0$ describes the bounding surface of the point set, e.g. a plane in 3D space (*halfspace models*) [Mäntylä, 1988].

The most natural way to represent a CSG model is the *CSG tree* which can be defined as follows:

```
<CSG tree>  ::=  <primitive> |
                 <CSG tree> <set operation> <CSG tree> |
                 <CSG tree> <rigid motion>
```

where `<primitive>` is an instance of one of the primitives of the primitive data base, `<rigid motion>` is either a translation or a rotation, and `<set operation>` is either $\cup$, $\cap$, or $\setminus$. The leaves of the CSG tree are the primitives, and the nodes are marked either as a Boolean set operation or with a rigid motion. Thus, in the CSG tree, the history of generation of the solid is stored. The solid itself corresponds to the upper node of the CSG tree (figure 2.17) [Mäntylä, 1988].

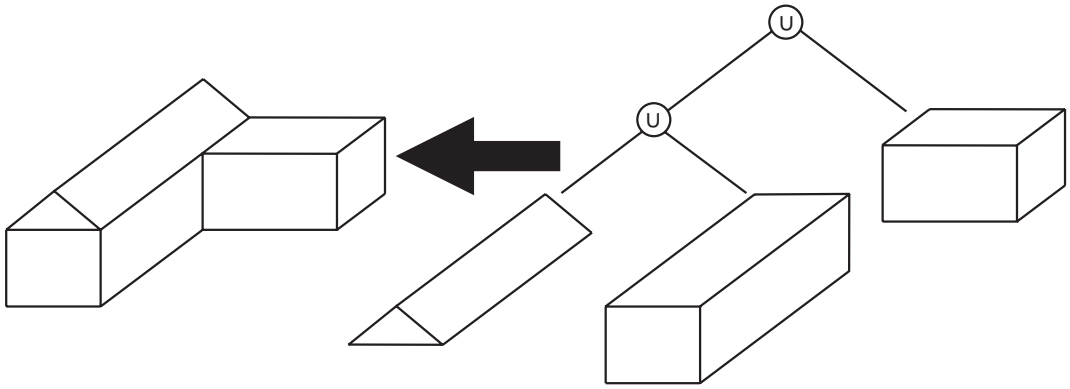


Figure 2.17: CSG: The CSG model (left) is is represented by the CSG tree (right) consisting of three primitives connected by a Boolean union operation ($\cup$).

CSG is a very powerful concept for object modelling in automation procedures for building extraction and for 3D city models, especially well suited for objects which are relatively simple and show symmetries, because many buildings can be represented by a combination of simple basic building primitives. CSG trees are always guaranteed to represent valid objects, they are unambiguous but not unique: each CSG tree models exactly one object, but the CSG tree of a given object is not unique because an object can be built in several ways. As Boolean set operations are an integral part of a CSG tree, these operations are closed for CSG trees, e.g., the union of two CSG trees will again be a valid CSG tree. CSG is not as flexible as boundary representation: the applicability of CSG depends on the primitive set which can be used [Müller, 1998]. If an inappropriate set of primitives is offered, object modelling using these primitives will become difficult.

Some tasks such as the classification of a point as being inside or outside the solid are very simple to solve using CSG. However, for certain applications, especially for visualization, a boundary representation is to be derived from a CSG model. This operation called *boundary evaluation* is rather complicated. First, the CSG primitives have to be converted to boundary models, and then these models have to be combined using the Boolean set operations, which, as already stated in section 2.3.1.3, turns out to be very complicated. Note that the conversion cannot be inverted: whereas in CSG trees, the history of generation of a solid model is stored,

this information is lost during the conversion to B-rep, and there is no general method available to reconstruct CSG trees from B-rep.

2.3.5 Solid modelling and building reconstruction

Having in mind the properties of the solid modelling schemes described in the previous sections, we now want to consider their applicability for building reconstruction. *Primitive instancing* can hardly be used for that purpose because the number of primitives required in the primitive data base would become too great and because the concept does not provide techniques for the combination of primitives. *Sweep methods* can be used as auxiliary tools to create simple shapes in the context of another modelling technique, e.g. they can be used for creating a cylinder as was shown in section 2.3.2 [Mäntylä, 1988]. Sweep methods can also be useful for the generation of a 3D city model based on an existing 2D TIS such as the cadastre. For instance, in the cadastre, all buildings are contained as closed 2D polygons. If some additional information on the building height and a DTM are available, the 2D polygons can first be projected to the DTM in order to obtain the floor heights, and the resulting polygons which are still planar but have been assigned the floor heights can be swept along the vertical co-ordinate axis by the building height. The building height can, for instance, be estimated from the number of floors (as it can be found in a TIS used for regional planning) multiplied by an average height per floor or by stereoscopically measuring one gutter point per building. Thus, a simple 3D city model consisting of prismatic buildings can be created rather easily from existing data, the level of detail being sufficient for many purposes [Kraus and Ries, 1999]. However, as the level of detail shall be increased so that the raw shape of the roof is to be modelled, too, sweeping techniques can no longer be used. For instance, a simple hip-roof building cannot be represented by sweeping a simple shape along a translation vector.

For that purpose, *CSG* and *boundary models* are the most commonly used modelling techniques in building reconstruction [Englert, 1998, Müller, 1998, Veldhuis, 1998, Rottensteiner, 2000]. The philosophy of CSG corresponds well to the way knowledge about building shapes can be represented in building reconstruction systems (cf. chapter 6). A minimum set of parameters can be chosen for the description of each primitive, and symmetries are thus modelled implicitly, which is a great benefit for the convergence of automation tools [Veldhuis, 1998]. As primitives can be combined in order to model more complex buildings, the number of primitives required is considerably reduced with respect to the primitive instancing scheme described in section 2.3.3. However, as stated in section 2.3.4, CSG trees are not easily visualized. For that purpose, boundary models can be used efficiently (cf. section 2.3.1.2). In contrast to CSG primitives, boundary models are guaranteed to be easily transferable to many other programs, which is another reason for the wide range of applications of this modelling technique, especially in CAD. These advantages of boundary representations are contrasted by their complexity: as already stated in section 2.3.1.2, a solid object such as a building can hardly be created from scratch without additional user interfaces. In addition, symmetries of object parts cannot be modelled easily; they would have to be added to the geometrical descriptions of a model [Veldhuis, 1998], thus again increasing the complexity of the model. In order to overcome the drawbacks of both CSG and B-rep, *hybrid modelling schemes* are commonly used in building extraction. According to [Mäntylä, 1988], there are two major architectures of hybrid modellers, depending on which modelling technique is chosen to be the *primary* representation:

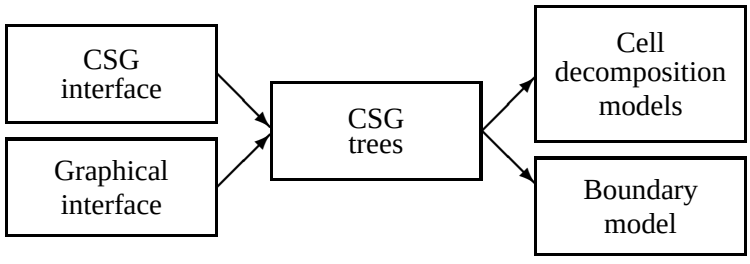


Figure 2.18: Architecture for a CSG based hybrid modeller according to [Mäntylä, 1988].

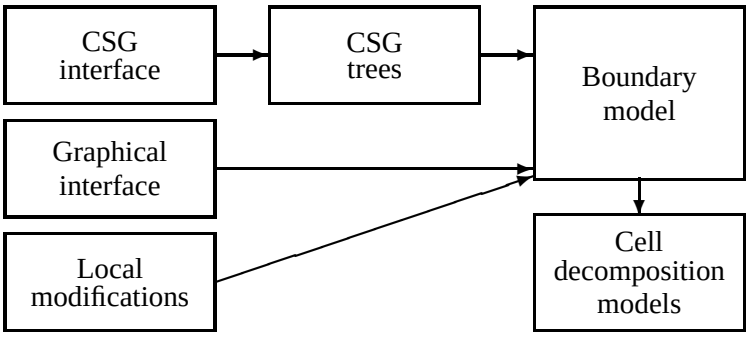


Figure 2.19: Architecture for a B-rep based hybrid modeller according to [Mäntylä, 1988].

1. CSG is the primary representation technique (figure 2.18): The CSG trees are the basic data structure. A CSG interface provides the possibility to select and manipulate primitives. Any other representation can be derived from that representation after each modelling step. For instance, a B-rep can be generated by boundary evaluation (cf. section 2.3.4), but the user has no direct access to that structure, even though internally, the B-rep can be maintained for visualization purposes. This architecture is used in the system *HASE*$^+$ for semi-automatic building reconstruction [Englert, 1998].

2. B-rep is the primary representation technique (figure 2.19): In this case, CSG is only used as an auxiliary method for the representation of a priori knowledge. Visualization is directly based on B-rep, and additional modelling techniques (e.g., sweeping operations) or local modifications might be realized by applying Euler operators to the B-rep data structure in the way described in section 2.3.1. Any other representation technique (e.g. a decomposition model) can be derived from the B-rep. In our system for semi-automatic building extraction, we use a similar architecture for hybrid modelling even though facilities for local modifications are provided ([Rottensteiner, 2000]; cf. chapter 6).

Chapter 3

Data management in topographic information systems

In chapter 2, modelling techniques for topographic objects have been presented and discussed. In this chapter, we will concentrate on how a great number of instances of various types of topographic objects can be collected and maintained in a topographical information system. In this context, it is our goal to obtain a hybrid representation of topographic objects in an object-oriented developing environment for visualization purposes even though using existing technology for data storage. We emphasize that it should be possible to handle huge amounts of data with our solution, for instance, topographic data available for a whole state. From the point of view of building extraction, this chapter deals with what is going to happen to the building models as soon as reconstruction itself is finished because the production of data cannot be a goal in itself. We will start with an overview on some basic concepts of topographical information systems in section 3.1. After that, our way of providing data for a topographical information system will be described in section 3.2.

3.1 Topographic information systems

According to [Kraus, 2000], a *Topographical Information System* (*TIS*) is defined as a computer based system for collecting and maintaining, storing and reorganizing, modelling and analyzing topographic data as well as for the multi-media representation and visualization of these data. Figure 3.1 depicts the components of a TIS. A TIS consists of a topographical data base and application programs. The topographical data base itself consists of the data in the form of topographical models and of the data base management system (DBMS) which provides the means of communication between data and the application programs. The application programs have to provide techniques for data analysis as well as for the integration of additional data from external data bases. They have to be designed in a way that enables them to provide information for the user: the user formulates queries to the information system, and the answers to his or her question provided by the information system have to be presented to him or her either graphically or alphanumerically.

With respect to the architecture of a data base system, three layers can be distinguished [Bill and Fritsch, 1991] (figure 3.2):

- The *external scheme* is the one directly visible for the user. It provides the data in one of the modelling techniques described in chapter 2, because these data structures are best suited for graphical representation. In order to make the implementation of new object types in the application programs easier, object-oriented program design should be used. Note that the objects of the external scheme cannot be obtained directly from the data base, but the data can only be retrieved from the data base via the conceptual scheme. Thus, the data have to be prepared in a special way. This is done by the data base management system.

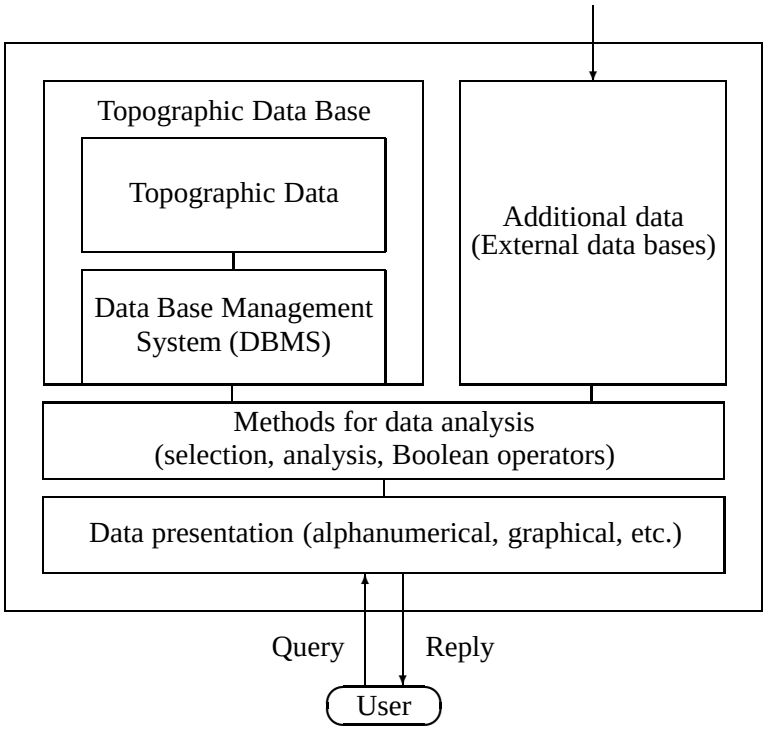


Figure 3.1: Components of a topographic information system (TIS) according to [Kraus, 2000]

- The *conceptual scheme* or *logical data base model* is especially important because it has to provide the tools for data manipulations and transactions in a way to ensure integrity of the data. It is also responsible for providing interfaces to the application programs.
- The *internal scheme* or *physical data base model* defines the actual way the data are organized, for instance in the form of files, lists, or tables.

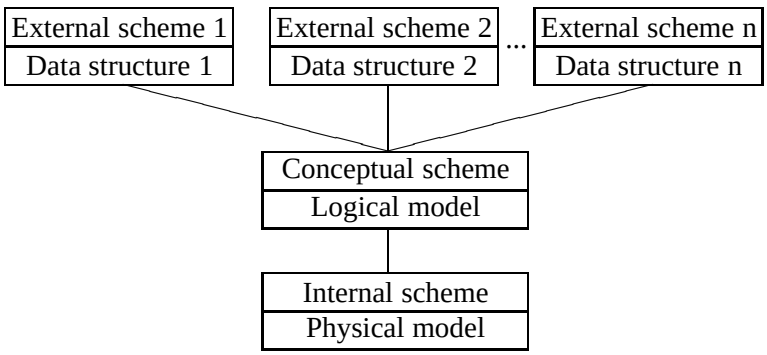


Figure 3.2: A model of a data base system according to [Bill and Fritsch, 1991]

According to [Bill and Fritsch, 1991], data base systems should fulfil certain requirements:

- There should be no redundancy in the data.
- Access to the data should be fast.
- The integrity and security of the data has to be guaranteed.
- The structure of the data models should be flexible.

- Both interactive and batch mode should be possible.
- Several users should be able to use a data base simultaneously.
- A user-friendly interface and user-friendly tools are required.

In the following sections we will give a short overview on logical data models (section 3.1.1) and on physical data models (section 3.1.2). In section 3.1.3, we want to describe an existing topographic data base management system capable of handling geometrical and topological data.

3.1.1 Logical data base models

This section is intended to give an overview on logical data base models with the aim of explaining the terms in a short way for later use in the text. The contents of this section are based on [Kraus, 2000] and [Bill and Fritsch, 1991]. According to both references, four logical data base models are commonly used:

- Hierarchical data base models
- Network based data base models
- Relational data base models
- Object oriented data base models.

Hierarchical data base models: Hierarchical data base models are well suited for data which are in some way related hierarchically. The data base model explicitly makes use of the hierarchies by providing a tree by which all data are accessible. The tree consists of a *root* representing the highest level in the hierarchy, sub-trees representing intermediate levels of the hierarchy, and *leaves* containing the actual data. Between tree nodes of consecutive levels, there is a "father-son"-relationship: each node can have n child nodes which, in turn, can be father nodes of sub-trees. In hierarchical data base models, each child node can only have one father node. The links between the nodes of the tree are realized by *pointers*. Hierarchical data base models offer a very efficient access along the links of the tree, thus through the hierarchy. It can be made even more efficient if the internal scheme of the data base uses some ordering criterion. With respect to topographical data, a sub-division of 2D space within the region of interest by a grid of varying grid width can provide a hierarchy rendering possible fast access to topographic objects by their co-ordinates. However, access to data via thematic attributes will become very inefficient, and in some cases, hierarchical data base models result in redundant data.

Network based data base models: These data base models are based on an enlargement of the concept of hierarchical data bases. Instead of one single hierarchy, several hierarchical access paths can be provided. In this way, redundancy in the data can be avoided. In contrast to hierarchical data base models, one child node can have more than one father node. Again, the links in the hierarchy are realized by pointers. Access via the pointers of the network is quite fast whereas access via other attributes becomes inefficient. Network based data base models are very flexible, and they are often used for topographic data bases, especially for storing the geometry and topology of topographic objects. As the pointers have to be updated if new data are inserted or old data are deleted, they should only be used for more or less static data.

Relational data base models: In relational data bases, all data are stored in *tables*. Each table is assigned a name. Its rows contain objects of identical type; the definitions of its columns (i.e., the column names and the data types of the column contents) describe the attributes of each of the instances of the object. By the enumeration of its name, its column names and the data types of the column contents, a table is completely described. Queries to a relational data base are formulated in a query language, for instance in *SQL* (*Structure*

Query Language) which is an international standard and has found widespread applications in commercial relational data base systems. The data types offered by SQL can be classified as character arrays (names), numbers, logicals and data types related to date and time. Tables can be modified or combined by several operations of *relational algebra* such as the application of Boolean operators, projection (i.e. selection of columns) or the Cartesian product. The relational data base concept offers several advantages:

- Data base queries are based on a comparison of the table contents. Thus, no pointers are required in relational data bases, and all relations are treated uniformly.
- The tables are independent and not related by pointers. That is why it is easy to maintain dynamic data sets.
- The tables can be easily expanded by new columns.
- It is easy to create user specific views from relational data bases.

These advantages are contrasted by slower access rates compared to hierarchical or network based data bases. Faster access rates can be achieved by the indexing techniques described in section 3.1.2 which, however, require additional disk space. Access to topographic data by geometrical attributes is not optimal in the classical relational data model. Expansions of that concept by new topological data types can be used to overcome that problem (cf. section 3.1.3).

In order to avoid inconsistent data (anomalies), normalization rules have been defined. The most important one, the first normal form, applies to relations only containing simple attributes (*atoms*) in each column. This means that, for instance, a surrounding polygon cannot be a single attribute of a piece of land. In this case, it would be necessary to create a table containing all points and one table for each polygon, the lines containing the point identifiers of the respective polygon. From the example it can be seen that when applying the first normalization rule, one ends up with a great number of tables.

We will come back to the application of relational data bases for TIS in sections 3.1.3 and 3.2.

Object oriented data base models: Object oriented data base systems are the most recent development in data base technology. It is their goal to offer more complex data types in order to overcome the restrictions of, e.g., the normalization rules for relational data bases. In the context of object oriented data base models, an *object* consists of both its data and the operations (methods) which can be applied to the object. Each object is a concrete *instance* of an *object class* defining the attributes and operations of all its instances. Each instance has its unique identifier by which it can be referred to in the data base. The concept of object oriented data bases is well suited for representing topographic objects as they are defined in chapter 2. The following principles apply to object oriented data base models (and to object oriented programming [Meyer, 1990]):

- *Encapsulation*: An application (another object) can only communicate with an object via *messages*. The operations provided by an object define the set of messages which can be understood by it; no other operations can be applied to an object.
- *Inheritance*: New object classes can be derived from another class (the *super-class*) by inheritance. The new classes inherit the attributes and methods of the super-class and offer additional attributes and operations. The relation between a derived class and its super-class is called *"isA"* relation because an instance of the derived class also *is a*n instance of the super-class.
- *Polymorphism*: This feature is closely connected to inheritance. Derived classes may re-define methods of their super-class(es). This is very useful for achieving class-specific behaviour using messages already available for the super-class.

- *Aggregation*: Composite objects may be constructed as consisting of a set of elementary objects. The *container object* can communicate with its *contained objects* via their methods. The relation between the container object and its components is called *"partOf"* relation because a component is a *part of* the container object.

As operations are an integral part of objects and because the actual implementations of the operations are hidden to an application, objects can be used more easily by application programs. It is possible to provide an object class as an abstract description for a wide variety of actual objects and derive new classes from the base class. Any application knowing the abstract description and using only the methods provided by it will still be able to handle objects of the derived classes which inherit these methods and, possibly, re-define them. The code of the application thus needs not to be altered. In this sense, object oriented data bases can be used more easily by any application than relational data bases where all the operations have to be realized in programs outside the data base. However, object oriented data bases are not yet as widely used in commercial products as relational data bases. Even though the "relational" and the "object oriented" principles contradict to each other (e.g., it is hard to imagine encapsulation in a relational data base where the data belonging to an object can be distributed in a considerably large number of tables), there are efforts going on to combine the advantages of the wide acceptance of relational data bases and the benefits of the object oriented paradigm in *object-relational* data base systems.

3.1.2 Physical data models

Physical data models describe the actual way of physically storing data on a secondary data storage device, e.g. on hard disc. The simplest way of doing so is storing all data sequentially on a file. However, as these files might become quite large, the data cannot be kept completely in the computer memory. In addition, data access will become rather slow, even more so if the data are not sorted. That is why the file has to be split into separate blocks, each block being assigned a unique identifier. A *hash table* which has to be stored separate from the data contains the key of the first record of each block. In order to access a specific record, the according block identifier has to be searched for in the hash table. After that, that block has to be loaded into memory, and the record has to be searched for within the block. This principle renders possible fast access to the data via the key element, but it is very slow for access via other attributes. It can be improved by assigning a unique index to each record and including additional files (*inverted files*) for all attributes which shall work as keys, each of the inverted files containing only records consisting of two elements, the attribute serving as a key and the index to the original data [Kraus, 2000].

Specific structures for spatial data: The principles for blocking described above are not appropriate for fast access to spatial data by their geometrical attributes. Typically, tree structures such as the quadtree (for 2D and 2.5D data) or the octree (for 3D data) are often used for blocking spatial data [Bill and Fritsch, 1991]. The *extendible cell (EXCELL) structure* divides the region of interest into an irregular raster based on a binary tree [Kraus, 2000]. The *gridfile technique* is a very efficient structure being well suited for large irregularly distributed data sets (figure 3.3). In the 2D case, the region of interest is subdivided into rectangular blocks. The partitioning is derived by binary decomposition of 2D space. The data are stored in units called *buckets*, each being defined to contain a maximum of n points. Each bucket is assigned a unique identifier (figure 3.3, left). The limits of the buckets define an irregular rectangular grid described by the co-ordinates (x_i, y_i) of the grid lines. Each of the rectangular blocks is assigned a bucket. Note that more than one block can be assigned a bucket, but not vice versa. (figure 3.3, centre). In addition to the buckets and to the subdivision limits (x_i, y_i), the *gridfile address matrix* which describes the relation between the grid blocks and the data buckets (figure 3.3, right) has to be stored. Usually, the subdivision limits can be kept in memory, whereas the address matrix and the buckets are kept on hard disc. In order to access a point by its position, first the correct indices of the address matrix have to be found by binary search in the subdivision limits. After that, the corresponding element of the matrix (i.e. the bucket identifier) has to be read from disc. Using the bucket

identifier, the bucket can be read from disc, too, and the relevant records can be searched for inside the bucket [Bill and Fritsch, 1991, Kraus, 2000].

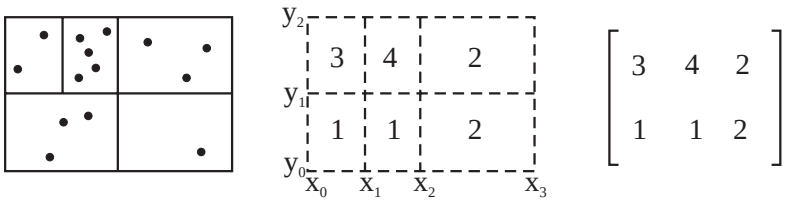


Figure 3.3: The gridfile structure. Left: Data distribution and data blocks (5 points maximum); centre: the irregular rectangular partitioning with interval limits (x_i, y_i); right: the gridfile address matrix. Adapted from [Kraus, 2000].

3.1.3 TOPDB: A relational data base with topological elements

In section 3.1.1, a short overview on relational data base management system has been given. It was stated there that relational data bases are not very efficient with respect to geometrical and topological information. In order to overcome that problem, the relational database management system *TOPDB* was developed at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology [Loitsch and Molnar, 1991]. *TOPDB*, in addition to the usual data types handled by *SQL*, e.g., *STRING*, *NUMBER*, *DATE*, etc., offers 2D topological data types and topological operators. In order to handle these features, a new query language had to be defined. This language called *TOPSQL* consists of a subset of the *SQL* standard and some extended statements necessary to take advantage of the new features. The topological types are

- *AREA*: a closed polygon
- *LINE*: a polygon which is not necessarily closed
- *POINT*: a point
- *WINDOW*: a rectangle with sides parallel to the respective co-ordinate axes.

These type definitions mean that, for instance, an entire closed polygon may represent just one element in a column. In addition to that, it is another non-standard feature of *TOPDB* that it provides the attribute *ARRAY* for handling an arbitrary number of identical topological types in a single column element. Note that some of these attributes contradict the first normal form of relational data bases (cf. section 3.1.1). Still, they are required for fast access to topographic data.

In order to exploit the specific properties of the topological data types and in order to use them for data base queries by geometrical/topological criteria, topological operators had to be specified which describe relations between two instances of these types:

- *.X.* The geometrical intersection between two topological elements. The operator will return "*TRUE*" even if the two elements just touch each other. For instance, the intersection of two *AREA*s is another *AREA*.
- *.P.* Similar as $.X.$, but always concerning just the vertices on both sides. The existence of any identical point is checked.
- *.XP.* Similar to $.X.$ for the operand on the left, and similar to $.P.$ for the operand on the right.
- *.PX.* Symmetrical to $.XP.$

- .>. Yields "*TRUE*" if the element to the right is "fully enclosed by" the one on the left.
- .<. Symmetrical to . > .
- .=. satisfied by identical elements only.

Even though *TOPDB* offers a relational data base model, the access to the new topological data types can be very efficient because the gridfile method (section 3.1.2) is used as a physical data model. The number of tables as well as the numbers of lines and columns which can be handled by *TOPDB* is, in principle, unlimited [Loitsch and Molnar, 1991]. In section 3.2 we will see how *TOPDB* can be used to handle topographic data in TIS on a national scale.

3.2 Managing hybrid topographic data in a country-wide TIS

As we have seen in section 3.1, depending on the type of object which is to be described, different modelling techniques are appropriate. The data structures corresponding to these modelling techniques are especially important for graphical visualization of results (figure 3.1). Using different modelling techniques for different object classes renders possible visualizations such as the perspective view in figure 3.4 which contains a 2.5D grid-based DTM with buildings in B-rep. A concept for such a structure called *irregular tiling* was presented by [Molnar et al., 1996]. In the context of data base architecture described in section 3.1, these modelling techniques correspond to the external scheme: they have to be provided for any application (e.g., a visualization tool) by the data base management system. In order to make the application software independent from the modelling techniques and thus more extensible, an *object oriented* external scheme is desirable.

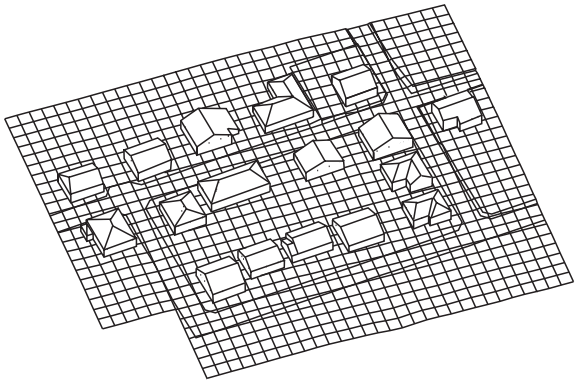


Figure 3.4: A visualization of a scene using hybrid object representation: Boundary representation for houses and a hybrid high-quality 2.5D DTM for the terrain.

With respect to the conceptual and internal schemes, existing relational data bases are still used. Taking into account the difficulties of these systems with handling topographic data (cf. section 3.1.1), these data are often separated from the thematic data. The topographic data can, for instance, be kept in a CAD system, where the individual objects can be accessed by their identifier. In the relational data base, the topographic objects are then represented by their identifier and a representative point, e.g. by the centre of gravity [Amhar, 1997]. There are examples for TIS in urban regions making use of relational data bases in different ways:

In [Koehl and Grussenmeyer, 1998], a fully relational concept for storing both B-rep of buildings containing additional thematic attributes alongside and a DTM in a TIN structure is presented. It is the main advantage of this concept that standard relational data base systems can be used. However, this advantage is contrasted by the fact that, using that concept, the data representing a single object, e.g. a building, are distributed over several tables: there has to be a table of solids, a table of faces, a table of loops, a table of edges, and a table of

points. In order to perform an operation using a certain object, first an application has to construct the object from that distributed data, which is very inefficient for large data sets.

In the system *CyberCity Modeler* for semi-automatic building extraction, a relational data model is used not only for storing the building models, but also for storing terrain and image data [Wang and Gruen, 1998].

[Amhar, 1997], in his attempt to produce digital orthophotos with correctly positioned buildings, uses a grid DTM as provided by the program *SCOP* [SCOP, 1994] and a digital building model based on the topological features of *TOPDB* (cf. section 3.1.3). He creates a table of type *TDBRPTAB* (table 3.1). Each line in that table corresponds to a single building in B-rep. The vertices of the building are contained in columns *EAVES* and *DETAIL*. The vertices contained in *EAVES* define the building outline, whereas *DETAIL* contains features such as the central ridge. In columns *WALL* and *ROOF* the topology is contained. Each face is contained in one of these columns as a series of indices of the *EAVES* and *DETAIL* columns. With respect to access rates, this concept is more efficient than the fully relational one described above. However, the division of vertices into *EAVES* and *DETAILS* is not a very clear one, and the modelling concept is not a very general one. For instance, neither loops nor features such as dormers can be handled by it.

```
IDBOBJ INTEGER UNIQUE INDEX NOT NULL IDENTIFIER
OBJTYP CHAR(16)       INDEX NOT NULL
EAVES  AREA           INDEX NOT NULL PERIOD(3) RESOLUTION(2,2,2)
DETAIL LINE           INDEX NOT NULL PERIOD(3) RESOLUTION(2,2,2)
WALL   INTEGER     NO INDEX     NULL ARRAY
ROOF   INTEGER     NO INDEX     NULL ARRAY
STATUS CHAR(16)       INDEX     NULL
```

Table 3.1: Definition of *TOPDB* tables of type *TDBRPTAB*. Taken from [Amhar, 1997]

[Yang et al., 2000] also use a relational data base system for managing 3D building models. In order to overcome the deficiencies of relational data bases with respect to topographic data, they treat both the geometry and the topology of buildings as *binary large objects* (*BLOB*s). The relational data base system can no longer perform operations on BLOBs, and additional programming efforts are required for storing building models to / retrieve them from the data base. A tree-like structure is used for structuring these data by geometrical criteria in order to increase access rates. The advantage of fast access is contrasted by the programming effort required to interpret the BLOBs and by the problems encountered when exchanging data with other systems.

3.2.1 SCOP.TDM: Country-wide management of digital terrain data

At the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology and at INPHO company, Germany, the program system *SCOP* for the generation and visualization of DTMs has been developed. The program has a module called *SCOP.TDM* which is capable of managing country-wide digital terrain data [Hochstöger, 1996]. *SCOP.TDM* is a specific application of *TOPDB* (cf. section 3.1.3) especially designed for storage and archiving of topographic data and digital terrain models. The program offers two working areas, each of them corresponding to an area on the hard disk:

1. The *topographic data market* consists of a system of tables for managing arbitrarily distributed topographic data as they are used to derive DTMs. In other words, the original elevation data (3D bulk points) as well as geomorphological data (e.g. break lines, spot heights) are managed in this working area. Along with the geometry and the topology of these features, meta data describing the object type, accuracy and other attributes are contained in these tables.

2. In the *derived products market*, secondary data are managed. In the current version of *SCOP.TDM*, the term “secondary data” just comprises DTMs derived from the original elevation and geomorphological data. The derived products market also consists of a system of tables which, however, only contain the meta data of the DTMs whereas the DTMs themselves are stored separately on the disk.

In the derived products market, the meta data of the DTMs are stored in tables of type *DPMFLTAB* (table 3.2). Each DTM is assigned a unique identifier (*IDDPMFILE*). It is stored on a file described by its file name (FILE) which can be used by application programs to directly access the data. Column *PRODUCTTYPE* describes the object type; up to now, the only object type available is "*DTM*". The format of the file containing the DTM is stored in column *DATAFORMAT*. It can, for instance, be the binary format of *SCOP* (*RDH*) or *ARC/INFO GRID*. *COORDSYSTEM* contains the type of the object co-ordinate system (e.g. UTM). Columns *XYEXTENSION*, *ZMINIMUM* and *ZMAXIMUM* describe a 3D bounding box parallel to the axes of the co-ordinate system containing the whole DTM. As these data are contained in the relational data base, they can be used for queries using *TOPSQL*. They serve as a kind of representation of the DTM in a very coarse level of detail, for instance in order to select only those DTMs from the data base which are inside a certain region of interest without actually having to access the DTM data files. Columns *RESOLUTION* and *ACCURACY* describe the grid size and the accuracy of the DTM, respectively. Finally, there are meta data such as the project name, the name of both creator and owner of the DTM, the date and time of creation, and information regarding the way the data were originally captured (*COMPILEMODE*).

```
IDDPMFILE    INTEGER       UNIQUE INDEX  NOT NULL  SYSNUM IDENTIFIER
DATAFORMAT   CHAR(16)              INDEX  NOT NULL
FILE         CHAR(64)              INDEX  NOT NULL
PRODUCTTYPE  CHAR(32)              INDEX  NOT NULL
COORDSYSTEM  CHAR(32)              INDEX  NOT NULL
RESOLUTION   NUMBER(12.6)          INDEX  NOT NULL
XYEXTENSION  WINDOW                INDEX  NOT NULL  RESOLUTION(2,2)
ZMINIMUM     NUMBER(12.2)          INDEX  NOT NULL
ZMAXIMUM     NUMBER(12.2)          INDEX  NOT NULL
ACCURACY     NUMBER(12.2)          INDEX      NULL
PROJECT      CHAR(32)              INDEX      NULL
CREATOR      CHAR(32)              INDEX      NULL
OWNER        CHAR(32)              INDEX      NULL
COMPILEMODE  CHAR(32)              INDEX      NULL
CREATIONDATE DATE                  INDEX      NULL
CREATIONTIME TIME                  INDEX      NULL
```

Table 3.2: Definition of *TOPDB* tables of type *DPMFLTAB*; simplified according to [Hochstöger, 1996]

The concept of the derived product market can be seen as a realization of the principle of BLOBs, too: the tables of type *DPMFLTAB* just contain meta data of the DTMs, among them the name of the data files containing the DTMs. The file names represent the actual BLOB (the contents of the data file). *TDM* offers tools for selecting DTMs from the derived product market using *TOPSQL*, and it can provide these data to application programs, e.g. the *SCOP* modules for creating visualizations of DTMs such as digital orthophotos, contour lines, hill-shaded or height-coded maps, or perspective views. We shall expand this concept to other types of topgraphic objects in section 3.2.2.

3.2.2 Object oriented modelling and a relational data base: a hybrid approach

In order to provide a common object oriented interface for programs visualising topographic data, the common features of all topographic objects have to be collected in an abstract class *topographicObject*. These features comprise the data which are, basically, identical to the data representing one row in tables of type *DPMFLTAB* in *SCOP.TDM* (table 3.2), and the operations which can be performed with such objects. These operations consist of simple geometrical operations such as computing both area covered by an object and its volume, computing the intersection points of the object with a 3D line, deciding whether a point is inside the object or not, etc. The description of class *topographicObject* is given in table 3.3.

Class topographicObject			
Member	Type	Description	
identifier	int	unique identifier of the object	Data members
className	string	identifier of the object class	
fileName	string	file containing the object's persistent form	
dimension	int	dimensionality of the object	
boundingBox	BBox	co-ordinate ranges of the object	
LoD	float	level of detail.	
accuracy	float	accuracy of the object	
coordSystem	string	name of the co-ordinate system	
projectName	string	project name	
creatorName	string	name of the creator of the object	
ownerName	string	name of the owner of the object	
compileMode	string	a string describing the generation process	
creationDate	date	date of creation of the object	
creationTime	time	time of creation of the object	
getConnection()	int	number of disjunct object parts	Poly-morphic function members
store()	void	store the object to a file (make it persistent)	
retrieve()	void	retrieve the object from a file	
getArea()	float	area covered by the object	
getVolume()	float	volume contained in the object	
isContained(point)	bool	is *point* contained in the object?	
intersect(line,vector)	int	number of intersections of *line* with the object. The intersection points are returned in *vector*	
intersect1(ray,point)	int	returns the intersection closest to the first point of *ray* in *point*	

Table 3.3: Description of class *topographicObject*. The list of member functions is not complete.

Note that the data members correspond exactly to the columns of *DPMFLTAB*, with a few exceptions. For instance, columns *XYEXTENSION*, *ZMINIMUM* and *ZMAXIMUM* are represented by an object of type *BBox* representing a 3D prism parallel to the co-ordinate axes. The level of detail is described by the linear extent of the largest detail NOT described by the given representation. The member functions of the object have to be designed in a way to render possible an efficient way of visualizing the object. In this context, the list of member functions given in table 3.3 has to be seen as an example rather than a complete enumeration of all geometrical methods.

The operations *store()* and *retrieve()* are of special interest: *store()* is responsible for making a topographic object persistent by writing it to a file in some format, whereas *retrieve()* will read such a file and reconstruct the object in memory from its persistent form. Note that in table 3.3 no assumption is made with respect to the actual internal representation of the object geometry. This means that none of the methods of that class can already be implemented: they are all polymorphic in the sense of object oriented programming (section 3.1.1). The class description of *topographicObject* as depicted in table 3.3 just represents an abstract interface for application programs. It has to be made available to them in an object oriented programming language such as C++. The actual implementation of the data structures and methods of *topographicObject* has to be left to derived classes.

Figure 3.5 shows an inheritance tree for topographic objects derived from the abstract base class *topographicObject*. Note that it is possible to hide existing code for specific modelling technique behind the

interface of the derived classes. The derived classes are:

- *topographicObject2D*: this class describes topographic objects in 2.5D representations, especially DTMs as described in section 2.2. It is still an abstract class and provides some additional features common to 2.5D representations. The descendants of *topographicObject2D* are already concrete realizations of topographic object modelling techniques. These are the descendants of class *topographicObject2D*:
  - *rasterDTM*: a hybrid grid-based DTM as described in section 2.2.1. The actual implementation makes use of subroutines of the program system *SCOP*.
  - *TINDTM*: a DTM represented by a TIN as described in section 2.2.2.
- *topographicObject3D*: this class describes topographic objects in 3D representations as described in section 2.3. It is also an abstract class and provides some additional features common to 3D representations. The descendants of *topographicObject3D* are already concrete realizations of solid object modelling techniques. Currently, there is only one descendant:
  - *topographicObject3DwithBrep*: the only class derived from *topographicObject3D* representing an actual modelling technique. It encapsulates a B-rep using the full winged-edge data structure (section 2.3.1).

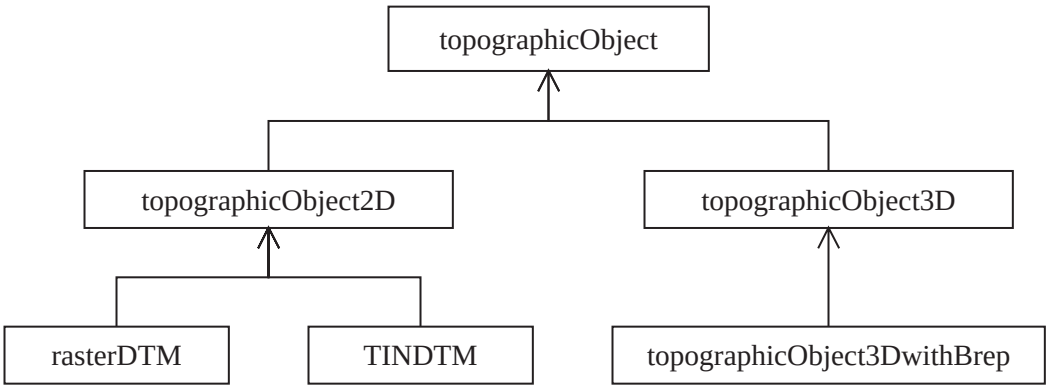


Figure 3.5: An inheritance tree for topographic objects. The arrows symbolize inheritance relations.

Class *topographicObject* provides an abstract interface describing topographic objects. Any application has to use the methods declared in the abstract interface for its own purpose, the result of this concept being that as soon as a new descendant of class *topographicObject* is implemented, the application can immediately handle that class, too. There is only one place where the existing code has to be modified: at some place in the software, an instance of the new class has to be created upon request. This place is inside another class which is responsible for the application view of the management of topographic objects and for the interfacing between the topographical data base (in our case *SCOP.TDM*) and the applications. This class is called *topographicObjectManager*. Its interface is depicted in table 3.4.

Class *topographicObjectManager* provides a list of references to instances of class *topographicObject* (*objectList*), and it offers methods for accessing, modifying and removing this list. Method *createObject* is a *polymorphic* or *virtual constructor*: depending on the string *className*, a new instance of the class described by that string will be created. The other operations work in close co-operation with the *TDM* interface. An application can tell a *topographicObjectManager* object to select objects according to some (geometric or thematic) criterion. The criterion is passed to the *topographicObjectManager* in a string corresponding to a condition formulated in *TOPSQL*. An appropriate *TOPSQL* statement is sent to *TDM* by method *selectObjects*. *TDM* passes the *TOPSQL* statement to *TOPDB*. The answer is a table which is passed back to the *topographicObjectManager* via *TDM* in a specific syntax. The *topographicObjectManager* will first

Class topographicObjectManager			
Member	Type	Description	
objectList	topoObjPtrList	list of pointers to *topographicObject*s	Data members
aTDMserver	TDMserver	interface to TDM	
createObject(className)	topoObjPtr	create an instance of class *className*	Function members
insertObject(topoObjPtr)	topoObjPtr	insert *topoObjPtr* to the data base	
selectObjects(SQLstring)	int	select objects from the data base according to *SQLstring*	
removeSelectedObjects()	void	remove the objects currently selected in *topoObjPtrList*	

Table 3.4: Description of class *topographicObjectManager*.

clear its object list. After that, for each line of the table, the contents of column *PRODUCTTYPE* are evaluated, and a new instance of the class described by that string is created using method *createObject*. The new object is then inserted into the object list which can be accessed by the application. In a similar way objects can be inserted into and removed from the data base. The interaction between *TOPDB*, *SCOP.TDM*, the *topographicObjectManager* and the application programs is depicted in figure 3.6.

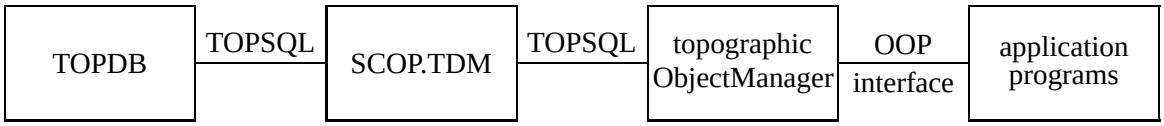


Figure 3.6: The interaction of *TOPDB*, *SCOP.TDM*, the *topographicObjectManager* and application programs. *OOP*: Object oriented programming.

Chapter 4

Photogrammetric data acquisition

In chapters 2 and 3 we have described methods for modelling 3D topographic data and methods for the management of these data in topographic information systems. The 3D co-ordinates of the object points required to describe the models were assumed to be known. However, these co-ordinates can hardly ever be measured directly. There are co-ordinate measurement machines in industrial environments, and there is GPS in surveying, but usually, in the process of data acquisition, the 3D co-ordinates of the object points have to be determined indirectly from other observations obtained with the help of some measuring device. The 3D co-ordinates of the object points alongside with the parameters of the measuring device have to be estimated from the observations in a *parameter estimation* or *adjustment* process. This is especially necessary because, usually, more observations than those required are made in order to increase both reliability and accuracy of the parameters which are to be determined. We will discuss parameter estimation techniques in section 4.1.

In our work, we rely on the hybrid photogrammetric adjustment program *ORIENT* which has been developed at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology since the mid-seventies [Kager, 1989, Kager, 2000]. In this context, the term *hybrid* refers to the fact that various types of observations can be adjusted simultaneously. The term *photogrammetric* should emphasize the fact that camera co-ordinates of perspective images represent our main source of observations in the process of 3D object reconstruction, i.e. the determination of the 3D co-ordinates of the object points. The mathematical model of the relations between observations and the (object and sensor) parameters as it is used by *ORIENT* will be described in section 4.2. Section 4.3 deals with the data structure used by *ORIENT*. This is necessary because this data structure together with the mathematical model is the reason why *ORIENT* is so flexible that it can be applied for object modelling and automation in object reconstruction. In section 4.4 we want to describe a system for interactive measurement in digital images which we use as our working environment for semi-automatic building extraction. Finally, using the theoretical background of the previous sections, section 4.5 will deal with some practical aspects of photogrammetric object reconstruction.

4.1 Parameter estimation

It has been stated above that the observations made by some measuring device are used to estimate parameters, in our case the 3D co-ordinates of object points and the sensor parameters. As the observations cannot be considered to be free of errors, they are modelled as random variables, and their stochastic properties are described by a distribution function. The relation between the observations and the parameters is given by:

$$\mathbf{E}(\mathbf{l}) = \mathbf{l} + \tilde{\mathbf{v}} = \mathbf{f}(\mathbf{x}) \tag{4.1}$$

where

- $\mathbf{l} = (l_1, l_2, \ldots, l_n)^T$ is the vector of observations containing n elements

- $\mathbf{E}(\mathbf{l}) = (E(l_1), E(l_2), \ldots, E(l_n))^T$ is a vector containing the expectation values of the observations $\mathbf{l}$
- $\tilde{\mathbf{v}} = (\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_n)^T$ is the vector of corrections to the observations
- $\mathbf{x} = (x_1, x_2, \ldots, x_u)^T$ is the vector of unknown parameters containing u elements
- $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x}))^T$ is a vector of n functions f_i describing the mathematical relation between the unknowns $\mathbf{x}$ and observation l_i.

As the functions f_i are, in general, not linear, equation 4.1 has to be linearized by a Tailor series expansion using approximate values $\mathbf{x_0}$ for the unknowns $\mathbf{x}$:

$$\mathbf{x} = \mathbf{x_0} + \delta\mathbf{x} \tag{4.2}$$

and

$$\bar{\mathbf{l}} + \tilde{\mathbf{v}} = [\mathbf{l} - \mathbf{f}(\mathbf{x_0})] + \tilde{\mathbf{v}} = \mathbf{A} \cdot \delta\mathbf{x} \tag{4.3}$$

where

- $\mathbf{x_0}$ is the vector containing the approximate values for the unknowns $\mathbf{x}$
- $\delta\mathbf{x}$ is the vector of corrections to the parameters (equation 4.2)
- $\bar{\mathbf{l}} = \mathbf{l} - \mathbf{f}(\mathbf{x_0})$ is the vector of observations reduced by the result of the evaluation of the functions $\mathbf{f}$ at the position $\mathbf{x_0}$. It has the same stochastic properties as $\mathbf{l}$.
- $\mathbf{A}$ is the Jacobian matrix of coefficients containing the partial derivatives of the functions $\mathbf{f}$ by the unknowns x_j: $a_{ij} = \frac{\partial f_i}{\partial x_j}$.

Equation 4.3 describes the Gauss-Markoff model. In general, the number n of observations will be greater than the number u of unknowns. In the case $n = u$, the unknowns can just be determined from the observations, and in case $n < u$, a set of $u - n$ parameters will not be determined at all. The estimation of the parameters is guided by the *maximum likelihood (ML) principle* [Mikhail and Ackermann, 1976]. This principle states that the unknown parameters have to estimated such that the (corrected) observations take the highest probability. Thus, the parameters $\mathbf{x}$ have to be estimated such that the joined conditional probability density $L(\mathbf{l}|\mathbf{x})$ becomes a maximum:

$$L(\mathbf{l}|\mathbf{x}) \longrightarrow max \tag{4.4}$$

Equation 4.4 can be evaluated if the probability density functions of the observations are known. Following the central limit theorem by Gauss, the observations $\mathbf{l}$ and thus also $\bar{\mathbf{l}}$ are in general assumed to be normally distributed with expectation $\mathbf{E}(\mathbf{l})$ and variance-covariance matrix $\mathbf{C_{ll}} = \sigma_o^2 \cdot \mathbf{Q_{ll}}$:

$$\mathbf{l} \sim N(\mathbf{l} + \tilde{\mathbf{v}}, \sigma_o^2 \cdot \mathbf{Q_{ll}}) \tag{4.5}$$

Equation 4.5 describes the *stochastic model* of parameter estimation just as the functions $\mathbf{f}$ in equation 4.1 describe its *functional model*. Assuming the stochastic model of equation 4.5, the ML principle leads to a minimization problem:

$$\tilde{\mathbf{v}}^T \cdot {\mathbf{Q_{ll}}}^{-1} \cdot \tilde{\mathbf{v}} \longrightarrow min \tag{4.6}$$

which is also known as the principle of *least squares adjustment*, e.g. [Gotthardt, 1968]. The solution for the estimated parameters $\delta\hat{\mathbf{x}}$ is given by:

$$\delta\hat{\mathbf{x}} = (\mathbf{A}^T \cdot \mathbf{Q_{ll}^{-1}} \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^T \cdot \mathbf{Q_{ll}^{-1}} \cdot \bar{\mathbf{l}} \tag{4.7}$$

After having solved equation 4.7, the corrections $\tilde{\mathbf{v}}$ can be computed from equation 4.3. Using these corrections, an estimate $\hat{\sigma}_o$ for the a posteriori r.m.s. error of the weight unit can be computed from equation 4.8:

$$\hat{\sigma}_o = \sqrt{\frac{\tilde{\mathbf{v}}^T \cdot {\mathbf{Q}_{\mathbf{ll}}}^{-1} \cdot \tilde{\mathbf{v}}}{n-u}} \tag{4.8}$$

Using that estimate for the r.m.s. error of the weight unit, the variance-covariance matrix of the unknowns, $\mathbf{C_{xx}}$, can be derived:

$$\mathbf{C_{xx}} = \hat{\sigma}_o^2 \cdot \mathbf{Q_{xx}} = \hat{\sigma}_o^2 \cdot (\mathbf{A}^T \cdot \mathbf{Q_{ll}^{-1}} \cdot \mathbf{A})^{-1} \tag{4.9}$$

Equations 4.5 to 4.9 describe the general case of ML estimation assuming normally distributed observations. In these equations, $\mathbf{Q_{ll}}$ is a matrix of dimension $n \times n$. The inverted matrix ${\mathbf{Q_{ll}}}^{-1}$ can be replaced by the *weight matrix* $\mathbf{P}$ in those equations. The diagonal elements $q_{l_i l_i}$ of $\mathbf{Q_{ll}}$ consist of the variances σ_i^2 of the observations divided by the a priori r.m.s. error of the weight unit σ_o^2, and its off-diagonal elements correspond to the co-variances of the observations in a similar way, these co-variances being linked to the correlations between the observations. In reality, these correlations are difficult to estimate. In addition, the inversion of a matrix having the dimension of the number of observations is a very time consuming task. That is why many least squares adjustment programs assume the observations to be uncorrelated, i.e., $q_{l_i l_j} = 0$ for $i \neq j$. In this case, $\mathbf{P}$ is a diagonal matrix, its diagonal element p_i being called the *weight* of observation i:

$$\mathbf{P} = \mathbf{Q_{ll}^{-1}} = diag(p_i) = diag\left(\frac{\sigma_o^2}{\sigma_i^2}\right) \tag{4.10}$$

If the functional model in equation 4.1 is non-linear, adjustment has to be performed iteratively. The estimates $\delta\hat{\mathbf{x}}$ for the parameter corrections are added to the approximate values $\mathbf{x_0}$ (equation 4.2), and the resulting parameter vector is now used as an approximation during the next iteration. The iteration process is finished as soon as some stopping criterion is fulfilled, e.g. if the norm of the parameter correction vector is below a certain threshold, if the (user-defined) maximum number of iterations has already been performed, or if the estimate for $\tilde{\mathbf{v}}^T \cdot {\mathbf{Q_{ll}}}^{-1} \cdot \tilde{\mathbf{v}}$ does not change any more [Kraus, 1997]. The determination of approximate values for the unknown parameters is often a very critical step as it is very hard to find a general solution working in all situations. Divergence of the iterative strategy will be a consequence of bad approximate values.

Up to now we have considered the observations to be normally distributed and free of gross errors. However, least squares adjustment might fail completely in the presence of such errors. That is why we will have a closer look at estimation techniques which are robust with respect to the influence of gross errors to the results.

4.1.1 Robust estimation techniques

By the term *robust estimation* we mean estimation techniques which are robust with respect to the presence of gross errors in the data. In this context, gross errors are defined as observations which do not fit to the stochastic model of parameter estimation. Least squares adjustment as described in the previous section is not a robust estimation technique: false observations (e.g., point numbering errors in photogrammetric plotting) can lead to completely false results and might even prevent convergence of adjustment. [Förstner, 1998] lists four robust techniques for parameter estimation:

- *Clustering*: This method is based on the determination of the probability density function $d_p(\mathbf{x})$ under the assumption that the observations represent the complete sample. $d_p(\mathbf{x})$ is represented by an accumulator in parameter space. The estimated values for the parameters correspond to the point of maximum probability in parameter space. Clustering is well-suited for problems with a high percentage of gross errors and a high relative redundancy. However, the dimension of the accumulator is equal to the dimension of

the parameter space, i.e., to the number of unknowns, which causes the method not to be applicable for problems with a great number of unknowns.

- *Random sample consensus (RANSAC)*: This technique is based on the principle of hypotheses generation and verification. The following steps have to be performed:

  1. Choose a minimum set of u observations from $\mathbf{l}$
  2. Determine $\mathbf{x} = f^{-1}(\mathbf{l})$ from the minimum set of observations. Thus, the functional model (equation 4.1) must be invertable.
  3. Check the other observations based on the prediction errors
  4. If the number of accepted observations is high enough, then stop, else go to step 1.

  The set of observations can be chosen randomly. If an estimate for the percentage of gross errors is available in the data, the number of trials required for finding a correct subset of the observations with a pre-defined probability can be estimated. Again, RANSAC is well-suited for problems where the number of unknowns is small.

- *Maximum likelihood (ML) type robust estimation*: ML-type robust estimation techniques are based on the ML principle assuming other distributions than the Gaussian one for the observations. We will have a closer look at this class of ML-type robust estimation techniques below.

- *Iterative elimination*: Eliminate possibly wrong observations iteratively on the basis of a statistical test. The residuals $\tilde{\mathbf{v}}$ should not be used for testing. We shall see a better test statistic below in the section on *data snooping*.

4.1.1.1 Robust estimation based on the maximum likelihood principle

Assuming the observations to be normally distributed, the maximum likelihood principle results in the minimization of the weighted square sum of the corrections (equation 4.6), as we have seen above. Assuming the observations to be uncorrelated (equation 4.10), this is equivalent to minimizing the sum of the squared normalized discrepancies d_i or the *L_2-norm* of the normalized discrepancies:

$$\sum_i d_i^2 \longrightarrow min \tag{4.11}$$

where the normalized discrepancies are the corrections divided by the r.m.s. error of the according observation:

$$d_i = \frac{\tilde{v}_i}{\sigma_i} \tag{4.12}$$

Normalization of discrepancies is necessary to be able to compare the residuals of different observation types: whereas $\tilde{v}_i$ has the dimension of the observations (e.g. [meter] for control points and [pixels] for co-ordinates measured in digital images), d_i is scalar. Assuming other distributions of the observations, the ML principle leads to other minimization problems than the least squares one: Instead of the sum of the squared normalized discrepancies, the sum of other functions $\tau(d_i)$ has to be minimized [Förstner, 1998]:

$$\sum_i \tau(d_i) \longrightarrow min \tag{4.13}$$

For instance, the assumption $\tau(d_i) = |d_i|$ yields the minimization of the sum of the absolute values of the normalized discrepancies, i.e. the *L_1-norm* which is already more robust with respect to gross errors in the data as can be seen by comparing the properties of the arithmetic mean (corresponding to the L_2-norm) and the median (corresponding to the L_1-norm) of a set of observations [Kraus, 1997]. Minimizing $\sum_i \tau(d_i)$ can

be performed using the method of iteratively modulating the weights $p_{i,k+1}$ in iteration $k+1$ depending on a *weight function* $w(d_{i,k})$ of the normalized discrepancies $d_{i,k}$ in iteration k [Förstner, 1998]:

$$p_{i,k+1} = p_i \cdot w(d_{i,k}) = p_i \cdot \frac{1}{d_{i,k}} \cdot \frac{\partial \tau(d_{i,k})}{\partial d_{i,k}} \tag{4.14}$$

In equation 4.14, the arguments of the weight function $w(d_{i,k})$ are the normalized discrepancies $d_{i,k}$ from equation 4.12. This is not the only possible choice. There are several possibilities for choosing the argument of the weight function, e.g. [Lang and Förstner, 1998]:

1. The corrections $\tilde{v}_i$: This is not a good choice for the reasons already described above.
2. The normalized corrections $\overline{v}_i$ (equation 4.19): In section 4.1.1.2, this entity will be shown to be an optimum test statistic for gross error detection. However, its estimation is computationally expensive because it involves the computation of the matrix of co-variances of the corrections.
3. The normalized discrepancies $d_{i,k}$ from equation 4.12: This seems to be a good trade-off between the need for normalization and the computational requirements.
4. Instead of $d_{i,k}$, the normalized discrepancies a priori $\overline{d}_{i,k+1} = \frac{\overline{l}_{i,k+1}}{\sigma_i}$ from iteration $k+1$, $\overline{l}_i$ from equation 4.3, are used. In this case, no information about iteration k is required, i.e., the corrections $\tilde{v}_{i,k}$ need not be stored. $\overline{d}_{i,k+1}$ differs from $d_{i,k}$ by the effects of linearization.

The weight functions $w(d_{i,k})$ should fulfil several requirements:

- An observation with $d_{i,k} = 0$ should receive its initial weight, thus $w(0) = 1$
- w should be monotonously decreasing
- The influence of gross errors on the results of adjustment should be reduced, thus $\lim_{d_{i,k} \to \infty} w(d_{i,k}) = 0$
- In order to completely eliminate observations marked as gross errors, $w(d_{i,k})$ should be cut off at a certain threshold t: $w(d_{i,k}) = 0$ for $|d_{i,k}| > t$.

A weight function fulfilling the first three of the above criteria is [Klein and Förstner, 1984]:

$$w(d_{i,k}) = \frac{1}{1 + (a \cdot |d_{i,k}|)^b} \tag{4.15}$$

with $a > 0$ and $b > 0$. It is shaped similarly to a bell curve, the steepness being determined by parameter b. Instead of a and b, more expressive parameters can be used: the size h of a normalized residual yielding $w(h) = 0.5$ and the inclination of the tangent at the position $w(h)$ represented by the intersection s of the abscissa and a line parallel to the tangent and passing the culmination point of the weight function (figure 4.1):

$$\begin{aligned} h &= \frac{1}{a} \\ s &= \frac{4}{a \cdot b} \end{aligned} \tag{4.16}$$

Using this parameterization, equation 4.14 can be re-formulated as

$$p_{i,k+1} = p_i \cdot \frac{1}{1 + \left(\frac{|d_{i,k}|}{h}\right)^{\frac{4 \cdot h}{s}}} \tag{4.17}$$

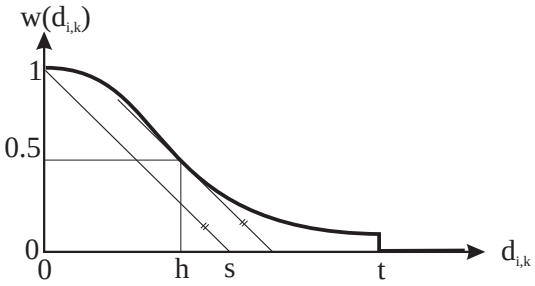


Figure 4.1: The weight function from equation 4.17 with parameters h and s and cut-off point t.

In practice, a default value for s is chosen as $s = h$. In this case, the form of the curve is completely determined by h. In addition, in the adjustment of uncorrelated observations, the linearized observation equations 4.3 are multiplied by $\sqrt{p_i}$ in order to achieve a "homogenization" of these equations. Adjustment is then performed using these homogenized observation equations which all can be considered to have the weight 1. In *ORIENT*, setting up the homogenized observation equations, the square roots of the weights are multiplied by the weight function $w(d_{i,k})$ rather than the weights themselves. Thus in *ORIENT*, the following re-weighting function is used [Kager, 1995]:

$$p_{i,k+1} = p_i \cdot \frac{1}{\left[1 + \left(\frac{d_{i,k}}{h}\right)^4\right]^2} \tag{4.18}$$

ORIENT offers the possibility to detect gross errors using the method of re-weighting the observations based on equation 4.18. From a practical point of view, the procedure starts with adjustment using the original weights p_i. As soon as convergence has been achieved, the iteration process will continue with the weight $p_{i,k+1}$ of observation i in adjustment $k + 1$ being modulated according to $w(d_{i,k})$ as described above. Observations fulfilling $d_{i,k} > t$ are marked as being suspected gross errors. Parameter h is the size of a normalized residual causing weight modulation to give an observation half its original influence, i.e., 25% of its original weight. If redundancy is great enough, blunders, i.e. observations not fitting to our mathematical model with the accuracy we expect, will successively lose influence by receiving a lower weight. However, an observation with a low influence according to equation 4.18 can be rehabilitated in the next iteration step if another one, this time the true blunder, has been eliminated in the meantime. Convergence speed depends on h; we usually select h a bit smaller than the greatest normalized residual which is not yet marked as an error (i.e. which is smaller than the initial value of t) and set the cut-off point $t = h$. For each selected value of h, adjustment is iteratively performed until convergence is achieved. After that, another (smaller) value of h is selected, and the process of adjustment is repeated with h being reduced at each step until h reaches a given threshold, e.g. $h_{min} = 3$. This threshold means that there is no observation left in adjustment with a residual greater than three times the r.m.s. error of the respective observation. By the strategy described above, for each value of h, initially only one observation is suspected to be a gross error, which causes a great number of iterations to be performed. If the number of outliers is expected to be high, the procedure can be sped up by suspecting n (e.g., $n = 5$) observations to be gross errors, i.e., by selecting h to be between the n^{th} and the $n + 1^{st}$ greatest normalized residual of the previous adjustment. In order to make this method of robust estimation work, a high redundancy is required, and the number of outliers should not exceed 30 %. As the method is based on iterative adjustment, approximate values for the unknowns are required. It might fail if the approximate values are too bad and in the presence of errors of a size preventing iteration from convergence.

In the context of [Leonardis and Bischof, 2000], ML type robust estimators are called *soft redescenders*: soft redescenders use a windowing technique for deciding which observations are to participate in adjustment and which are not (the window is given by the threshold t), but the iterative strategy based on re-weighting gives observations the chance to first lose influence before being eliminated and to come back after having been eliminated.

4.1.1.2 Data snooping

This method is based on a statistical test of the normalized residuals. It has been developed by Prof. Baarda from Delft [Förstner, 1978, Kraus, 1997]. In the section on robust estimation we used the normalized discrepancies $d_{i,k}$ (equation 4.12) for finding out possibly wrong observations. The concept of data snooping uses the normalized residuals a posteriori $\overline{v}_i$, i.e. the residuals $\tilde{v}_i$ divided by their r.m.s. error $\sigma_{\tilde{v}_i}$:

$$\overline{v}_i = \frac{\tilde{v}_i}{\sigma_{\tilde{v}_i}} = \frac{\tilde{v}_i}{\sigma_o \cdot \sqrt{q_{\tilde{v}_i\tilde{v}_i}}} \tag{4.19}$$

$q_{\tilde{v}_i\tilde{v}_i}$ is the i^{th} diagonal element of the co-factor matrix $\mathbf{Q}_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}}$ of the residuals. It can be computed by applying the laws of error propagation to equation 4.3, using the co-factor matrix of the unknown parameters $\mathbf{Q}_{\mathbf{xx}}$ from equation 4.9 and the co-factor matrix of the observations $\mathbf{Q}_{\mathbf{ll}}$ [Kraus, 1997]:

$$\mathbf{Q}_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}} = \mathbf{Q}_{\mathbf{ll}} - \mathbf{A} \cdot \mathbf{Q}_{\mathbf{xx}} \cdot \mathbf{A}^T \tag{4.20}$$

The normalized residuals $\overline{v}_i$ are normally distributed with expectation 0 and variance 1. Thus, a statistical test of the hypothesis $H_0 : \overline{v}_i \sim N(0,1)$ can be performed. It has been formally proven [Förstner, 1978] that in the presence of one single error, the corresponding observation will receive the largest normalized residual $\overline{v}_i$, which is not the case for the residuals and the normalized discrepancies. However, in real life projects, there will in general be more than one gross error in the data. That is why an iterative strategy is applied [Förstner, 1998, Kraus, 1997]:

1. Determine a global solution using all observations.
2. Test the normalized residuals.
3. Eliminate the observations which are most likely to be gross errors based on the normalized residual test. Stop if no such observations occur.
4. Determine the global solution omitting all rejected observations and go to step 2.

Computing the normalized residuals $\overline{v}_i$ is quite an effort because it implies the computation of the matrix $\mathbf{Q}_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}}$ (equation 4.20) which has the number of observations as its dimension. Especially in cases where the number of possibly false observations is rather large, this effort is prohibitive: the number of observations to be rejected at an instance should remain small, so that the matrix $\mathbf{Q}_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}}$ has to be re-computed several times in the iterative process described above. However, the concept is based on a sound theoretical background, and it can show errors in the data which remain undiscovered using the re-weighting scheme described in the previous section. Approximate values for the unknowns are required as the test statistics can only be computed after adjustment. Errors of a size that prevents convergence of the iteration process cannot be found by data snooping, either.

4.2 Co-ordinate systems and mapping functions

The hybrid photogrammetric adjustment system *ORIENT* has been developed at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology since the mid-seventies. It offers the possibility of simultaneous hybrid least squares adjustment of various types of observations (figure 4.2):

- perspective image co-ordinates
- image co-ordinates of line scanner data
- image co-ordinates of rotational scanner (satellite) data

- SAR image co-ordinates
- control points
- model co-ordinates
- geodetic measurements
- fictitious observations: points belonging to planes or to polynomial surfaces
- fictitious observations: points belonging to straight lines, circles, or to any intersecting curve of two polynomial surfaces
- fictitious observations: points belonging to 3D spline curves
- observed mapping parameters.

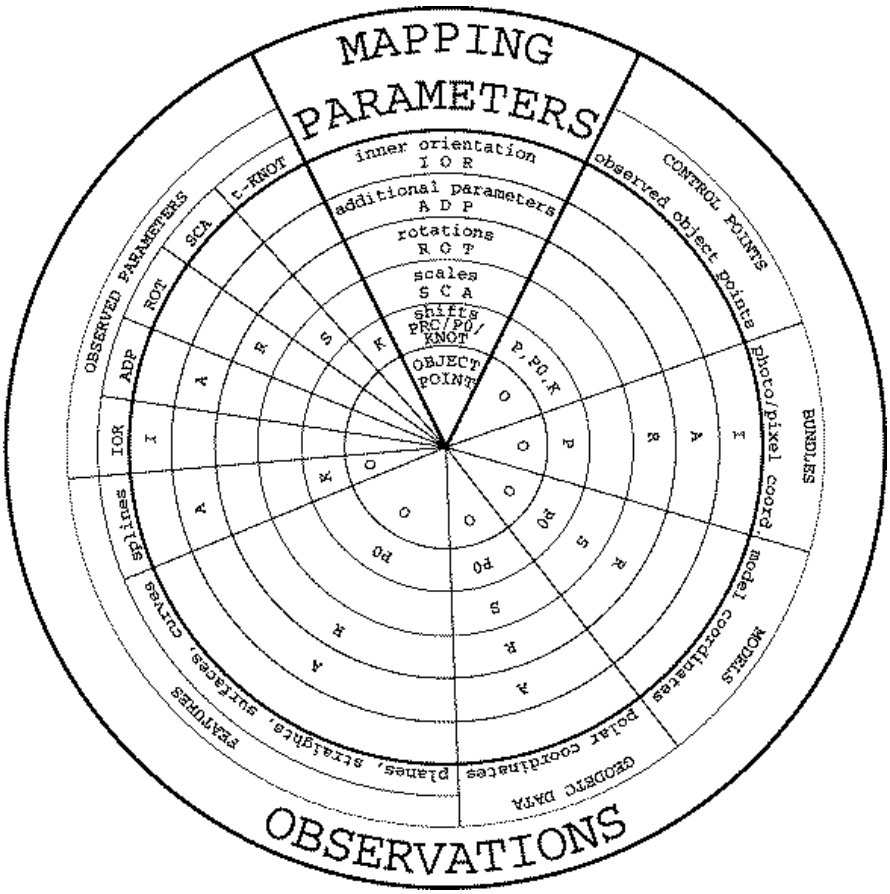


Figure 4.2: Observation and parameter classes in *ORIENT*.

Adjustment is based on the ML principle described in section 4.1. *ORIENT* assumes the observations to be normally distributed and uncorrelated, which means that the stochastic model given by equation 4.10 is applied. In addition, *ORIENT* offers two blunder detection techniques (cf. section 4.1.1; [Kager, 1989]):

- Robust estimation based on iterative re-weighting of observations.
- Data snooping.

The mathematical model of adjustment is based on a very strict concept of using all types of observations except the 3D splines uniformly by basically applying the same mapping function relating the observations to the (mapping and object) parameters. In the following sections we want to describe this mathematical model.

4.2.1 The mapping functions

It has been stated above that it is our goal to determine the parameters of an object indirectly from observations we can perform using some measurement device. Our observations are the co-ordinates of points which are observed in a co-ordinate system related to the measurement device. This co-ordinate system will hence be called the *observation co-ordinate system*. It is a 3D Cartesian system, the axes being denoted by (u, v, w). The object is to be described in another 3D Cartesian co-ordinate system called the *object co-ordinate system* or *reference system*. The observed point $\mathbf{p} = (u, v, w)^T$ is an image of an object point $\mathbf{P} = (X, Y, Z)^T$. The imaging process is mathematically described by a *mapping function* $\mathbf{T}(\mathbf{S}, \mathbf{P})$ containing the *mapping parameters* $\mathbf{S}$. There is one set of mapping parameters $\mathbf{S}_m$ attached to each observation co-ordinate system m. Images of the same point $\mathbf{P}_n$ can be measured in various observation co-ordinate systems m. Thus, the relation between the observed point $\mathbf{p}_{mn}$ and the object point $\mathbf{P}_n$ can be formally written as follows:

$$\mathbf{p}_{mn} = \begin{pmatrix} u_{mn} \\ v_{mn} \\ w_{mn} \end{pmatrix} = \begin{bmatrix} T_{u_m}(\mathbf{S}_m, \mathbf{P}_n) \\ T_{v_m}(\mathbf{S}_m, \mathbf{P}_n) \\ T_{w_m}(\mathbf{S}_m, \mathbf{P}_n) \end{bmatrix} = \mathbf{T}_m(\mathbf{S}_m, \mathbf{P}_n) \tag{4.21}$$

The functions $\mathbf{T}(\mathbf{S}, \mathbf{P})$ describe the mathematical model of adjustment in *ORIENT*. Both the number and the interpretation of the mapping parameters depend on the observation type. However, *ORIENT* treats all types of observations uniformly by using basically the same mapping function and the same categories of mapping parameters for all observation types except 3D splines. Leaving aside the obvious indices m and n, the basic formula relating the observed point $\mathbf{p}$ to the object point $\mathbf{P}$ is given by the spatial similarity transformation (figure 4.3) [Kraus, 1997]:

$$\mathbf{M} \cdot [\mathbf{p} - \mathbf{p}_0(\mathbf{adp})] = \lambda \cdot \mathbf{R}^T(\theta) \cdot (\mathbf{P} - \mathbf{P}_0) \tag{4.22}$$

with

- $\mathbf{M} = diag(m_u, m_v, m_w)$: a mirror matrix containing the mirror coefficients $m_i = \pm 1, i \in \{u, v, w\}$ for the u, v and w axis, respectively.
- $\mathbf{p} = (u, v, w)^T$: observed point.
- $\mathbf{p}_0 = (u_0, v_0, w_0)^T$: interior reference point.
- $\mathbf{adp}$: additional parameters modifying the interior reference point (e.g. camera distortion).
- λ: the scale factor between the observation and the object co-ordinate systems.
- $\mathbf{R}(\theta)$: a 3×3 rotational matrix which is computed from three rotational angles θ. $\mathbf{R}$ can be parameterized in several ways, the most common one being the typical parameterization of aerial photographs: $\theta = (\omega, \phi, \kappa)^T$. In this case, the coefficients of $\mathbf{R}$ are given by equation 4.23.
- $\mathbf{P} = (X, Y, Z)^T$: the object point.
- $\mathbf{P}_0 = (X_0, Y_0, Z_0)^T$: exterior reference point.

The coefficients r_{ij} of $\mathbf{R}$ can be computed from trigonometric functions of three rotational angles ω, ϕ, and κ [Kraus, 1993]:

$$\mathbf{R} = \begin{pmatrix} \cos\phi\cos\kappa & -\cos\phi\sin\kappa & \sin\phi \\ \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa & \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa & -\sin\omega\cos\phi \\ \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa & \sin\omega\cos\kappa - \cos\omega\sin\phi\sin\kappa & \cos\omega\cos\phi \end{pmatrix} \tag{4.23}$$

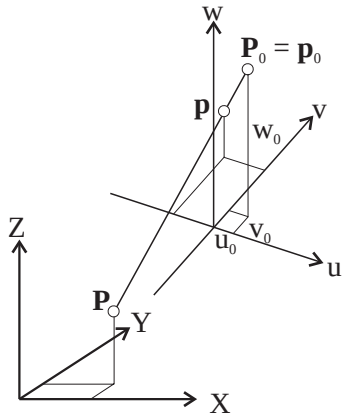


Figure 4.3: The spatial similarity transformation.

With the exception of $\mathbf{M}$ which describes a property of the observation co-ordinate system, all groups of parameters in equation 4.22 can be determined in an adjustment. Basically, these groups of parameters appear in the mapping functions of all observation types, but they might obtain different interpretations and/or be given constant default values. We want to emphasize that it is possible to

1. keep single groups of parameters fixed for each observation type,
2. declare several observation co-ordinate systems to share groups of mapping parameters (e.g., two perspectives may be declared to have the same rotational parameters if the photos were made using a stereo camera) without having to formulate condition equations, just by manipulating the data base (section 4.3),
3. declare groups of parameters constant for individual observation co-ordinate systems.

In the following sections, we want to describe the specific mapping functions for three types of observations which are of special interest in the context of semi-automatic building extraction.

4.2.2 Perspective observations (Photos)

For observed photo points, the mapping function is a perspective transformation. The observation co-ordinate system is the camera co-ordinate system, and the third co-ordinate of the observed point $\mathbf{p}$ is 0. $\mathbf{P}_0$ is the projection centre in the object co-ordinate system, its camera co-ordinates being $\mathbf{p}_0 = (u_0, v_0, f)^T$, where f is the principal distance (the camera constant). In case the photograph is geometrically positive, $\mathbf{M}$ can be assumed to be the identity matrix. The scale λ describes the location of $\mathbf{P}$ on the projection ray and thus is no longer constant for all points. Without considering the additional parameters adp, by dividing the first two lines of equation 4.22 by the third one and by doing some re-arrangement, we obtain the well-known equations for central projection. One observed image point yields two observation equations 4.1, one for u and one for v [Kraus, 1997]:

$$\begin{aligned} E(u) &= u + \tilde{v}_u = u_0 - f \cdot \frac{r_{11} \cdot (X - X_0) + r_{21} \cdot (Y - Y_0) + r_{31} \cdot (Z - Z_0)}{r_{13} \cdot (X - X_0) + r_{23} \cdot (Y - Y_0) + r_{33} \cdot (Z - Z_0)} \\ E(v) &= v + \tilde{v}_v = v_0 - f \cdot \frac{r_{12} \cdot (X - X_0) + r_{22} \cdot (Y - Y_0) + r_{32} \cdot (Z - Z_0)}{r_{13} \cdot (X - X_0) + r_{23} \cdot (Y - Y_0) + r_{33} \cdot (Z - Z_0)} \end{aligned} \tag{4.24}$$

The additional parameters adp describe lens distortion. The influence of camera distortion is modelled by polynomials describing small variations of the principal point of autocollimation $(u_{pp}, v_{pp})^T$.

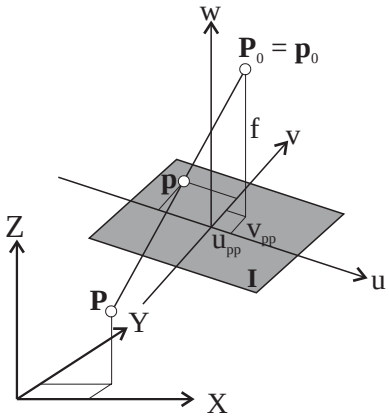


Figure 4.4: Perspective transformation.

They are considered by replacing u_0 and v_0 in equation 4.24 by

$$\begin{aligned} u_0 &= u_{pp} + du_0(adp, u', v') = u_{pp} + \sum_i a_i \cdot du_{0i}(u', v') \\ v_0 &= v_{pp} + dv_0(adp, u', v') = v_{pp} + \sum_i a_i \cdot dv_{0i}(u', v') \end{aligned} \tag{4.25}$$

where $u' = \frac{u-u_{pp}}{\rho_0}$ and $v' = \frac{v-v_{pp}}{\rho_0}$ are normalized camera co-ordinates and ρ_0 is a normalization radius. This means that camera distortion is described by the sum of (mostly polynomial) functions du_{0i} and dv_{0i} of the reduced image co-ordinates. For each index i there is such a couple of functions, and a_i is the corresponding distortion parameter. In *ORIENT* it is possible to choose a subset of all possible parameters a_i to describe the distortion of a specific camera. Table 4.1 gives the most important parameters a_i together with the corresponding terms du_{0i} and dv_{0i} and a geometrical interpretation.

i	$du_{0i}(u', v')$	$dv_{0i}(u', v')$	geometric meaning
1	0	u'	affinity - skewness of axes
2	0	v'	affinity - scaling of y-axis
3	$u' \cdot (r^2 - 1)$	$v' \cdot (r^2 - 1)$	radial distortion; 3^{rd} degree
4	$u' \cdot (r^4 - 1)$	$v' \cdot (r^4 - 1)$	radial distortion; 5^{th} degree
5	$r^2 + 2 \cdot u'^2$	$2 \cdot u' \cdot v'$	tangential (asymmetric) distortion
6	$2 \cdot u' \cdot v'$	$r^2 + 2 \cdot v'^2$	tangential (asymmetric) distortion
24	1	0	u-shift of principal point: PPA $\longrightarrow$ PPBS
25	0	1	v-shift of principal point: PPA $\longrightarrow$ PPBS

Table 4.1: Table of distortion parameters in *ORIENT* [Kager, 1995]. $r^2 = u'^2 + v'^2$

With respect to perspective images, the observation co-ordinate system is often referred to as *camera co-ordinate system*. It is the co-ordinate system the parameters of inner orientation ($\mathbf{p}_0, adp$) refer to. If digital images are used, there is another co-ordinate system attached to the images: the *sensor co-ordinate system* (figure 4.5).

Its origin is situated in the centre of the left upper pixel, and the axes (r, c) point in the direction of the image rows and columns, respectively. The units of the sensor co-ordinate system are [pixels]; thus, the sensor co-ordinates (r, c) of a point P can be interpreted as its row and column indices, respectively.

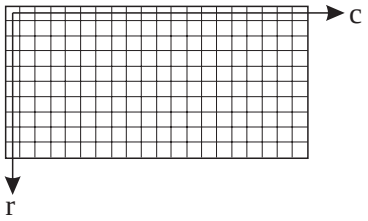


Figure 4.5: The sensor co-ordinate system.

We have not yet considered the relation between the sensor co-ordinate system (r, c) (figure 4.5) and the image co-ordinate system (u, v). For video cameras, camera mounted CCD sensors and remote sensing images, these co-ordinate systems can be assumed to be identical, thus $(u, v) = (r, c)$. This is no longer true for photographs from analogue metric cameras which were scanned off-line. Metric cameras have fiducial marks, i.e. small targets on the camera body which are imaged in the photographs. In this case, the image co-ordinate system is defined by these fiducial marks; their image co-ordinates are provided by the camera manufacturer in a *calibration protocol*. The form as well as the distribution of fiducial marks depend on the camera manufacturer.

If the sensor co-ordinate system and the image co-ordinate system are not identical, the relation between them can be described by an affine transformation $\mathbf{T_a}$ [Kraus, 1993]:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{T_a}(r, c) = \begin{pmatrix} c_{00} \\ c_{01} \end{pmatrix} + \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \cdot \begin{pmatrix} r \\ c \end{pmatrix} \tag{4.26}$$

When points are digitized in digital images or when they are measured automatically using feature extraction or matching techniques (cf. chapter 5), the sensor co-ordinates of the points are determined rather then the camera co-ordinates. In the case of automatic procedures, these co-ordinates themselves are the results of an adjustment procedure and, thus, correlated. Even in case they were uncorrelated, the variance-covariance matrix of the observation co-ordinates would contain off-diagonal elements due to error propagation applied to equation 4.26. However, in *ORIENT* these correlations are not considered.

4.2.3 Surface observations

Using perspective observations as described in section 4.2.2, object points can be reconstructed. However, this means that all object points have to be measured in at least two images (cf. section 4.5). Sometimes this is not possible due to occlusions, or it is not desirable for efficiency reasons. In such cases, points can be declared to be contained in surfaces or curves in object space. Thus, additional observations for these object space features can be included in adjustment. In this section we want to describe how this can be accomplished using *ORIENT*. The concept of surface observations (*GESTALTS*) has been described in [Kager, 1980] and in [Kager, 1989].

Surfaces, too, are described in a local observation co-ordinate system. Again, the transformation formula is given by equation 4.22 with the additional assumption $\lambda = 1$. In this section, we will use the short-hand $\mathbf{p}_R = (u_R, v_R, w_R)^T = \mathbf{R}^T \cdot (\mathbf{P} - \mathbf{P}_0)$ for the right-hand side in equation 4.22. The observation, "a point $\mathbf{P}$ is on a surface" can be expressed as "$\mathbf{P}$'s distance from that surface is observed to be 0". For reasons of simplicity, we do not use the Euclidean distance but its projection to one of the axes of the observation co-ordinate system. The interior reference point $\mathbf{p}_0$ receives a special interpretation: we consider one of its components to be a polynomial function of degree n of the other two components of $\mathbf{p}_R$, and these polynomials provide the equation of that surface in the observation co-ordinate system, e.g. $w_0 = w_0(adp, u_R, v_R)$. Similar considerations can be made for u and v, so that the observation equations for a point $\mathbf{P}$ being contained in a surface can be formulated in one of the following three ways:

$$
\begin{aligned}
E(u) \quad &= 0 + \tilde{v}_u \quad = m_u \cdot u_R + \sum_{j=0}^{n} \sum_{k=0}^{n-j} a_{jk} \cdot (m_v \cdot v_R)^j \cdot (m_w \cdot w_R)^k \\
E(v) \quad &= 0 + \tilde{v}_v \quad = m_v \cdot v_R + \sum_{i=0}^{n} \sum_{k=0}^{n-i} b_{ik} \cdot (m_u \cdot u_R)^i \cdot (m_w \cdot w_R)^k \\
E(w) \quad &= 0 + \tilde{v}_w \quad = m_w \cdot w_R + \sum_{i=0}^{n} \sum_{j=0}^{n-i} c_{ij} \cdot (m_u \cdot u_R)^i \cdot (m_v \cdot v_R)^j
\end{aligned}
\tag{4.27}
$$

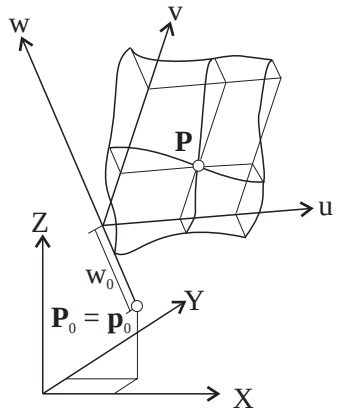


Figure 4.6: Surface observations.

Figure 4.6 shows an example of a polynomial surface described by the third equation 4.27. w_0 is varied depending on u_R, v_R so that the point $\mathbf{p}$ will always be in the plane $w = 0$. The additional parameters adp from equation 4.22 are replaced by the set of polynomial coefficients a_{jk}, b_{ik} and c_{ij} which describe the surface in the observation co-ordinate system [Kraus, 1997]. An application is free to decide which of the coefficients are to be used for a certain surface: the list of additional parameters may consist of any subset of these coefficients, the others being assumed to be 0. The coefficients (m_u, m_v, m_w) of the mirror matrix $\mathbf{M}$ are important for modelling symmetries: these coefficients can take two values: $m_q \in \{-1, 1\}$ for $q \in \{u, v, w\}$. By assigning identical surface coefficients and transformation parameters $(\mathbf{P}_0, \theta)$, but different values of m_q to different surfaces, symmetries with respect to the co-ordinate plane(s) orthogonal to co-ordinate q of the observation co-ordinate system can be modelled.

It is one of the benefits of this way of mathematical formulation that geometrical constraints between surfaces can be modelled by it: We have already discussed symmetries between surfaces. By assigning identical coefficients and identical but unknown rotations θ to a set of surfaces, parallelism can be enforced, and rectangularity of two planes can (e.g.) be obtained by formulating them as being the $uv-$ and the vw-planes of the same co-ordinate system, respectively. Care has to be taken with respect to the determinability of parameters. For instance, by a surface, only one of the co-ordinates of $\mathbf{P}_0$ can be determined, whereas the others either have to be determined by other observations or kept fixed. In addition, this co-ordinate is dependent on the constant term, thus, in adjustment, either the constant term of the surface equation or $\mathbf{P}_0$ can be determined. With respect to the rotations, a similar statement holds true: by a surface, only two angles can be determined, and these angles are dependent on the linear terms of the surface equation. However, it is possible to build a system of surfaces, all being assigned identical rotations and exterior reference points, but having different surface parameters. By such a system of surfaces, it is possible to determine, for instance, all co-ordinates of the exterior reference point and all rotational angles.

3D polynomial curves can be formulated as the intersection of two surfaces by using a set of two equations 4.27, and a 3D point is determined by the intersection of three surfaces, i.e. by all three equations 4.27. A point on a

curve gives support to the parameter sets of both surfaces. As the attachment of parameter sets to observations is done by reference, one set of surface parameters can be used for more than one curve. It is an important property of this mathematical model that no homologous image points are required to determine the parameters of a curve: an image point gives two observation equations 4.24. If this point is assigned to a curve, i.e. two surfaces, two equations 4.27 are obtained. An image point assigned to an object line gives four observation equations, but only three new unknowns (the point co-ordinates), thus one observation is redundant. The curve (i.e. surface) parameters are determined from these "redundant" observations, by intersection of bundles of rays from different images rather than from homologous points. However, care has to be taken on the determinability of the coefficients of the intersecting surfaces. For example, thinking of a straight line being the intersection of two planes, the tilts of the planes orthogonal to the line either have to be determined by other observations or have to be declared constant.

We have already stated above that by the mathematical model given by equations 4.27, the projection of the distance of $\mathbf{P}$ from the surface to one of the co-ordinate axes is minimized rather than the distance itself. This is not critical if the surface normal is approximately parallel to the co-ordinate axis, but if this is not the case, some algebraic value is minimized instead of the metric distance. In [Kager, 2000], the principle of surface observations is expanded to actually minimizing the metric distance between $\mathbf{P}$ and a surface given by an implicit equation. The convergence behaviour of these implicit surfaces has not yet been examined.

4.2.4 Observed parameters

The parameters of the mapping function 4.22 can usually not be observed directly, the exception being the object co-ordinates $\mathbf{P}$ which are considered to be observed as control points. However, if a parameter might not be determinable from other observations, an observation for that parameter might be useful to avoid singularities. With respect to observed object co-ordinates, equation 4.22 degenerates to

$$\begin{aligned} E(u) &= u + \tilde{v}_u &&= X \\ E(v) &= v + \tilde{v}_v &&= Y \\ E(w) &= w + \tilde{v}_w &&= Z \end{aligned} \tag{4.28}$$

assuming the parameter co-ordinate system (the object co-ordinate system in the case of control points) to be identical to the observation co-ordinate system (figure 4.7). With respect to observed rotational angles $\theta = (\omega, \phi, \kappa)^T$, we get similar equations, the difference being that parameter space is not identical to the object co-ordinate system $(X, Y, Z)^T$, but by the space of rotational angles, i.e., X in equation 4.28 is replaced by ω, Y by ϕ and Z by κ. Similar considerations can be made with respect to all groups of parameters, thus, there are not only observed rotation angles and observed control point co-ordinates, but also observed surface coefficients, etc. Depending on the parameter type and on specifications by the user, only one or two lines of equation 4.28 might be used in adjustment.

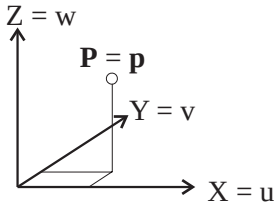


Figure 4.7: Example for observed parameters: observed object co-ordinates.

Apart from very few special cases, the parameters cannot be observed directly. For instance, for a metric camera which is levelled so that its viewing axis is horizontal, one of the rotational angles can be assumed to be observed, the standard deviation of that observation corresponding to the accuracy of the level. Other "observed" parameters are often derived from other groups of observations. For instance, control point co-ordinates are usually obtained from geodetic or GPS measurements and, thus, correlated. Again, these correlations are omitted by our stochastic model.

4.3 Data structure: the ORIENT data base

Having described the mathematical model of hybrid photogrammetric adjustment used by *ORIENT* in the previous sections, we now want to give an introduction into *ORIENT*'s data structure because we consider it essential for understanding the possibilities offered by that program. The contents of this section are largely based on [Kager, 1995] and [Stadler, 1997].

4.3.1 Basic data structures: rooms and points

All data in *ORIENT* are stored in the *ORIENT data base*. The basic entity of storage within that data base is called a "*room*". All data are stored in "rooms". The term "room" can be literally understood to be a 3D space with a 3D Cartesian co-ordinate system attached to it[1].

ORIENT distinguishes two basic groups of room types, i.e. *observation rooms* and *parameter rooms*. In the first group of rooms, the observations are stored, and the co-ordinate system assigned to the room is one of the observation co-ordinate systems (cf. section 4.2). On the other hand, parameter rooms contain the unknowns to be determined in adjustment, and the co-ordinate system is a parameter co-ordinate system (cf. section 4.2.4).

Each room consists of (table 4.2):

1. a *room type*: In the case of a parameter room, the room type declares the parameter type contained in the room, whereas for observation rooms, it describes which mapping function has to be used for the points contained in that room.

2. an *identifier*. A room is uniquely defined by its type and identifier: there may be several rooms having the same identifier, but different room types (typically, the rotation angles attached to a photo and the photo itself share the same identifier).

3. a *header* containing *references* (not the data themselves!) to the parameters of the mapping function and a sub-type to encode properties of the co-ordinate system attached to the room.

4. a *point list*: all data are stored as points contained in one of the (parameter or observation) rooms. Each point consists of a unique identifier, its co-ordinates with their respective r.m.s. errors and a status word used to distinguish points taking part in adjustment ("active points") from points which should not to take part ("inactive points"). There is a difference between points in parameter rooms and points in observation rooms: only the latter ones have r.m.s. errors attached to their co-ordinates. Finally, depending on the room type, the number of co-ordinates per point may vary: surface observation points have no co-ordinate at all (as their ficticiously measured distance from the surface is 0), photo points have two co-ordinates, and control points have three.

[1] The German word "Raum" can be translated to English both as "room" and "space". Although in our context, "space" would be a better translation, the term "room" will be used throughout this section because it is used in all *ORIENT* documentations.

Room		
Member	Type	Description
identifier	int	room number
roomType	int	room type
roomHeader	hdrTyp	A description of the parameters of the mapping function
points	pointList	the list of points contained in the room

Table 4.2: A room in the *ORIENT* data base

Data structure and adjustment: Supported by the data structure described above, with respect to adjustment, the user is offered a great freedom to specify which observations should take part in adjustment and which parameters are to be determined:

- The user can specify which observation types should take part in adjustment.
- For each observation type specified, the user can select
  - a list of room identifiers to further select observations.
  - which groups of parameters of the type specific mapping function are to be kept constant and which groups are to be determined as unknowns.
- In order to exclude single points from adjustment, they can be de-activated in the respective observation room. For points in observation rooms, single co-ordinates can be de-activated, too. Thus it is, for instance, possible to introduce height control points into adjustment by inserting the control point into an observation room of type "control point" and de-activating its planimetric co-ordinates.
- In order to keep single parameters constant even though the group of parameters has been specified to be unknown, the respective point can be de-activated in its parameter room. Inactive parameters rooms will be used for the evaluation of the mapping functions, but they will not be determined in adjustment. Thus it is, for instance, possible to simultaneously adjust photo observations from metric and non-metric cameras, keeping fixed the inner orientation parameters of the first ones and determining those of the second ones on-the-job.
- However, it is not possible to de-activate single co-ordinates of points in parameter rooms. This may cause singularities which can be avoided by introducing observations for the singular parameters.

In section 4.3.2, we will have a closer look on how the parameter groups of the basic mapping function (equation 4.22) are stored in the data base. Section 4.3.3 is dedicated to explain details about the header information of observation types. Finally, section 4.3.4 deals with some aspects of implementation of the *ORIENT* data structure.

4.3.2 Parameter rooms

All the parameters which can be determined in adjustment are contained in the *ORIENT* data base as co-ordinates of points in parameter rooms. Let us see what the data structure looks like for all groups of parameters.

Object points $\mathbf{P}$ and exterior reference points $\mathbf{P}_0$: Both the object points $\mathbf{P}$ and the exterior reference points $\mathbf{P}_0$ are stored in a parameter room of type "*reference system*". In our application, there is only one room of that type available in the data base, and it is attached to the object co-ordinate system (multiple reference systems could, for instance, be used to separate different temporal epochs in deformation analysis). Note that

ORIENT does not distinguish different classes of points in the object co-ordinate system: the exterior reference points are distinguished from the mere object points just by the fact that they are referenced in the headers of one or more observation rooms.

Rotational angles θ: The rotational angles are contained in rooms of type "*rotation parameters*". Each room of that type contains exactly one point having three co-ordinates, i.e., the three rotation angles (exception: rotational or line scanner images). The sub-type of the room describes how the rotation angles are to be interpreted, i.e., which parameterization is used for the rotation angles. Currently, eight types of parameterizations are available in *ORIENT*, the most common one being the typical parameterization of aerial photographs: $\theta = (\omega, \phi, \kappa)^T$.

Scale λ: The scales are contained in rooms of type "*scale*". Each room of that type contains exactly one point having one co-ordinate, i.e., the scale.

Interior reference point $\mathbf{p}_0$: We have seen in section 4.2 that the interior reference point can take various interpretations. In many cases it is just a vector containing only zeroes, in other cases (e.g. for surface observations, cf. section 4.2.3), it is not a parameter on its own but rather replaced by functions depending on other parameters which describe a variable shift of the observation co-ordinate system. Especially for photos it has a specific meaning: the parameters of inner orientation of photos are stored in rooms of type "*inner orientation*". Each room of that type contains exactly one point having three co-ordinates (u_{pp}, v_{pp}, f), i.e., the co-ordinates of the principal point (u_{pp}, v_{pp}) and the principal distance f. If no distortion parameters are available, this point corresponds to $\mathbf{p}_0$.

Additional parameters $\mathbf{adp}$: Additional parameters may yield different interpretations as they are parameters of quite different functions, for instance the functions describing camera distortion (cf. section 4.2.2) and those describing surfaces (cf. section 4.2.3). Still, all these parameters are contained in rooms of type "*additional parameters*". Each room of that type contains a list of points, each point representing one additional parameter, thus having one co-ordinate. The interpretation of the additional parameters depends on the identifiers of the points contained in the room:

- Additional parameters for perspective photos: The point identifier is identical to the index i in table 4.1. This means that only those terms of the distortion polynomial (equation 4.25) are evaluated for which a point with identifier i is found in the corresponding additional parameter room, and only these parameters can be determined in adjustment. By inserting/removing points in these rooms, the mathematical model of camera distortion can be modified. The normalization radius ρ_0 for camera distortion (equation 4.25) is stored in the header of the additional parameter room.
- Additional parameters for surfaces: The point identifier encodes the coefficient indices in equations 4.27. The point numbers take the form `TIJK` where `T` indicates the type of equation (`T` = 1: the point is one of the coefficients a_{jk}, i is considered to be 0; `T` = 2: the point is one of the coefficients b_{ik}, j is considered to be 0; `T` = 3: the point is one of the coefficients c_{ij}, k is considered to be 0) whereas `IJK` encode i,j, and k, respectively. For instance, point number 3000 corresponds to coefficient c_{00}, i.e. the constant term of a z-equation. Again, only points contained in the additional parameter room are evaluated, and only these coefficients can be determined in adjustment. By inserting/removing points in such a room, the surface equation can be modified.

Mirror matrix $\mathbf{M}$: It has already been stated in section 4.2.1 that the mirror matrix describes a property of the observation co-ordinate systems. Thus, the coefficients of the mirror matrix are constants in adjustment, but they are encoded in the sub-type of the observation rooms.

4.3.3 Observation rooms

All the observations which can take part in adjustment are contained in the *ORIENT* data base as co-ordinates of points in observation rooms. As stated in section 4.3.1, each observation room contains references to its type specific mapping parameters in its room header. As observation rooms contain references to mapping parameters rather than the parameters themselves, it is easily possible to assign the same parameters to different rooms, thus yielding, e.g., two observation co-ordinate systems to be parallel or a set of photos to have the same inner orientation. This feature will be especially important when we will formulate models of buildings using the concept of surface observations. In this section we want to describe the contents of the headers of the observation types that are of special interest for us.

Perspective photos: Perspective photos are contained in rooms of type "*photo*". Each point in a photo has two co-ordinates and two r.m.s. errors, one for each co-ordinate. As we have seen in section 4.2.2, the scale parameter is not relevant for photos. The header of a photo thus contains the items as follows:

- *SUBTYPE*: In the sub-type of a photo, the coefficients of the mirror matrix $\mathbf{M}$ are encoded.
- *ERP*: This item is the identifier of the exterior reference point $\mathbf{P_0}$, i.e., the projection centre of the photo. This point has to be available in the reference system.
- *ROT*: The identifier of a room of type "*rotation parameters*" containing the photo's rotation angles θ.
- *IOR*: The identifier of a room of type "*inner orientation*" containing the photo's interior reference point $\mathbf{p}_0$ (the principal point and the focal length).
- *ADP*: The identifier of a room of type "*additional parameters*" containing a subset of the distortion parameters of the complete version of table 4.1 [Kager, 1995].

For each active point contained in a photo room, two observation equations 4.24 will be introduced into adjustment.

Surface observations: Perspective photos are contained in rooms of type "*GESTALT*"[2]. Points in a *GESTALT* room have no co-ordinates at all because the actual observation is the zero distance to the surface. The header of a *GESTALT* room contains the items as follows:

- *SUBTYPE*: In the sub-type of a *GESTALT*, the coefficients of the mirror matrix $\mathbf{M}$ are encoded.
- *ERP*: This item is the identifier of the exterior reference point $\mathbf{P_0}$ of the *GESTALT*. This point has to be available in the reference system. Note that an application has to take care on whether that point can be determined in adjustment because it is dependent from the constant terms of the surface equations. If an application decides to determine these terms rather than $\mathbf{P_0}$, $\mathbf{P_0}$ should be set constant and thus de-activated in the reference system.
- *ROT*: The identifier of a room of type "*rotation parameters*" containing the *GESTALT*'s rotation angles θ. Note that an application has to take care on whether the rotations can be determined in adjustment because they might be dependent from the linear terms of the surface equations. If an application decides to determine these linear terms rather than θ, the rotations should be de-activated. If only one or two angles should be determined and the others kept fixed, observations for the angles to be fixed have to be introduced into adjustment.

[2]The German word "Gestalt" means something like "shape".

- *ADPU*: The identifier of a room of type "*additional parameters*" containing a subset of the surface parameters a_{jk} in equations 4.27. If *ADPU* is different from zero, an equation for u (first line in equation 4.27) will be introduced into adjustment for each point contained in the *GESTALT*, otherwise no u-equation will be introduced.

- *ADPV*: The identifier of a room of type "*additional parameters*" containing a subset of the surface parameters b_{ik} in equations 4.27. If *ADPV* is different from zero, an equation for v (second line in equation 4.27) will be introduced into adjustment for each point contained in the *GESTALT*, otherwise no v-equation will be introduced.

- *ADPW*: The identifier of a room of type "*additional parameters*" containing a subset of the surface parameters c_{ij} in equations 4.27. If *ADPW* is different from zero, an equation for w (third line in equation 4.27) will be introduced into adjustment for each point contained in the *GESTALT*, otherwise no w-equation will be introduced.

For each active point contained in a *GESTALT* room, between one and three observation equations 4.27 will be introduced into adjustment, depending on which of the header items *ADPU*, *ADPV* and *ADPW* are non-zero.

Observed parameters: For each parameter room type there exists an observation room type, the points in such an observation room having the same number of co-ordinates as the corresponding parameter room. It is possible to specify just one or two of the co-ordinates to be observed. The most important group of observed parameters are the observed object points, i.e. the control points which are stored in rooms of type "*control point*". In addition, there are rooms of type "*observed rotations*", "*observed scale*", "*observed inner orientation*" and "*observed additional parameters*". These observations are often used to prevent singularities.

4.3.4 Implementation aspects

ORIENT is written in *FORTRAN*. The core of the system is provided by the *ORIENT* data base interface, the implementation of the mapping functions (including the derivatives of the mapping functions) and the normal equation solution using a sparse matrix technique [Gsandtner and Kager, 1988]. In order to make the program extensible more easily and in order to enable application programs to directly use *ORIENT* methods, the system is currently partly re-implemented. Application programs can now directly access the *ORIENT* data base via an object oriented interface in C++ (figure 4.8). The *FORTRAN ORIENT* data base interface consists of a set of subroutines responsible for creating, manipulating and deleting rooms and points in rooms, and it also contains methods for storing the data to and retrieving them from disk. A low-level C++ wrapper for these *FORTRAN* routines was created which provides object oriented access to the data base. The core of this interface is a C++ class corresponding to a room in the data base and another C++ class being responsible for room management. Access and manipulation of points is transferred to the room class. These classes still access the *FORTRAN* subroutines.

Especially for visualization purposes, but also for re-implementing adjustment, an object oriented interface for the mapping functions was created [Stadler, 1997]. Originally, this was done in *EIFFEL* [Meyer, 1990], but due to problems encountered in mixing programming languages, a C++ interface for the mapping functions was created, too. This interface is based on a class hierarchy: the spatial similarity transformation (equation 4.3) is wrapped up in a base class for an inheritance tree. All the other mapping functions are derived from that base class, only the actual transformation methods being partly re-implemented as polymorphic methods. As the mapping functions obtain their data from the *ORIENT* data base (especially the header information), the mapping function classes make use of the low-level C++ interface to the *ORIENT* data base (figure 4.9).

The *ORIENT* data base only contains the vectorized data. For instance, there is no possibility to attach digital images to *ORIENT* photo rooms. Considering *ORIENT* as an adjustment server for application programs, this is not necessary, but applications of *ORIENT* such as the program *ORPHEUS* for digital photogrammetry

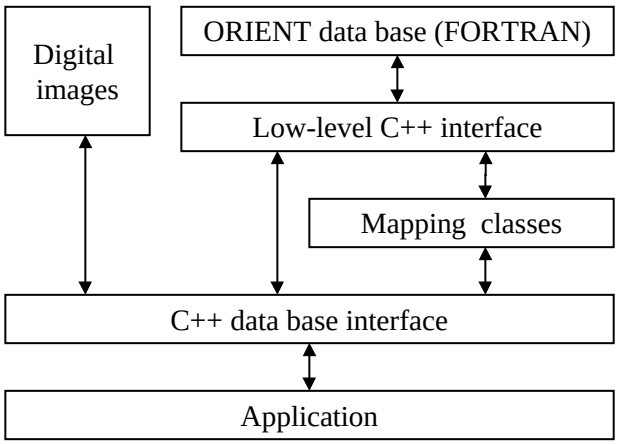


Figure 4.8: The *ORIENT* data base interface.

(cf. section 4.4) and the automation tools based on digital image processing which will be described in part III need this feature. That is why a C++ data base interface was created which does not only give access to the data themselves via the low-level data base interface, but also to digital images and to the mapping functions. This is achieved by a class "*rasterRoom*" derived from the low level class *room* which offers access to the mapping function of the room and can also contain a digital image pyramid (figure 4.9). This C++ data base interface is used by all applications of *ORIENT* which are currently developed at the Institute of Photogrammetry and Remote Sensing.

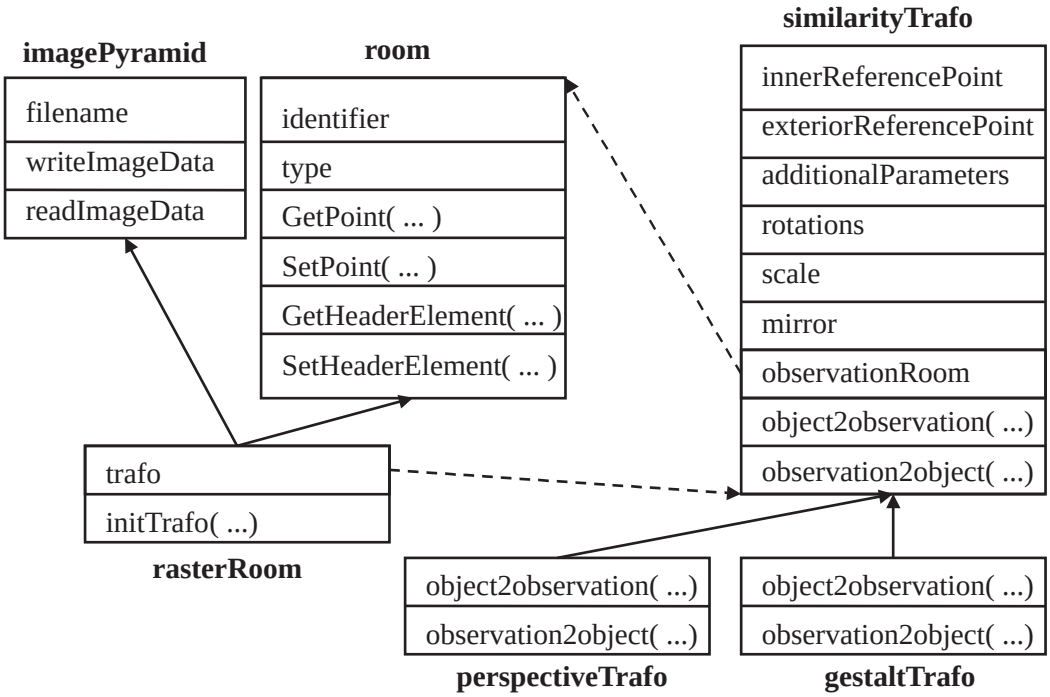


Figure 4.9: The relations of the room classes and the mapping function classes. Full arrows: inheritance relations. Dashed arrows: containment of references. If the reference is one to an instance of a base class, it may also point to an instance of a derived class.

4.4 ORPHEUS

The program *ORPHEUS*[3] was developed at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology. Basically providing a graphical user interface for *ORIENT*, it additionally offers new modules for interactive measurement of points and/or lines in digital images (figure 4.10). Thus, *ORPHEUS* is a *digital multi-image monocomparator*. Moreover, *ORPHEUS* is the development environment of the Institute of Photogrammetry and Remote Sensing for automatic tools using digital image processing techniques. That is why the algorithms described in this work were implemented and tested in *ORPHEUS* which was especially used for data management and the visualization of results. *ORPHEUS* offers modules for:

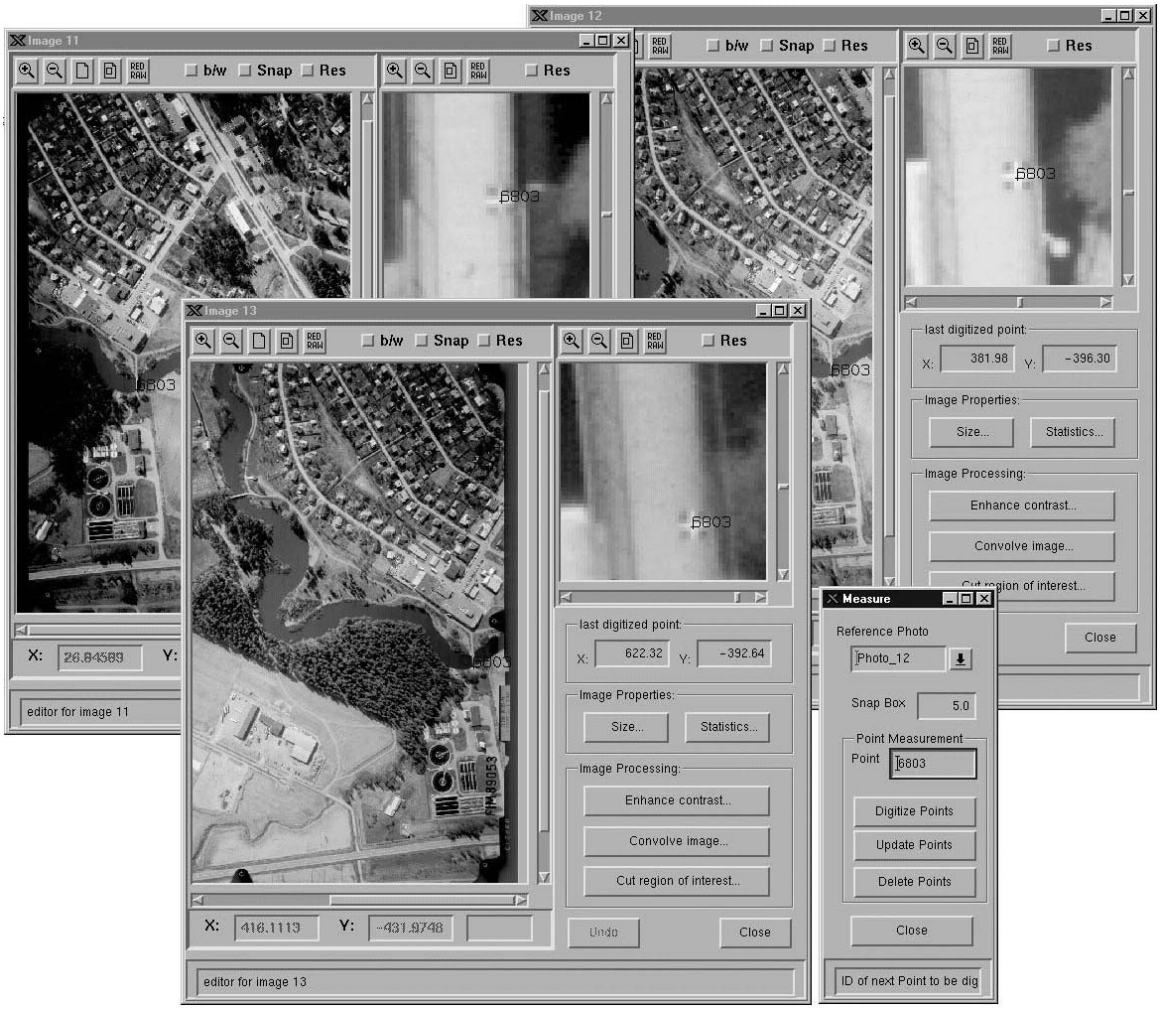


Figure 4.10: An example for interactive measurement using *ORPHEUS*.

- Observation management: up to now, 3D models, control points, perspective image (photo) points and surface observations can be managed using *ORPHEUS*. Both digital images and vector data digitized off-line can be imported.

- Graphical display of digital images: several digital images can be displayed simultaneously on the computer screen, their number being restricted only by system limits and the screen extents. As image pyramids (cf. section 5.4.1) are used, the display of very large (e.g. scanned aerial) images is quite fast. Data already digitized in the images are displayed on the screen, too. In addition to an overview of the

[3] *ORPHEUS* is an acronym for *OR*IENT: a *ph*otogrammetric *e*ngineering *u*tility *s*ystem.

digital image, a zoom window is provided (figure 4.10). In the zoom window, bilinear resampling for the display of enlarged image parts can be applied.

- Digital image processing: *ORPHEUS* offers tools for contrast enhancement, filtering (also information preserving filters). These tools can be applied to the whole digital images or just to the zoom window.
- Interactive measurement of points/lines in digital images: it is possible to measure points in an (in principle; see above) arbitrary number of images which can be displayed simultaneously.
- Interactive determination of approximate values for adjustment.
- Adjustment: A graphical user interface for adjustment by least squares is available.
- Data Export: A simple 3D-DXF and VRML export is provided.
- Production of 3D photo models in the VRML format based on interactive measurement.

The architecture of *ORPHEUS* can be seen in figure 4.11. *ORPHEUS* is one of the examples of the application framework *XX* developed at the Institute of Photogrammetry and Remote Sensing [Molnar et al., 1996]. This framework was mainly developed to provide a platform-independent graphical user interface for the software developed at that institute. In addition, it was designed in a way to integrate old *FORTRAN* modules which cannot be re-written with reasonable effort. It is the main idea of the framework to separate the user interface *UX* from the application programs. The module *SX* is responsible for message passing between the user interface and the application programs which are called *agents*. All these programs have to be written in C++ in object oriented design. In the current version, message passing is performed based on a *CORBA* interface [Redlich, 1996] (full arrows in figure 4.11). Via these messages, for instance, the agent programs are informed about user events by *UX*, and the results of computations by the agent programs can be sent to the user interface (again to *UX*) so that they can be inspected by the user. Note that great amounts of data, e.g. digital images, cannot be sent by these messages. In this case, the data are usually written to hard disk (dashed arrows in figure 4.11), and just the meta data (e.g. the file names) are transferred by messages.

The *agents* are application specific programs. They may interact with *FORTRAN* servers, i.e. existing programs written in *FORTRAN* which have to be slightly modified in order to render possible communication via *CORBA*. In the case of *ORPHEUS*, there are two agents involved. The program *agPyrServer* is required for creating and modifying image pyramids. The main functionality of the program is contained in the agent *agODBserver* which is responsible for presenting the project specific data via *UX* and for manipulating these data. It uses *ORIENT* in a server version (*ORIserver*) as a server for adjustment, and it uses a server version of *SCOP.TDM* (*TDMserver*) for management of topographic data.

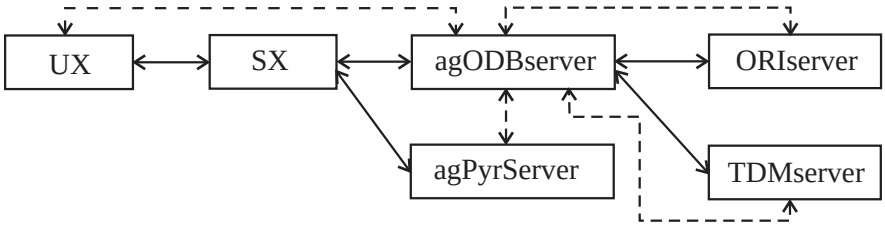


Figure 4.11: Architecture of *ORPHEUS*. Full arrows: message passing using the *CORBA* standard. Dashed arrows: bulk data (e.g. digital images) are exchanged via disk files.

The program *agODBserver* is a typical application of the C++ *ORIENT* data base interface described in section 4.3.4. It consists of a data base viewer which accesses the data base and is responsible for visualization of the data base contents, especially for visualizing the rooms in the way depicted in figure 4.10. If a room is displayed on the computer screen, it is added to the list of currently displayed rooms. This list also has a reference to the client-specific C++ interface for accessing the program *ORIserver* via *CORBA*. It can be accessed by any

ORPHEUS tool for interactive measurement. If such a tool (for instance the tool for interactive measurement of points) is activated, the list of displayed rooms receives a message telling it which tool has been activated, and the measurement tool may access all displayed rooms for its specific visualization purposes. It may also activate modes for interactive editing in all these windows. On the other hand, it is notified about all user interactions occurring in one of the displayed rooms so that it may perform its specific actions which can comprise manipulations of the data base. For instance, the tool for interactive measurement of points is informed about all points digitized in any window by a mouse click. As soon as that has happened, the recently digitized point has to be inserted into the room where it was digitized, it has to be inserted into the reference system, and it has to be drawn in the window where it was digitized. These actions could not be performed by the list of displayed rooms or by the room visualizer themselves because it is the measurement tool which gives a meaning to the user interactions. A measurement tool can also access the client specific interface for *ORIserver* and thus hybrid adjustment using *ORIENT*. The program for semi-automatic building extraction which will be described in section 6 is a typical example for such a measurement tool in *ORPHEUS* making not only use of the visualization facilities of the list of currently displayed rooms and the C++ data base interface, but also of hybrid adjustment after certain user interactions.

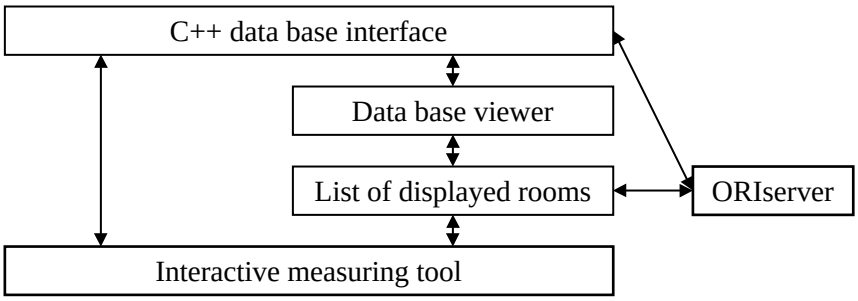


Figure 4.12: Architecture of data management, data visualization and measurement tools in *ORPHEUS*.

4.5 Practical aspects of photogrammetric reconstruction of 3D objects

4.5.1 Image acquisition

Due to the application-specific requirements regarding speed, accuracy and object dimensions, different image acquisition methods for 3D reconstruction have to be used. The most commonly used sensors for 3D reconstruction are video cameras, CCD cameras, analogue photographic cameras and remote sensing scanners.

- *Video cameras*: Conventional analogue video cameras are connected to a PC with a frame grabber which performs conversion of the analogue video signal to digital images. The size of these images is typically 768×572 pixels which corresponds to 0.44 MB per band. These cameras are relatively cheap, and they are well-suited for real-time applications; this is why they are used for industrial and medical purposes. On the other hand, both their sensor size and their resolution are restricted. Currently, really digital video cameras are gaining increasing importance.

- *Amateur cameras with CCD sensors*: CCD Sensors can be mounted in the image planes of conventional photographic cameras. In addition, such cameras need a device for data storage, e.g. a PCMCIA drive, Flash card, etc. They can then be used just like analogue cameras, the advantage being that the images can be checked immediately after they have been taken on a laptop PC, and bad photographs can be replaced by better ones. The sensor size varies considerably between different sensor models: A typical one-chip sensor may have about 2000×3000 pixels which corresponds to 6 MB per grey scale image or to 18 MB for a true color image. The format of these sensors is about 2.4×1.6 cm^2; thus, it is still 33%

smaller than a common small format analogue photograph. These cameras can be used for architectural applications and basically for everything that can be photographed because their handling is very flexible. However, in order to achieve an economic operating cycle, camera objectives with small focal lengths have to be used which enlarge the aperture angle but bring about geometrical problems due to distortions. The latest achievement is a digital aerial camera consisting of four CCD chips delivering four perspective images which can be resampled to one quasi-perspective digital image [Hinz et al., 2000].

- *Analogue metric cameras*: Photographs taken by metric cameras correspond with high accuracy to central perspective images. These cameras deliver analogue images which have to be scanned off-line. They are used for high-precision applications or if the format of the CCD sensors is too small for an economic operating cycle, which is especially true for, e.g., mapping purposes. Even the digital aerial camera cited above is not yet operational, and for high-precision applications, the resampling process required for combining the four images is not appropriate. Scanning off-line turns out to be a very time-consuming process, which is especially true for aerial images: The format of aerial images is usually 23×23 cm^2, and due to the high demands for accuracy, they have to be scanned with high resolution, thus yielding an enormous amount of data:

  - 15 μm: 256 MB per grey scale image (16000×16000 pixels).
  - 30 μm: 64 MB per grey scale image (8000×8000 pixels).

  The image size for terrestrial metric cameras is typically 12×9 cm^2 which corresponds to 8000×6000 pixels or 48 MB per grey scale image at a pixel size of 15 μm.

- *Rotational and line scanners*: In the case of (satellite or airborne) line scanners, a comparable amount of data must be handled. Images consisting of a single line are composed into an "infinite" strip making use of the sensor motion. Another mapping function than the perspective transformation (section 4.2.2) has to be used. In the standard case, the line scanner is arranged perpendicular to the direction of movement. Each line is oriented individually using the orbit description of the respective satellite or aircraft. A refinement of these orientations is possible using reference points on ground. Two line scanners simultaneously looking forward and backward realize a stereo configuration, e.g. [Tempelmann et al., 2000]. Remote sensing images are taken in many different spectral bands of the visible and infrared spectrum with a spatial resolution typically between 40 m and 5 m on ground per pixel. Airborne sensors reach a much higher resolution, but calibration of the data is rather complex due to the random characteristics of the sensor motion.

4.5.2 Sensor orientation procedures

The goal of 3D reconstruction is the inversion of the perspective, i.e. the computation of object co-ordinates from measured camera co-ordinates. For that purpose, several tasks have to be solved in advance:

- Camera calibration, i.e., determining the position of the projection centre and of the distortion parameters in the camera co-ordinate system, thus the determination of $\mathbf{p}_0$ and $\mathbf{adp}$ in equations 4.24 and 4.25. For metric cameras, this step is performed by the camera manufacturer, and the calibrated parameters are contained in the calibration protocol. Otherwise, cameras can be calibrated in an off-line procedure [Kraus, 1993].

- Inner orientation, i.e., establishing the relation between the camera co-ordinate and the sensor co-ordinate systems. As stated in section 4.2.2, this is only necessary for scanned analogue images. In this case, the sensor co-ordinates of the fiducial marks have to be measured. As their camera co-ordinates are known, the parameters of the affine transformation (equation 4.26) can be determined. After that step, the position of the projection centre relative to the images is given by the camera parameters (u_{pp}, v_{pp}, f).

- Outer orientation, i.e., determining the projection centre $\mathbf{P}_0$ and the rotations of the image plane relative to the object co-ordinate system (figure 4.4). The parameters of outer orientation can be determined from points with known object co-ordinates (control points). If high precision is required, control points can be targeted. At least 3 control points are required for the outer orientation of a single image.

It is not sufficient to know the parameters of inner and outer orientation in order to reconstruct a point $\mathbf{P}$ from its image point $\mathbf{p}$: in order to compute $\mathbf{P}$'s three object co-ordinates (X, Y, Z) from its camera co-ordinates (u, v), only two equations 4.24 are available. The position of $\mathbf{P}$ along the imaging ray cannot be determined without additional information. This information can be given by an assumption about the object, e.g. by the assumption $Z = 0$ for a planar object, or by another ray coming from a second image.

4.5.3 Stereo reconstruction

Stereo reconstruction is based on the same principle as the human visual system uses for depth recovery. Two cameras viewing the same scene under different perspective transformations are used for being able to reconstruct 3D objects. In order to keep the perspective distortions of the two images similar (and to render possible stereoscopic view in a stereo plotter), the viewing directions should be approximately parallel. Each scene point is projected on different locations in the two sensors and can be localized using the perspective equations 4.24.

Stereoscopy is a widely used method for surface reconstruction. Originally, analogue cameras were used for image capture, followed by a manual evaluation to get 3D points from stereoscopic images. Growing computational resources enabled the development of systems that strongly support this process, both in terms of sensors and of data processing.

A general stereoscopic system takes at least two different views of the scene to be observed. It can be realized by one moving imaging sensor or by several sensors at different locations.

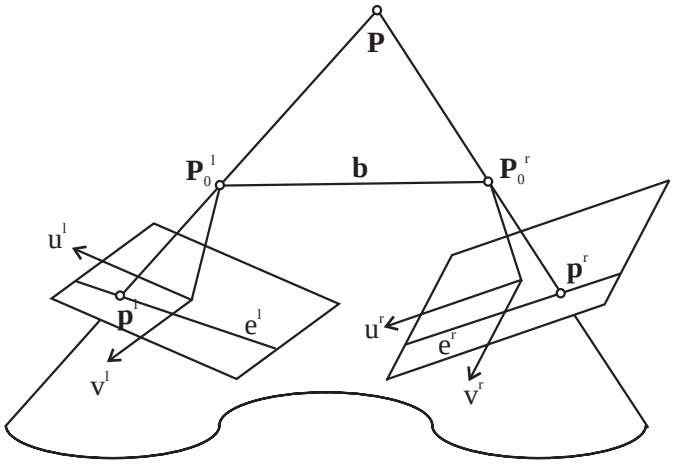


Figure 4.13: Geometry of the common 2-camera stereo model.

Figure 4.13 depicts the standard case of two cameras with identical physical properties. The projection centre of the left image l is $\mathbf{P}_0^l$, and the projection centre of the right image r is $\mathbf{P}_0^r$; the corresponding rotation matrices are $\mathbf{R^l}$ and $\mathbf{R^r}$. The vector $\mathbf{b} = \mathbf{P}_0^r - \mathbf{P}_0^l = (b_X, b_Y, b_Z)^T$ is called *stereoscopic base line*. If the orientation parameters of both images are known, an object point $\mathbf{P}$ can be determined by the intersection of the image rays from two corresponding image points $\mathbf{p}^l$ and $\mathbf{p}^r$: each point gives two equations 4.24 which can be solved for the unknown object co-ordinates $(X, Y, Z)^T$. As there are four equations but only three unknowns, the problem is over-determined and can be solved by a least squares adjustment.

An important property of any arrangement of two images of an identical scene and, thus, of all stereo arrangements, is *epipolarity* (figure 4.13): the vectors $\mathbf{b}$, $\overline{\mathbf{P}_0^l\mathbf{P}}$ and $\overline{\mathbf{P}_0^r\mathbf{P}}$ form a plane called *epipolar plane* ε. This

plane intersects the image planes in the epipolar lines e^l and e^r. The coplanarity condition can be written as follows [Brandstätter, 1991]:

$$(\mathbf{p}^l - \mathbf{p}_0^l)^T \cdot \mathbf{R}^{l^T} \cdot \mathbf{S} \cdot \mathbf{R}^r \cdot (\mathbf{p}^r - \mathbf{p}_0^r) = 0 \tag{4.29}$$

with

$$\mathbf{S} = \begin{pmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{pmatrix} \tag{4.30}$$

and $\mathbf{p}_0^l$ and $\mathbf{p}_0^r$ being the projection centres in the camera co-ordinate systems of the left and right image, respectively. If the orientation parameters of both images are known, for a point $\mathbf{p}^l$ in the left image, the corresponding point $\mathbf{p}^r$ in the right image is situated on the epipolar line e^r which is given by equation 4.29. This is an important property which is used in image matching to reduce search space (cf. chapter 5).

4.5.4 Bundle block configurations

Stereo configuration is often used in 3D reconstruction. However, using that configuration leads to problems with occluded object parts and with objects which are too large to be covered by a stereo model. In addition, there is a reliability problem: the depths of points determined from stereo images are not checked by other observations. A more general configuration is *bundle block configuration*. In this case the object is photographed from arbitrary positions so that each part of the object is at least visible in two (better: three or more) images and the intersection angles at the object points are close to 90° (figure 4.14).

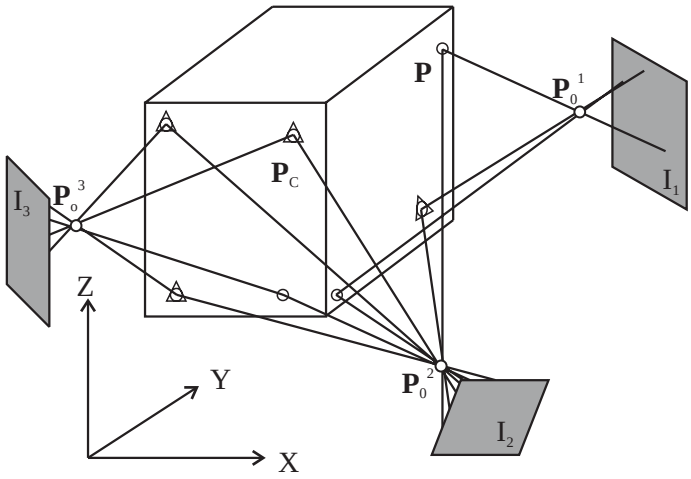


Figure 4.14: A bundle block configuration.

In the images, the camera co-ordinates of control points (points with known object co-ordinates, e.g. $\mathbf{P}_C$ in figure 4.14) and tie points (points with unknown object co-ordinates, e.g. $\mathbf{P}$ in figure 4.14) are measured. From these observations, the orientation parameters of all images I_i and the object co-ordinates of the tie points $\mathbf{P}$ are determined simultaneously by least squares adjustment. For each measured image point, two equations 4.24 are used in the adjustment, the unknowns being $\mathbf{P}, \mathbf{P}_0^i$ and the three angles determining $\mathbf{R}^i$. As equations 4.24 are non-linear, they have to be linearized using approximate values for the unknowns; adjustment has to be performed iteratively in the way described in section 4.1. If non-metric cameras are used, the parameters of inner orientation (u_{pp}, v_{pp}, f) as well as the additional parameters modelling distortions will also be unknown; camera calibration will then be performed "on the job". Of course, other types of observations (e.g. those listed in section 4.2) can be adjusted together with the camera and control point co-ordinates, too.

On the one hand, bundle block adjustment is used for the determination of the orientation parameters of all photographs. On the other hand, bundle block configuration also increases both the reliability and the accuracy of object reconstruction: An object point $\mathbf{P}$ can be determined by intersection from more than two images, which provides local redundancy for gross error detection and which in general will result in a better intersection geometry. For instance, if $\mathbf{P}$ has been observed in three images, there are six equations 4.24 for the determination of its three unknown object co-ordinates. This local redundancy facilitates the elimination of gross errors in the data as it was described in section 4.1.1.

Part III

Automated data acquisition in digital photogrammetry

Chapter 5

Automatic reconstruction of object surfaces

The goal of current research work in the field of vision-based 3D reconstruction is the automation of both orientation and object reconstruction tasks by applying digital image processing and pattern analysis. The central tasks to be solved in this context are [Lang and Förstner, 1998]:

1. *Object location:* The determination of the pose (position and orientation) of some pre-defined objects in a digital image or in a set of digital images. The relation between the object or between parts of the object and the image(s) has to be determined. In the context of photogrammetric plotting, the object to be located might, for instance, be a fiducial mark, a targeted control point or an untargeted control point for which an explicit object model is available. Object location is typically solved by top-down procedures.

2. *Object reconstruction:* The determination of the shape and, eventually, the structure of the object. For that purpose, at least two images (or another data source) have to be used because a 3D object cannot be reconstructed from a single 2D image. A relation between the images has to be established to find corresponding structures in images from which the 3D information can be deduced. The problem of object reconstruction is not yet solved in a general manner in the sense that the structure of a scene can be derived automatically, including semantic aspects. However, object reconstruction techniques for particular object classes have been developed, e.g. for the derivation of DEMs or the extraction of man-made objects which can be described by polyhedra. Bottom-up procedures are required for object reconstruction purposes, but in case explicit model knowledge about the object is available, these procedures can be combined with model driven techniques.

One of the central issue in both tasks is concerned with the solution of the *correspondence problem*, i.e. the establishment of a relation either between two or more images or between one or more images and an object which is to be reconstructed or located. This problem is solved by *matching techniques*. In case a correspondence between a pre-defined object model and one or more image(s) has to be established, we speak about *object-to-image matching*, whereas the establishment of correspondences between a set of images is called *image matching*. Astonishingly enough, both object location and object reconstruction may involve similar matching algorithms. However, even though matching is of crucial importance in the solution of these problems, it just provides intermediate results.

In this chapter, we want to concentrate on the automation of object surface reconstruction on the basis of matching techniques without semantic interpretation. We start with a discussion of feature extraction in section 5.1 because the derivation of a symbolic image description is an essential part of almost all automation techniques in computer vision. After that, an overview about matching techniques will be given in section 5.2. An important issue in object reconstruction is concerned with the question how approximate values can be obtained. This problem is often solved by using a coarse-to-fine strategy by iteratively applying matching to *image pyramids*. Section 5.3 is dedicated to such *hierarchical* approaches. Finally, in section 5.4 we want to describe a general framework for object reconstruction we have developed in the course of this work. This framework aims at

providing general strategies applicable for a great variety of matching and reconstruction tasks without trying to find a solution to the general case of automatic scene interpretation. The framework is the basis of automatic fine measurement of buildings as it will be presented in chapter 6.

5.1 Extraction of features from digital images

On a symbolic level, a digital image can be represented by image features which contain the information relevant for subsequent image interpretation or object reconstruction tasks. We will start with an overview on techniques for the extraction of different feature types in section 5.1.1. After that, we will describe a method for simultaneously extracting all these types of features in digital images developed by [Fuchs, 1998] because a variation of this method is applied in our framework for object reconstruction (section 5.4).

5.1.1 Feature classes

Many techniques for feature extraction have been proposed in literature. These techniques differ by the type of features which are to be extracted, and they also apply different mathematical models, which results in the fact that feature extraction can be based on the analysis of either the first or the second derivatives of the grey levels. We can distinguish three classes of features:

1. *Homogeneous image regions*
2. *Image edges*
3. *Image points*.

Extraction of homogeneous image regions: Homogeneous regions are image areas fulfilling a certain similarity criterion, e.g. homogeneity of intensity or colour. It is the goal of segmentation algorithms to split the image into homogeneous regions which are connected and bordered by non-intersecting edges. The individual regions can either be represented by the closed polygons being their boundaries or by enumeration of their member pixels. There are various approaches for region extraction:

- Thresholding techniques: According to some (user-defined or automatically derived) threshold(s), the pixels of a digital image are classified. After that, neighbouring pixels belonging to the same class have to be merged [Gonzalez and Wintz, 1977].
- Region growing: First, seed regions have to be extracted, and these seed regions are iteratively grown at their borders by accepting new pixels being consistent with the pixels already being contained in the region. After each iteration, the homogeneity value of the region has to be re-calculated using also the new pixels. The results of region growing heavily depend of a proper selection of seed points [Gonzalez and Wintz, 1977].
- Split and merge: The image is successively split into sub-areas as long as the homogeneity criterion is not fulfilled. In a second step, neighbouring areas showing similar homogeneity measures are merged, e.g. [Fuchs and Heuel, 1998].

Extraction of image edges: Edges are characterized by abrupt changes of intensity or colour in the images. Edges can either be boundaries between two regions (*step edges*) or small elongated objects of only a few pixels width so that the boundaries on both sides of that object are too close to be separated (*line edges*) [Nalwa and Binford, 1986]. Edge detection consists of several steps:

1. Finding edge candidate pixels: The digital image is convolved by either a first or a second order derivative kernel depending on the underlying edge model. [Canny, 1986] uses the first derivative of a Gaussian kernel for that purpose. Edge candidate pixels correspond to pixels with great first derivatives so that a threshold has to be found to separate these candidates from homogeneous pixels. The selection of a threshold is very critical in this context. Any threshold should be derivable from a statistical evaluation of the image data. The problem of thresholding is avoided by using second derivative operators, e.g. the Laplacian of Gaussian (LoG) operator [Nalwa and Binford, 1986]. Edges correspond to the zero crossings of the second derivatives of the grey levels. As derivation is very sensitive with respect to noise, the images have to be smoothed beforehand. That is the reason why the Gaussian convolution kernel is used in the above algorithms. For that reason it is also clear that operators using the second derivatives are more afflicted by noise than first derivative operators.

2. Finding the optimum candidates: If first derivatives are evaluated, step 1 will result in edge candidate regions usually being more than one pixel wide. After that, the optimum candidate pixel (the one having the greatest derivative across the edge direction) has to be found.

3. Finding edge elements: The result of the previous step is still a binary raster image containing the edge candidate pixels. In this step, the optimum position of the edge element is computed with sub-pixel accuracy. An edge element is considered to be an edge point with a tangent to the edge [Nalwa and Binford, 1986]. The position of the edge element can be computed from zero crossings of the directional second derivative of a Gaussian function across the edge [Canny, 1986] or from an approximation of the grey levels in a neighbourhood of the edge candidate pixel by either a plane, a third-order polynomial or a $\tanh$ function [Nalwa and Binford, 1986].

4. Edge tracking: Neighbouring edge elements of similar edge direction have to be connected to form edge pixel chains by applying edge tracking algorithms, e.g. [Kerschner, 1995].

5. Edge approximation: The edge pixel chains have to be thinned out and approximated by some analytic functions, e.g., by straight line segments [Fuchs, 1998] or by splines [Forkert et al., 1995]. The parameters of these functions can be estimated by adjustment using the edge elements as observations.

Extraction of image points: Points are small features without physical dimension. They either appear at *end points*, *corners* or *junctions* of edges, or they can be small blobs inside homogeneous regions of a small extent (*circular symmetric points*) [Fuchs, 1998]. They can be found by

- Optimizing a certain detectability criterion. For instance, using the Förstner operator [Förstner and Gülch, 1987], points are extracted at positions which promise the minimum r.m.s. errors in Least Squares Matching (LSM) (section 5.2.1.2). By using this operator, all of the above point types can be detected.

- Analyzing the curvatures of image edges. These operators can detect corner points at positions of maximum curvature of edges previously detected by some edge detection technique.

5.1.2 Polymorphic feature extraction

A framework for simultaneous extraction of point and line features is *polymorphic feature extraction* based on the Förstner operator [Förstner and Gülch, 1987, Fuchs, 1998]. The framework is based on a statistical analysis of the grey level gradients $\nabla g(r,c)$, thus the first derivatives of the grey level function $g(r,c)$. In this framework, the grey level function $g(r,c)$ is assumed to consist of a "true" image function $f(r,c)$ and white additive Gaussian noise $n(r,c)$:

$$g(r,c) = f(r,c) + n(r,c) \tag{5.1}$$

Note that under these assumptions, $n(r,c)$ is normally distributed with expectation 0 and variance σ_n^2, the *noise variance*. To be more precise, $n(r,c)$ follows a Poisson distribution which can be approximated by a Gaussian distribution with a signal dependent variance [Brügelmann and Förstner, 1992]:

$$\sigma_n^2 = \sigma_n^2(r,c) = \sigma_n^2[f(r,c)] = a + b \cdot f(r,c) \approx a + b \cdot g(r,c) \tag{5.2}$$

Using this noise model, in the homogeneous regions, $g(r,c)$ can be assumed to be normally distributed with expectation μ_g (i.e. the intensity inside the homogeneous region) and variance $\sigma_n^2 = a + b \cdot \mu_g$ from equation 5.2.

The grey level gradient $\nabla g(r,c)$ can be computed from:

$$\nabla g(r,c) = \begin{bmatrix} \Delta g_r(r,c) \\ \Delta g_c(r,c) \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} g(r+1,c) - g(r-1,c) \\ g(r,c+1) - g(r,c-1) \end{bmatrix} \tag{5.3}$$

or by a convolution of $g(r,c)$ with the first derivative of a Gaussian function $\mathbf{G}_\sigma$ of standard deviation σ:

$$\nabla g(r,c) = \begin{bmatrix} \Delta g_r(r,c) \\ \Delta g_c(r,c) \end{bmatrix} = \begin{bmatrix} \partial/\partial r \mathbf{G}_\sigma \\ \partial/\partial c \mathbf{G}_\sigma \end{bmatrix} \star g(r,c) \tag{5.4}$$

if smoothing with a Gaussian kernel of variance σ shall be performed. The components of the grey level gradients in homogeneous regions are, again, normally distributed with expectation 0 and variance $\sigma_{n'}^2 = k \cdot \sigma_n^2$ where k depends on the operator used for computing the derivatives. k can be derived from applying the law of error propagation to equations 5.3 or 5.4, respectively. It equals the squared sum of components of the convolution kernel. Thus, if the operator in equation 5.3 is used, $k = 0.5^2 + 0.5^2 = 0.5$. For the operator in equation 5.4 it can be shown that $k = \frac{1}{8\pi\sigma^4}$ [Fuchs, 1998].

The noise variance σ_n^2 can be estimated from the histogram of the squared grey level gradients $||\nabla g(r,c)||^2$. [Förstner, 1991] gives a solution for grey level images under the assumption of normally distributed noise. [Brügelmann and Förstner, 1992] show how the parameters a and b for the signal dependent noise variance in equation 5.2 can be estimated. In that paper, the theory is also expanded to handling colour images with independent bands.

From the grey level gradients $\nabla g(r,c)$ of a small window, e.g. 3×3 or 5×5 pixels^2, a measure W for texture strength can be calculated as the average squared norm of the grey level gradients normalized by $\sigma_{n'}^2$ [Fuchs, 1998]:

$$W = \mathbf{L} \star ||\frac{1}{\sigma_{n'}^2} \cdot \nabla g(r,c)||^2 = \mathbf{L} \star (\frac{\Delta g_r^2 + \Delta g_c^2}{\sigma_{n'}^2}) = \mathbf{L} \star (\frac{\Delta g_r^2 + \Delta g_c^2}{k \cdot \sigma_n^2}) \tag{5.5}$$

with $\mathbf{L}$ being a linear lowpass filter, e.g. a 3×3 or a 5×5 binomial filter (this can be interpreted as an approximation for a Gaussian filter with $\sigma = 0.71$ or $\sigma = 1$, respectively). W will be high in windows containing great grey level differences. Note that due to the normalization, $\Delta g_r/\sigma_{n'}$ and $\Delta g_c/\sigma_{n'}$ are normally distributed with expectation 0 and variance 1. If the model from equation 5.2 is used for computing σ_n^2, it has to be computed for each pixel in dependence of the grey level $g(r,c)$. If σ_n^2 is supposed to be identical for all pixels, normalization of W can be performed a posteriori.

In addition to W, a measure Q for isotropy of texture can be computed from the normalized grey level gradients. Q can be derived from the equations for Least Squares Matching (LSM) (section 5.2.1.2): suppose an image point has been found at a certain position (r_1, c_1) in image I_1 and the co-ordinates of corresponding point in image I_2 are to be found such that $(r_2, c_2)^T = (r_1, c_1)^T + (\Delta r, \Delta c)^T$. If the unknown shifts $(\Delta r, \Delta c)$ are to be computed from the grey level differences $\Delta g_{12} = g_2[(r_1, c_1)^T + (\Delta r, \Delta c)^T] - g_1(r_1, c_1)$ in a small neighbourhood of (r_1, c_1) by least squares adjustment, the normal equation matrix $\mathbf{N}$ looks as follows if the observation weights are contained in the filter matrix $\mathbf{L}$:

$$\mathbf{N} = \begin{pmatrix} \mathbf{L} \star \frac{\Delta g_r^2}{\sigma_{n'}^2} & \mathbf{L} \star \frac{\Delta g_r \cdot \Delta g_c}{\sigma_{n'}^2} \\ \mathbf{L} \star \frac{\Delta g_r \cdot \Delta g_c}{\sigma_{n'}^2} & \mathbf{L} \star \frac{\Delta g_c^2}{\sigma_{n'}^2} \end{pmatrix} \tag{5.6}$$

Note that the normal equations can be derived from the grey levels of one image only. The standard deviations of the resulting shifts can be estimated a priori from the inverse $\mathbf{Q_{xx}} = \mathbf{N}^{-1}$ of $\mathbf{N}$. In addition, the error ellipse can be analyzed a priori. Its axes are proportional to the square roots of the eigen-values λ_1 and λ_2 of $\mathbf{N}^{-1}$. Thus, the ratio λ_2/λ_1 gives a measure for the isotropy of texture. In [Förstner, 1991], the following measure for Q is proposed:

$$Q = 1 - \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2 = \frac{4 \cdot \det(\mathbf{N})}{trace^2(\mathbf{N})} \tag{5.7}$$

Q from equation 5.7 is bounded by 0 and 1. It equals 0 if all the gradients in the small image patch are parallel, and it is 1 if the gradient directions are equally distributed. Figure 5.2 shows the W and Q images derived from the image in figure 5.1 with $\mathbf{L}$ being a 3×3 Gaussian filter ($\sigma = 0.71$).

Figure 5.1: Original image.

By applying thresholds W_{min} and Q_{min} to W and Q, each pixel can be classified as belonging either to a homogeneous region, to a point region or to a region containing an edge:

1. $W < W_{min}$: the pixel is inside a homogeneous region.
2. $(W \geq W_{min}) \wedge (Q < Q_{min})$: the pixel is inside an edge region.
3. $(W \geq W_{min}) \wedge (Q \geq Q_{min})$: the pixel is inside a point region.

As the classification result is especially sensitive to the selection of the threshold W_{min} for texture strength, this threshold has to be selected very carefully. W_{min} has to depend on the image contents. As we have seen above, $\Delta g_r/\sigma_{n'}$ and $\Delta g_c/\sigma_{n'}$ follow a normal distribution with expectation 0 and variance 1. Thus, the squared sum of these entities follows a $\chi^2_{2.N_b}$ distribution with $2 \cdot N_b$ degrees of freedom, where N_b is the number of image bands. Thus, the selection of W_{min} can be replaced by selecting a significance level α (e.g., $\alpha = 0.05$) for a hypotheses test and select W_{min} to be

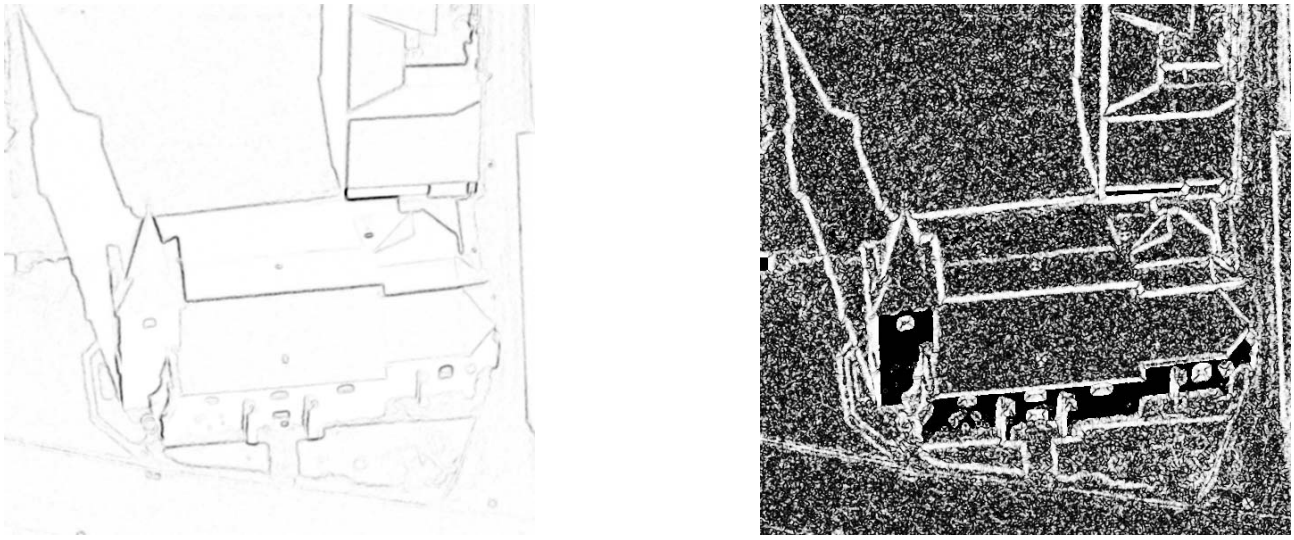


Figure 5.2: Left: W Image. Right: Q Image. White: Q = 0; black: Q = 1 or Q is undetermined due to $trace(\mathbf{N}) = 0$.

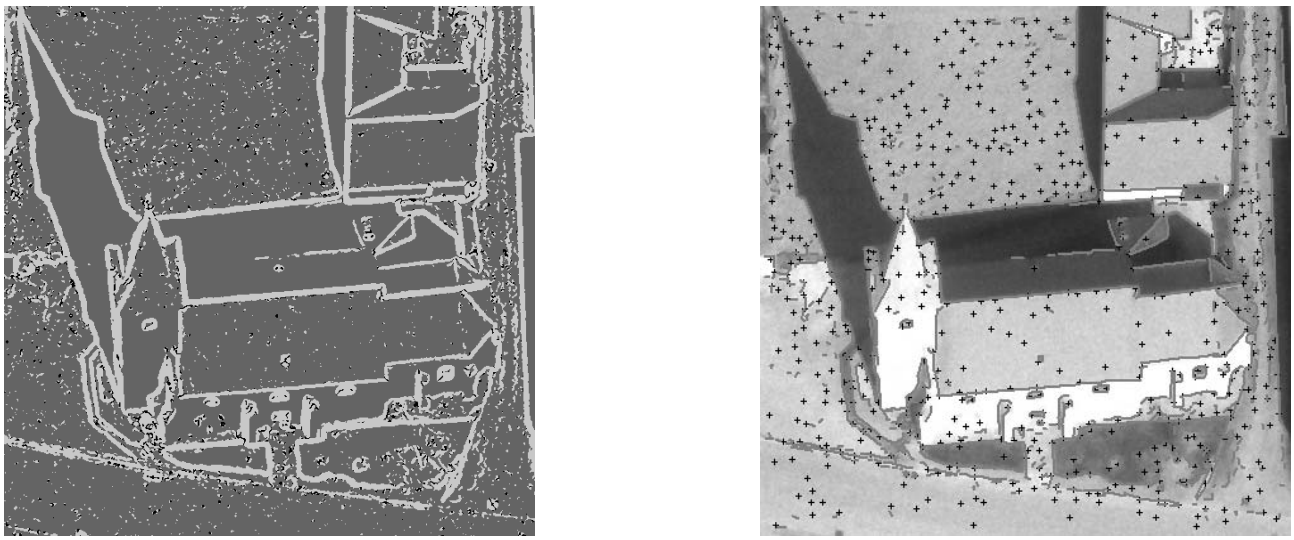


Figure 5.3: Left: Classified image: point regions (black), line regions (light grey), homogeneous regions (dark grey). Right: Extracted points and lines superimposed to the original image.

$$W_{min} = F(\chi^2_{2 \cdot N_b;\alpha}) \tag{5.8}$$

where $F(\chi^2_{2 \cdot N_b;\alpha})$ is the α fractil of the χ^2 distribution. For $N_b = 1$, i.e., grey level images, $F(\chi^2_{2 \cdot N_b;\alpha}) = F(\chi^2_{2;\alpha}) = -2 \cdot \log(\alpha)$ [Fuchs, 1998]. Another way to select W_{min} in dependence on the image contents is given by using a fixed value for σ_n instead of estimating it from the image data contents and choosing [Förstner, 1991]

$$W_{min} = j \cdot median(W) \tag{5.9}$$

with j being a constant for a certain class of images. The selection of j corresponds to the selection of a significance level, and taking the median of W replaces the estimation of σ_n. This way of selecting W_{min} can be used if for some reason the user thinks that the estimation of σ_n does not deliver appropriate results. The selection of Q_{min} is less critical because Q is bound by 0 and 1. Thus, Q_{min} can be chosen to be e.g. 0.7.

In the implementation we use for our framework for object reconstruction (cf. section 5.4), we use binomial filters as an approximation for the Gaussian filter kernel in equations 5.5 and 5.6. W_{min} is chosen according to equation 5.9 with a user-defined factor j. By default, $j = 2.5$ is chosen. Thresholding results in a digital image where each pixel can take one of three values, depending on whether the pixel was classified as being in a homogeneous, an edge or a point region (figure 5.3, left). After classification, point, edge, and homogeneous regions are treated independently from each other to actually extract these feature types. In our implementation, we restrict ourselves to extracting edges and points, which we will describe in sections 5.1.3 and 5.1.4, respectively.

5.1.3 Extraction of points

For extracting points, larger filter kernels than for edge extraction have to be used. Thus, in the regions classified as point regions, W is re-computed using a larger size for the filter kernel $\mathbf{L}$ in equation 5.5 than for classification. Point extraction is then split into three steps [Förstner, 1991, Fuchs, 1998]:

1. Select the optimal windows inside the point regions by searching for relative maxima of texture strength W. A point is supposed to be located in a window of the size of $\mathbf{L}$ centered at a relative maximum of W. By selecting a window size for the search for relative maxima, the minimum distance of two extracted points is specified (non-maxima-suppression).

2. Compute the co-ordinates of each point using two models:

   (a) The point is a corner point.

   (b) The point is a circular symmetric point.

   For both models, $\hat{\sigma}_o$ has to be estimated, thus we obtain $\hat{\sigma}_{o_c}$ from assuming the point to be a corner point and $\hat{\sigma}_{o_s}$ from assuming it to be a circular symmetric point.

3. Using $\hat{\sigma}_{o_c}$ and $\hat{\sigma}_{o_s}$, a Fisher test can be performed in order to classify the point. According to the results of classification, each point is assigned the co-ordinates and the r.m.s. errors corresponding to the optimal point model.

Figure 5.4: Two point models. Left: a corner point. Right: a circular symmetric point. The point $\mathbf{p}$ is computed from an intersection of all lines l by least squares adjustment.

For the first (the corner) model, the point is supposed to be the intersection of two or more grey level edges. Thus, for each pixel inside the optimal window, a straight line l through the pixel centre orthogonal to the grey level gradient ∇g can be formulated. The point $\mathbf{p}$ is supposed to be the intersection of all these lines l, and it can be computed from a least squares adjustment, taking the distance d of $\mathbf{p}$ from l to be fictitiously observed by 0 (figure 5.4, left). Assuming the weights of these observations to be proportional to the lengths of the grey level gradient ∇g, the co-ordinates (r_c, c_c) of the point according to the corner model can be computed from [Förstner, 1991, Förstner and Gülch, 1987]:

$$\begin{pmatrix} \sum \Delta g_r^2 & \sum \Delta g_r \Delta g_c \\ \sum \Delta g_r \Delta g_c & \sum \Delta g_c^2 \end{pmatrix} \cdot \begin{pmatrix} r_c \\ c_c \end{pmatrix} = \begin{pmatrix} \sum r \Delta g_r^2 + \sum c \Delta g_r \Delta g_c \\ \sum c \Delta g_c^2 + \sum r \Delta g_r \Delta g_c \end{pmatrix} \tag{5.10}$$

For the second (the circular symmetric) model, the point is supposed to be the centre of a small region which more or less resembles a circle. Thus, for each pixel inside the optimal window, a straight line l through the pixel centre in direction of the grey level gradient ∇g can be formulated. The point $\mathbf{p}$ is supposed to be the intersection of all these lines l, and again it can be computed from a least squares adjustment, taking the distance d of $\mathbf{p}$ from l to be fictitiously observed by 0 (figure 5.4, right). Assuming the weights of these observations to be proportional to the lengths of the grey level gradient ∇g, the co-ordinates (r_s, c_s) of the point according to the circular symmetric model can be computed from [Förstner, 1991, Förstner and Gülch, 1987]:

$$\begin{pmatrix} \sum \Delta g_c^2 & -\sum \Delta g_r \Delta g_c \\ -\sum \Delta g_r \Delta g_c & \sum \Delta g_r^2 \end{pmatrix} \cdot \begin{pmatrix} r_s \\ c_s \end{pmatrix} = \begin{pmatrix} \sum r \Delta g_c^2 - \sum c \Delta g_r \Delta g_c \\ \sum c \Delta g_r^2 - \sum r \Delta g_r \Delta g_c \end{pmatrix} \tag{5.11}$$

In both cases, the sums are to be taken over all windows inside the optimal window. The r.m.s. errors of the weight units can be obtained from equation 4.8, and the r.m.s. errors of the point co-ordinates are computed from equation 4.9. For the classification of a point, a Fisher test can be performed by comparing $\hat{\sigma}_{o_c}$ and $\hat{\sigma}_{o_s}$: $\hat{\sigma}_{o_c}^2 / \hat{\sigma}_{o_s}^2$ is Fisher-distributed, and the test can answer the question whether one of the models (the one with the smaller r.m.s. error of the weight unit) fits significantly better to the data or not. If one of the models is detected to fit significantly better than the other one, the point is assumed to belong to the according point class. If none of the models fits significantly better, i.e., if the difference of the variances can be explained by chance, the point is assumed to be a corner point. A point is represented by a set of attributes:

- a unique identifier ID,
- the point co-ordinates (r, c),
- the point's variance-covariance matrix $\mathbf{C}_{rc}$,
- the point type TYP with $TYP \in \{c, s\}$ indicating whether the point is a corner point or a circular symmetric point,
- the point's texture strength W_P.

5.1.4 Extraction of edges

In the regions classified as edges, all the steps for edge detection described in section 5.1.1 have to be carried out. Due to the smoothing of W by $\mathbf{L}$, the edge regions will be more than one pixel wide. The actual grey level edge is supposed to be at the position of maximum texture strength, and all pixels in edge regions where this is not the case have to be suppressed. However, in edge regions, neighbouring pixels in direction of the edge are supposed to have similar texture strengths. Thus, non-maximum suppression may only be performed in direction of the grey level gradient, i.e., orthogonal to the edge. The left part of figure 5.5 shows a method for non-maximum suppression according to [Canny, 1983]. For each pixel $\mathbf{P}$ classified as an edge pixel, the texture strengths $W(\mathbf{P}_1)$ and $W(\mathbf{P}_2)$ on the straight line v in direction of the grey level gradient ∇g have to be interpolated from the eight neighbouring pixels of $\mathbf{P}$ in the texture strength image W. If $((W_1 < W_P) \wedge (W_2 < W_P))$, then $\mathbf{P}$ is a relative maximum of W and thus an edge pixel, otherwise it is not. In this way, the edge regions are reduced to streaks of a width of one pixel.

It is possible to locate the edge with a resolution better than one pixel. One way for doing so is depicted in the right part of figure 5.5 [Canny, 1983]: given the interpolated texture strengths computed for non-maxima suppression, W can be approximated by a 2^{nd} order polynomial in the cross-section between $\mathbf{P}_1$ and $\mathbf{P}_2$. The position $\mathbf{P}_{max}$ of maximum texture strength W_{max} is identical to the position of the maximum of the

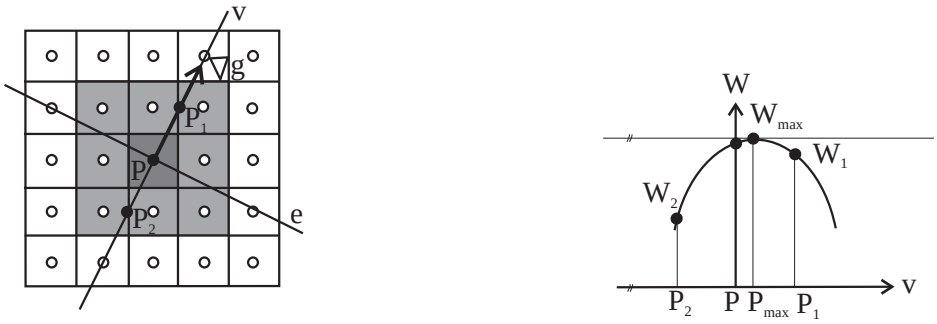


Figure 5.5: Left: Non-maxima suppression according to [Canny, 1983]: Texture strengths $W_1 = W(\mathbf{P}_1)$ and $W_2 = W(\mathbf{P}_2)$ have to be interpolated along the straight line v in direction of ∇g, i.e. orthogonal to the edge e. Right: sub-pixel estimation of the position of the edge point $\mathbf{P}_{max}$.

polynomial, i.e., it is found at the position where $\partial W/\partial v = 0$. By the point $\mathbf{P}_{max}$ and the direction angle of the edge e, an *edge element* is defined. [Nalwa and Binford, 1986] give another possibility for deriving both the position of the centre and the tangent of an edge element by approximating the grey level function g by a $\tanh$ function in a local neighbourhood of each edge pixel.

An edge element is characterized by a set of attributes:

- its (subpixel) position (r, c),
- its gradient vector ∇g,
- its texture strength W_e.

Edge tracking: Neighbouring edge elements have to be connected to edge pixel chains by an edge tracking algorithm. We use the algorithm described by [Kerschner, 1995] and [Forkert et al., 1995] for that purpose. Starting from an edge element not yet belonging to a previously accumulated edge, a neigbouring edge element is searched for which is located approximately in the tangent direction and approximately parallel to the tangent direction. Due to noise and discretization errors, the estimation of the tangent direction is quite uncertain, and there might also be gaps of a width greater than one pixel between neighbouring edge elements. Edge tracking is still performed in the raster domain, and it has to cope with these problems. The algorithm by [Kerschner, 1995] first searches through the first generation of neighbouring pixels in direction of the tangent in a way that the utmost angle between the tangent and the connection between the edge and the next candidate is 90°. Figure 5.6 shows an example for the possible candidates starting at an edge element e. If no candidate is found, the second generation of neighbouring pixels is searched for in a similar way. If an appropriate candidate is found, it is added to the current chain, and the procedure is continued at that pixel. Otherwise, the chain is finished in the current direction, and tracking will start again at the first pixel, but in the opposite direction, again until no more neighbour candidate is found.

Edge approximation: The resulting edge element chains are polygons with very short edges. They have to be approximated by analytic functions, e.g. by straight lines (polygons) [Fuchs, 1998] or by cubic splines [Forkert et al., 1995]. By the approximation procedure, the original edge element chains will be smoothed. The most important problem in this context is finding appropriate initial positions for the vertices of the approximating polygon or for the node points of an approximating cubic spline, i.e., the problem of thinning out the edge element chains. This can be done in two ways:

- *Merging* algorithms try to successively merge edge elements into one straight line segment as long as the approximation error is beneath a certain distance threshold and establish a polygon vertex and a new

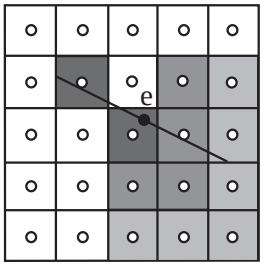


Figure 5.6: Edge tracking. Dark grey: pixels already belonging to the tracked edge chain. e: the last edge element in the chain. Medium grey: possible candidates among the first generation neighbours. Light grey: possible candidates in the second generation of neighbours.

straight line segment at positions where the threshold is hurt. The results of merging algorithms depend on the starting point.

- *Splitting* algorithms start by connecting both endpoints of the edge element chain with a straight line. After that, the edge element having the greatest distance from this straight line is searched for. If this distance is greater than a given threshold (e.g. 1 pixel), a new polygon vertex is established at that edge element, and the straight line is split into two segments. This procedure is recursively applied to the new segments until there is no more line segment with a maximum distance greater than the threshold [Fuchs, 1998].

After the initial node points have been detected by one of the above thinning algorithms, the edge pixel chains can be approximated by analytic functions. Approximation is essential in this context because, thinking of the results of the merging algorithm described above, the "surviving" edge elements are very often those which fit the actual edge worst. Approximation can be performed individually for each of the initial straight line segments, or it can be done for the whole edge at once. In the first case, after all segments have been approximated, the new positions of the polygon vertices have to be computed from intersections of the approximated curves [Fuchs, 1998]. The second case corresponds to approximating the edge by a (linear or cubic) spline [Forkert et al., 1995].

The straight line segments can be represented by two parameters, the directional angle φ and the length d of an orthogonal vector from the origin of the (r, c) co-ordinate system to the line (figure 5.7). The co-ordinates are reduced to the co-ordinates of the weighted centre of gravity $\mathbf{P}_g = (r_g, c_g)^T$ of the n edge elements belonging to the straight line segment. The equation of an image line segment l is:

$$l : (r - r_g) \cdot \cos\varphi + (c - c_g) \sin\varphi - d = 0 \tag{5.12}$$

where the centre of gravity can be estimated from:

$$\hat{\mathbf{P}}_g = \begin{pmatrix} \hat{r}_g \\ \hat{c}_g \end{pmatrix} = \frac{1}{\sum\limits_i W_i} \cdot \begin{pmatrix} \sum\limits_i W_i \cdot r_i \\ \sum\limits_i W_i \cdot c_i \end{pmatrix} \tag{5.13}$$

In equation 5.13, the edge elements i are weighted by their texture strengths W_i [Förstner, 1991]. Using the variance-covariance matrix $\mathbf{C}_g$ of the centre of gravity:

$$\mathbf{C}_g = \begin{pmatrix} \hat{\sigma}_r^2 & \hat{\sigma}_{rc} \\ \hat{\sigma}_{rc} & \hat{\sigma}_c^2 \end{pmatrix} = \frac{1}{\sum\limits_i W_i} \cdot \begin{pmatrix} \sum\limits_i W_i \cdot (r_i - \hat{r}_g)^2 & \sum\limits_i W_i \cdot (r_i - \hat{r}_g) \cdot (c_i - \hat{c}_g) \\ \sum\limits_i W_i \cdot (r_i - \hat{r}_g) \cdot (c_i - \hat{c}_g) & \sum\limits_i W_i \cdot (c_i - \hat{c}_g)^2 \end{pmatrix} \tag{5.14}$$

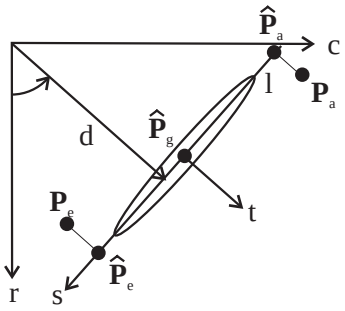


Figure 5.7: The representation of a line. (φ, d): polar co-ordinates of the vector from the origin of the sensor co-ordinate system to the line. $\hat{\mathbf{P}}_g$: centre of gravity of the edge elements.

φ can be estimated from the direction of the smallest eigenvector of $\mathbf{C}_g$, i.e. the direction of the smaller main axis of the error ellipse of the centre of gravity $\hat{\mathbf{P}}_g$ (figure 5.7), and d is the distance of the line already given by $\hat{\mathbf{P}}_g$ and φ from the origin of the co-ordinate system [Fuchs, 1998]:

$$\begin{aligned} \hat{\varphi} = &\quad \tfrac{1}{2} \cdot \arctan \frac{2 \cdot \hat{\sigma}_{rc}}{\hat{\sigma}_c^2 - \hat{\sigma}_r^2} \\ \hat{d} = &\quad \hat{r}_g \cdot \cos\hat{\varphi} + \hat{c}_g \cdot \sin\hat{\varphi} \end{aligned} \tag{5.15}$$

The co-ordinates of the endpoints $\hat{\mathbf{P}}_a$ and $\hat{\mathbf{P}}_e$ of the line segment can be derived from projecting the first edge element $\mathbf{P}_a$ and the last one $\mathbf{P}_e$ to the adjusted straight line (figure 5.7).

In order to derive the stochastic model of a line segment, we have to consider the situation in a local line co-ordinate system (s, t) with s being the direction of the line and t orthogonal to it (figure 5.7):

$$\begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} -\sin\varphi & \cos\varphi \\ \cos\varphi & \sin\varphi \end{pmatrix} \cdot \begin{pmatrix} r - \hat{r}_g \\ c - \hat{c}_g \end{pmatrix} = \mathbf{R}_\varphi{}^T \cdot \begin{pmatrix} r - \hat{r}_g \\ c - \hat{c}_g \end{pmatrix} \tag{5.16}$$

In this co-ordinate system, the line can be expressed as

$$l : t = k \cdot s + m \tag{5.17}$$

As the system (s, t) is centered at the centre of gravity of all edge elements, in this co-ordinate system, σ_k and σ_m are not correlated, thus $\sigma_{km} = 0$, and the variance-covariance matrix $\mathbf{C}_{km}$ of the line parameters is [Fuchs, 1998]:

$$\mathbf{C}_{km} = \begin{pmatrix} \hat{\sigma}_k^2 & \hat{\sigma}_{km} \\ \hat{\sigma}_{km} & \hat{\sigma}_m^2 \end{pmatrix} = \hat{\sigma}_o^2 \cdot \begin{pmatrix} \frac{1}{\sum_i s_i^2} & 0 \\ 0 & \frac{1}{n} \end{pmatrix} \approx \hat{\sigma}_o^2 \cdot \begin{pmatrix} \frac{12}{l_l^3} & 0 \\ 0 & \frac{1}{l_l} \end{pmatrix} \tag{5.18}$$

with n being the number of edge elements, l_l the length of the line segment and $\hat{\sigma}_o$ the r.m.s. error of the weight unit in the line approximation process. The approximation on the right side is valid for large n under the assumption that the distance between two adjacent edge elements is constant.

The stochastic model of a point on the line is given by its variances σ_s^2 and σ_t^2. σ_s^2, the variance in direction of the line, can be assumed to be identical to the rounding error, whereas σ_t^2, the variance orthogonal to the line (which is, in fact, the one actually describing the quality of a line point) can be derived by applying the law of error propagation to equation 5.17 using $\mathbf{C}_{km}$ from equation 5.18 to describe the stochastic properties of k and m:

$$\sigma_t^2 = \sigma_m^2 + s^2 \cdot \sigma_k^2 \approx \hat{\sigma}_o^2 \cdot (\frac{1}{l_l} + s^2 \cdot \frac{12}{l_l^3}) \tag{5.19}$$

As the line segment is represented by its end points $\hat{\mathbf{P}}_a$ and $\hat{\mathbf{P}}_e$, we are interested in the stochastic properties of these points. In the line co-ordinate system, these points are characterized by $s \approx \pm l/2$. Replacing s in equation 5.19 by that approximation yields

$$\mathbf{C}_{st} = \begin{pmatrix} \sigma_s^2 & \sigma_{st} \\ \sigma_{st} & \sigma_t^2 \end{pmatrix} \approx \begin{pmatrix} \sigma_s^2 & 0 \\ 0 & \frac{4 \cdot \hat{\sigma}_o^2}{l_l} \end{pmatrix} \tag{5.20}$$

for the variance-covariance matrix $\mathbf{C}_{st}$ of the co-ordinates (s, t) of one of these points in the line co-ordinate system. By applying the law of error propagation to equation 5.16, the variance-covariance matrix $\mathbf{C}_{rc}$ of the sensor co-ordinates (r, c) of a line end point, we get:

$$\mathbf{C}_{rc} = \mathbf{R}_\varphi \cdot \mathbf{C}_{st} \cdot \mathbf{R}_\varphi{}^T \tag{5.21}$$

Using equation 5.21, the correlations of the co-ordinates of both end points of a line segment are omitted. If the line segments are combined after the approximation procedure, the vertices of the resulting polygon have to be estimated from an intersection of adjacent line segments l_1 and l_2. The variance-covariance matrix $\mathbf{C}_v$ of such a vertex can be computed from the variance-covariance matrices $\mathbf{C}_{rc}^1$ and $\mathbf{C}_{rc}^2$ of the end points of l_1 and l_2 according to [Fuchs, 1998]

$$\mathbf{C}_v = \frac{1}{4} \cdot (\mathbf{C}_{rc}^1 + \mathbf{C}_{rc}^2) \tag{5.22}$$

Representation of edges: Depending on the application, there are several ways of representing edges. Not all of the aggregation steps described in the previous paragraphs have to be performed. Edges can be represented by

- Edge elements: In some top-down applications, it is not necessary to perform edge tracking: for instance, in order to match a model edge with the image contents, all edge elements parallel to the transformed model edge can be assigned to the model edge and used for the determination of the actual model edge.

- Straight line segments: Especially for the extraction of man-made objects, the individual line segments as the result of the splitting algorithm can be considered to be individual straight image lines. Such a line segment is then described by its beginning and end points $\hat{\mathbf{P}}_a$ and $\hat{\mathbf{P}}_e$, respectively, and by the variance-covariance matrices of these points (equations 5.20 and 5.21).

- Polygons: As stated above, the polygon vertices have to be computed from intersection of the line segments. The polygon can either be described by the set of line segments it consists of, the variance-covariance matrices being replaced by those from equation 5.22. Alternatively, it can be represented by a list of its vertices and their variance-covariance matrices from equation 5.22.

- Cubic splines: Cubic splines can be represented as polygons, the polygon vertices being replaced by the spline node points.

5.1.5 Feature adjacency graphs

Evidently, by just describing the image by an unstructured cluster of features, a considerable amount of information would be thrown away. Considering the topological relations of the image features for object reconstruction might be convenient in many cases. As experience shows, errors due to noise are contained in the extracted features. If the topological relations of the features were known, they could be used to perform consistency tests in order to eliminate segmentation errors. That is why it is a good idea to also extract the topological relations between the features to create a *feature adjacency graph (FAG)*. Depending on the feature types considered in the image model, there are several ways how this can be accomplished. In [Fuchs, 1998], in the course of polymorphic feature extraction (cf. section 5.1.2), points, edges, and homogeneous regions are extracted, and the feature adjacency graph is derived from the classified image (figure 5.3) by an analysis of the exoskeleton: all features having common region borders are supposed to be neighbours (figure 5.8).

Figure 5.8: Left: the results of classification: Point regions P_i, edge regions L_i, homogeneous regions S_i, and their borders. Right: the corresponding feature adjacency graph. The full lines describe direct neighbourhood relations, the broken lines indirect ones. According to [Fuchs, 1998].

In our variation of polymorphic feature extraction, we restrict ourselves to the extraction of points and edges. We want to obtain a representation of the topological properties based on the geometrical distribution of these features. The basis of our approach to topology extraction is a 2D Delaunay triangulation of the extracted points and the edge vertices. To the Delaunay triangulation the extracted image edges are added as constraints, i.e. the vertices of the polygons have to be connected by edges (labelled as "image edges") in the graph in the way described in section 2.2.2. Obviously, the new graph describes the original image better than the Delaunay graph (figure 5.9; [Halmer et al., 1996, Mischke and Rottensteiner, 1997]).

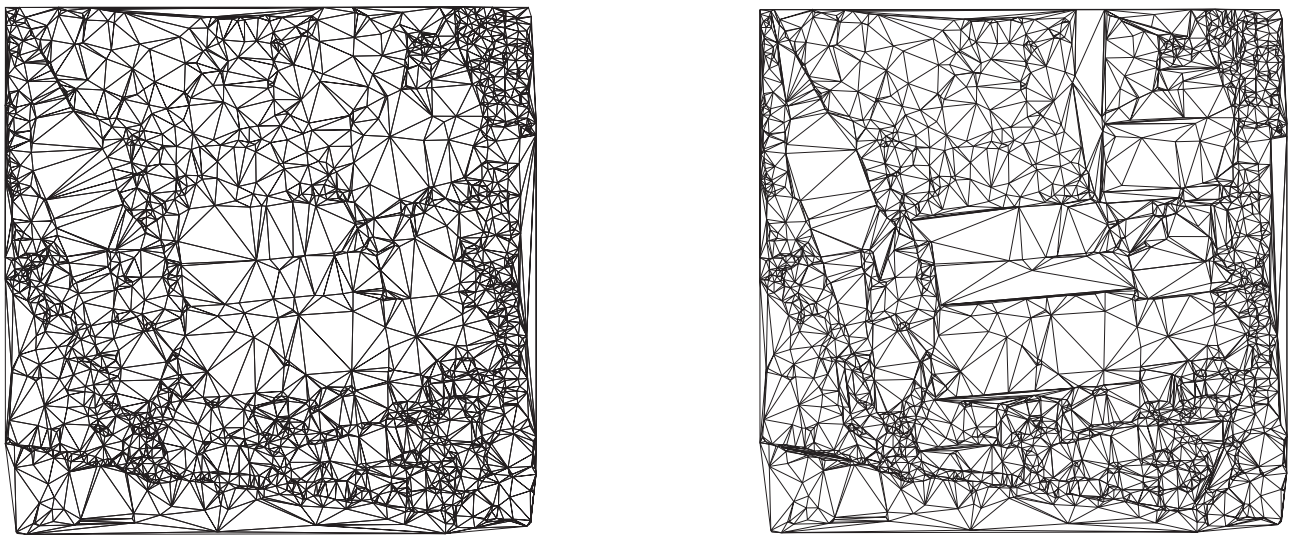


Figure 5.9: Graphs derived from the features extracted in figure 5.3. Left: Delaunay triangulation of the extracted points and the line vertices. Right: Constrained triangulation: Line vertices are connected by edges.

5.2 Matching techniques

As stated in the introductory section of chapter 5, the term *matching* means the establishment of a relation between two or more images and/or the object to be reconstructed. Depending on the geometric models used for the mapping functions describing this relation, two cases can be distinguished (figure 5.10):

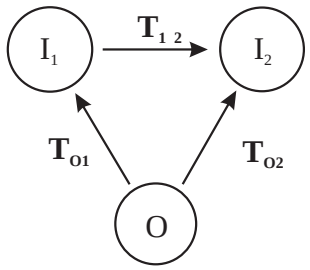


Figure 5.10: Image matching vs. object reconstruction. Taken from [Lang and Förstner, 1998].

1. *Image matching:* Image matching techniques directly relate the images I_1 and I_2 by a mapping function $\mathbf{T}_{12}$. In this case, the object model is implicitly contained in the formulation of $\mathbf{T}_{12}$ which will be very complex in general but can be locally approximated by an affine transformation if the object surface can be assumed to be smooth, thus yielding a reduction of computational complexity compared to object space matching. However, in the presence of occlusions the smoothness assumption will be hurt, and image matching algorithms will face problems [Lang and Förstner, 1998, Gülch, 1994].

2. *Object space matching:* In this case, the object O is reconstructed directly by inverting the mapping transformations $\mathbf{T}_{O1}$ and $\mathbf{T}_{O2}$. An explicit model for the object O has to be available, and the problem is solved by establishing correspondences between image features and features of the object model. Object space matching techniques have the advantage that they are closer to physical reality so that they may be capable of handling occlusions if sophisticated object models are used. On the other hand, the number of parameters to be estimated in the inversion process can be very high in some cases [Lang and Förstner, 1998]. For instance, the object can be modelled by a 2.5D raster DEM $Z(X,Y)$ using the finite element method (cf. section 2.2.1) and by a grey level distribution $G(X,Y)$ in object space, i.e. a digital orthophoto. Then both the grid points of $Z(X,Y)$ and the grey levels of the digital orthophoto $G(X,Y)$ can be determined simultaneously by demanding the grey levels in the images to be identical to the grey levels on the object at corresponding points, thus $g_i\{u_i[X,Y,Z(X,Y)],v_i[X,Y,Z(X,Y)]\} = G(X,Y)$ for all images i, where $\{u_i[X,Y,Z(X,Y)],v_i[X,Y,Z(X,Y)]\}$ are the perspective equations 4.24. However, this approach leads to an enormous amount of unknowns [Wrobel, B., 1987, Heipke, 1990].

From another point of view, matching algorithms can be characterized by the *image model* they use [Gülch, 1994]:

- *Raster based matching:* These algorithms use a raster representation of the image, i.e. they try to find a mapping function between image patches by directly comparing the grey levels or functions of the grey levels. They offer the highest potential for accuracy, but they are very sensitive to occlusions [Ackermann, 1984, Lang and Förstner, 1998, Gülch, 1994]. Raster based image matching techniques will be discussed in section 5.2.1.

- *Feature based matching:* In this case, a symbolic description of the images is derived first by extracting salient features from the images using some feature extraction operator (cf. section 4.24). After that, corresponding features from different images have to be found under certain assumptions regarding the local geometry of the object to be reconstructed and the mapping geometry. These algorithms

are more flexible with respect to surface discontinuities and requirements for approximate values than raster based techniques [Krzystek, 1995, Gülch, 1994]. Feature based image matching will be discussed in section 5.2.2.

- *Relational matching:* Relational or structural matching techniques rely on the similarity of topological relations of features which are stored in feature adjacency graphs rather than on the similarity of grey levels or the similarity of point distributions. This is motivated by the fact that topology is an image property which is invariant under perspective transformation. Matching of relational descriptions or relational matching thus is a very powerful concept which might work in rather general cases. That is why it can be applied for object recognition tasks [Burge and Burger, 2000]. However, its computational complexity is very high because it leads to rather complex search trees, especially in the applications we are considering in this work [Vosselman, 1995].

5.2.1 Raster based matching techniques

Raster based matching techniques use the grey levels themselves or functions of the grey levels as the description of the images. It is the goal to estimate the parameters of the transformation $\mathbf{T}_{12}$ between two images (figure 5.10). One of the images is chosen to be the reference image (the *template*), its grey levels will be denoted by g_R; the other image will be called search image and its grey levels denoted by g_S. $\mathbf{T}_{12}$ can be a dense disparity map. In this case, the whole images will be used as reference and search images, respectively. However, raster based matching can also be applied to small image patches only. In this case, the reference image is either a synthetical one derived from a given target description or a small image patch in a region surrounding a feature point previously extracted by a feature extraction algorithm or provided by a human operator. The search image is then an image patch centered at approximate values.

5.2.1.1 Cross correlation

Cross correlation is an algorithm for the location of corresponding image patches based on the similarity of grey levels. A reference point is given in the reference image, and its co-ordinates are searched for in the search image. For that purpose, the reference image is moved in the search image, and the position of maximum similarity of grey levels is searched for. At each position of the reference image in the search image, a similarity value, e.g. the cross correlation coefficient $k_{R,S}$ of the grey levels is calculated [Rottensteiner, 1993].

$$k_{R,S}(\Delta r, \Delta c) = \frac{\sum\limits_{r_R,c_R} [g_R(r_R, c_R) - \bar{g}_R] \cdot [g_S(r_R + \Delta r, c_R + \Delta c) - \bar{g}_S]}{\sqrt{\sum\limits_{r_R,c_R} [g_R(r_R, c_R) - \bar{g}_R]^2 \cdot \sum\limits_{r_R,c_R} [g_S(r_R + \Delta r, c_R + \Delta c) - \bar{g}_S]^2}} \tag{5.23}$$

In equation 5.23, $\bar{g}_R$ and $\bar{g}_S$ denote the arithmetic mean grey level in the reference image and the part of the search image which is covered by the reference image, respectively. All sums are to be taken over all pixels of the reference image. In order to speed up computation, equation 5.23 can be re-written as follows using the shorthands $g_R = g_R(r_R, c_R)$, $g_S = g_S(r_R + \Delta r, c_R + \Delta c)$ and $k_{R,S} = k_{R,S}(\Delta r, \Delta c)$:

$$k_{R,S} = \frac{\sum g_R \cdot g_S - \sum g_R \cdot \sum g_S}{\sqrt{[\sum g_R^2 - (\sum g_R)^2] \cdot [\sum g_S^2 - (\sum g_S)^2]}} \tag{5.24}$$

In equation 5.24, the expressions $\sum g_R$ and $[\sum g_R^2 - (\sum g_R)^2]$ are constant, and most terms of $\sum g_S^2$ and $\sum g_S$ remain so, too. Thus, as the reference image moves over the search image, only the sums of of one row/column of g_S and g_S^2, respectively, have to be added to $\sum g_S$ and $\sum g_S^2$, and the sums another row/column have to be removed. Only $\sum g_R \cdot g_S$ has to be fully re-computed for every new position of the reference image. The position of the point corresponding to the reference point is given by the position of the maximum of the

similarity measure, the result only being accepted if a certain threshold is met. In case the cross correlation coefficient $k_{R,S}$ is used, the threshold can be chosen rather easily (e.g. $k_{R,S} < 0.7$) because that coefficient is bounded by -1 and 1. Thus the position of that point can be determined with a resolution of 1 pixel. From equation 5.23 it can further be seen that $\mathbf{T}_{12}$ in this case just comprises two shifts:

$$\mathbf{T}_{12} : \begin{pmatrix} r_S \\ c_S \end{pmatrix} = \begin{pmatrix} r_R \\ c_R \end{pmatrix} + \begin{pmatrix} \Delta r_{max} \\ \Delta c_{max} \end{pmatrix} \tag{5.25}$$

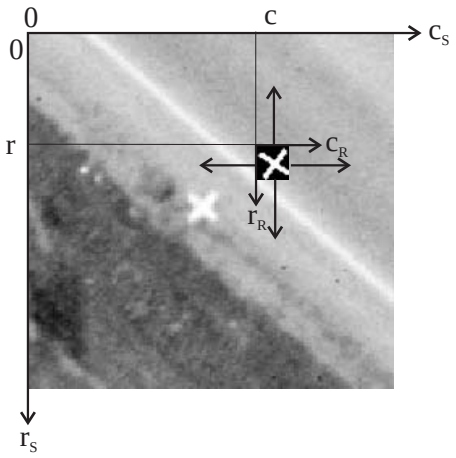


Figure 5.11: Cross correlation.

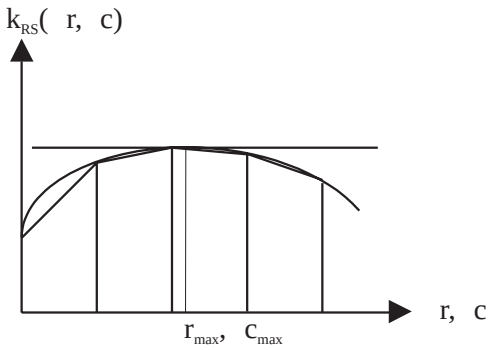


Figure 5.12: Subpixel estimation.

Subpixel estimation can be performed by approximation of the correlation coefficients $k_{R,S}$ by a second-order polynomial function:

$$k_{R,S} = a_0 + a_1 \cdot r + a_2 \cdot c + a_3 \cdot r \cdot c + a_4 \cdot r^2 + a_5 \cdot c^2 \tag{5.26}$$

The coefficients a_i in equation 5.26 can be determined from the correlation coefficients in a small, e.g. 3×3 pixels^2 window by least squares adjustment using equation 5.26 as the observation equations for $k_{R,S}$. The subpixel shift vector $(\Delta r_{max}, \Delta c_{max})^T$ can then be computed as the position of the maximum of the polynomial function from its differentials:

$$\begin{pmatrix} \frac{\partial k_{R,S}}{\partial r} \\ \frac{\partial k_{R,S}}{\partial c} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} + \begin{pmatrix} 2 \cdot a_4 & a_3 \\ a_3 & 2 \cdot a_5 \end{pmatrix} \cdot \begin{pmatrix} \Delta r_{max} \\ \Delta c_{max} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{5.27}$$

The accuracy of subpixel estimation was empirically determined to be about $\pm$ 0.2 - $\pm$ 0.3 pixels for targeted points.

Cross correlation is tolerant with respect to the quality of the approximations for the shifts; problems may arise with repeating patterns because in this case there will exist different positions with high similarity. The algorithm will also fail if the images are not similar, which will happen in several cases [Rottensteiner, 1993]:

- The approximations are too bad. New approximations have to be provided by a human operator in order to overcome that problem. In some cases measurement can also be aborted. The important issue is to detect failure at all.
- $\mathbf{T}_{12}$ from equation 5.25 cannot be used because the image is rotated or scaled. If the rotation is unknown, search can be repeated with successively rotated reference images until the correct position has been found. In the second case, better approximations for the scale are required.

- The image patch is too big. In this case, search has to be repeated with a reduced patch size.
- There are occlusions. This is one of the greatest problems concerned with raster based matching techniques and can hardly be handled by them. In applications where occlusions are to be expected it is better to use feature based matching techniques.

5.2.1.2 Least Squares Matching (LSM)

Least Squares Matching is the most accurate image matching technique [Ackermann, 1984, Prinz, 1995]. Just as cross correlation, it is based on the similarity of grey levels. However, $\mathbf{T}_{12}$ is more complex in this case. Assuming the image patch in question to be small and the object surface to be smooth, $\mathbf{T}_{12}$ can be modeled to be an affine transformation $\mathbf{T_a}$ from equation 4.26:

$$\mathbf{T}_{12}(r_R, c_R) = \begin{pmatrix} r_S \\ c_S \end{pmatrix} = \mathbf{T_a}(r_R, c_R) \tag{5.28}$$

If the parameters c_{ij} of $\mathbf{T}_{12}$ are known exactly and if there are no radiometric errors, the grey levels of the search image and the reference image transformed by $\mathbf{T}_{12}$ according to equation 5.28 are assumed to be identical up to randomly distributed noise n [Ackermann, 1984]:

$$g_R(r_R, c_R) + n = g_S(r_S, c_S) = g_S[\mathbf{T}_{12}(r_R, c_R)] \tag{5.29}$$

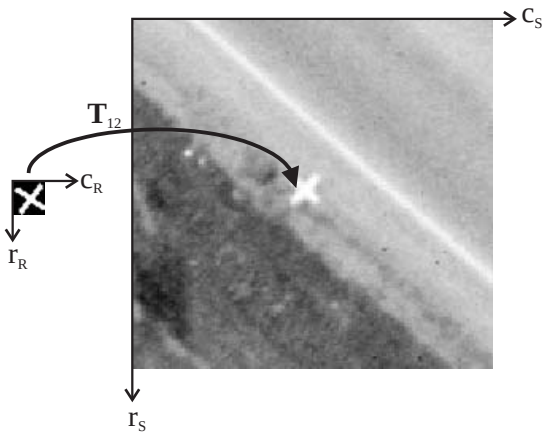


Figure 5.13: Least Squares Matching.

The reference image is transformed to the search image using approximate values c_{ij}^0 for the parameters of $\mathbf{T}_{12}$. Due to radiometric errors and to the fact that the parameters of $\mathbf{T}_{12}$ are not known exactly, there will be grey level differences between the two images. It is the basic idea of LSM to estimate the parameters of $\mathbf{T}_{12}$ from these observed grey level differences by a least squares adjustment. Equation 5.29 is linearized using the approximate values c_{ij}^0 and setting $c_{ij} = c_{ij}^0 + \delta c_{ij}$. Approximating the first derivatives of the grey levels of the search image $\partial g_S/\partial r$ and $\partial g_S/\partial c$ by the differences of grey levels of the reference image $\Delta g_{R_r}, \Delta g_{R_c}$ gives the observation equations which can be established for each pixel [Prinz, 1995]:

$$n = \sum_{i,j} (\Delta g_{R_r} \cdot \frac{\partial r_S}{\partial c_{ij}} + \Delta g_{R_c} \cdot \frac{\partial c_S}{\partial c_{ij}}) \cdot \delta c_{ij} - \{g_R(r_R, c_R) - g_S[\mathbf{T}_{12}^0(r_R, c_R)]\} \tag{5.30}$$

Least squares adjustment using the observation equations 5.30 delivers the unknown corrections δc_{ij} for the transformation parameters. Due to the non-linearity of equation 5.29, least squares adjustment has to be performed iteratively using the corrected transformation parameters of the previous adjustment as approximations

for the successive one. The iteration process implies the calculation of $g_S[\mathbf{T}_{12}^0(r_R, c_R)]$ by resampling. The mathematical model can be expanded to other transformations $\mathbf{T}_{12}$ and to handle also radiometric parameters. However, introducing radiometric parameters might prevent the iterative process from convergence. Thus, it is better to apply radiometric corrections before stepping into the LSM algorithm [Prinz, 1995]. Due to the great number of observations (one observation per pixel of the reference image), LSM is the most accurate image matching technique. The transformation parameters can be estimated with an accuracy of up to $\pm$ 0.1 pixels [Ackermann, 1984]. However, it is very sensitive with respect to the quality of the approximations. They have to be known already with an accuracy of a few pixels. For that reason, LSM is often used to improve accuracy as a final step following the application of another matching technique, e.g. cross correlation, for establishing the approximations $\mathbf{T}_{12}^0$. Just as cross correlation, LSM will fail if the two image patches are not similar; it is especially confronted with problems if there are occlusions due to surface discontinuities. Additional care has to be taken on the determinability of the parameters $\hat{a}_{ij}$. Analyzing equation 5.30, it can be seen that pixels in homogeneous regions with Δg_{R_r} and Δg_{R_c} being close to zero do not deliver any information for the determination of the parameters. Some parameters cannot be determined if there are certain dependencies between the grey level differences. For instance, the rotations cannot be determined for circular targets.

LSM can be expanded to more than two images. In case N images are used, $\frac{N\cdot(N-1)}{2}$ transformations $\mathbf{T}_{ij}$ with $1 \leq i < j$ can be established because the grey levels of each image pair can be compared. However, these transformations are not independent. Again, one image, e.g. I_1, is chosen to be the reference image. Now all transformations $\mathbf{T}_{ij}$ with $1 < i < j$ can be expressed as $\mathbf{T}_{ij} = \mathbf{T}_{1j}(\mathbf{T}_{1i}^{-1})$ which leads to more complex normal equation systems. In addition, geometrical constraints can be included in the mathematical model [Baltsavias, 1991, Tsingas, 1992].

5.2.2 Feature based matching techniques

Feature based matching techniques do not use the grey levels themselves as the description of the images but rather an abstract image representation derived from a feature extraction algorithm. The form of the description as well as the type of features used for matching (points, image edges, homologous patches) depend on the task to be solved. In any case, the correspondence problem between features from different images has to be solved. Again, the parameters of the transformation $\mathbf{T}_{12}$ between two images (figure 5.10) are to be estimated in order to solve this problem.

Having detected features in two or more images, correspondences between homologous features from different images have to be found. Under the assumptions made in section 5.2.1.2, the affine transformation (equation 5.28) can be used again as the mathematical model for the transformation $\mathbf{T}_{12}$. However, the image patches used for feature based matching are usually larger (e.g. 200 $\times$ 200 pixels^2) than those used for raster based techniques; on the other hand, the result is not a single point or a raster of displacement vectors, but a set of homologous points / image lines from which a set of 3D points / lines in object space can be computed by spatial intersection. These 3D features can be used to derive a description of the object surface.

A useful approach for establishing correspondences is given by the hypothesis generation and verification paradigm which splits the task into two sub-tasks [Krzystek, 1995, Tang et al., 1996, Lang and Förstner, 1998]:

1. The generation of correspondence hypotheses: Find initial matches between features from different images.

2. The evaluation of hypotheses: Eliminate false hypotheses under the assumption of a transformation $\mathbf{T}_{12}$ (which implicitly contains a model of the object surface). Only hypotheses consistent with $\mathbf{T}_{12}$ will be accepted.

Note that the results of step 2 will still contain errors. However, feature based matching techniques can deliver very dense 3D data, e.g. a very dense point cloud. Filtering techniques have to be applied to reduce the point

density, but also to reduce the influence of remaining false matches and thus increase both accuracy and reliability of the results. For instance, [Krzystek, 1991] estimates the grid heights of a DEM from the 3D points by a finite element approach (cf. section 2.2.1). Even though the features and thus also the 3D points are less accurate than points determined interactively by stereoscopic plotting using an analytic plotter, the resulting DEM is shown to be of a comparable quality because the grid points of the DEM are estimated from up to 10 times more 3D points than in the interactive case [Krzystek and Wild, 1992].

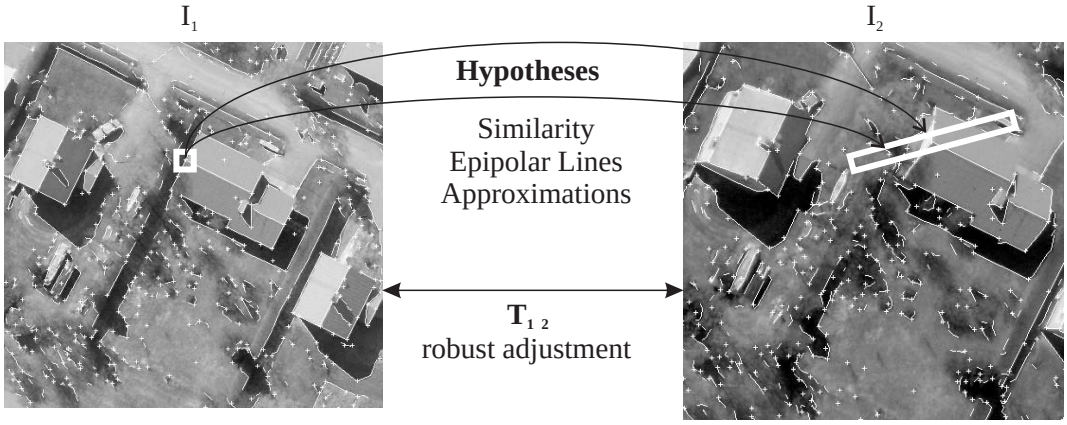


Figure 5.14: The principle of feature based matching using two images I_1 and I_2: first, correspondence hypotheses are generated making use of approximations, geometrical constrains and similarity measures to reduce search space. After that, these hypotheses are evaluated by robust estimation using a mathematical model for the (local) transformation $\mathbf{T}_{12}$ between both images.

In the following sections we will assume that the problem of finding homologous image patches has already been solved in advance. As the size of the object usually exceeds the patch size for matching, the object is split into object patches which one after the other provide homologous regions of interest in the images. The problem of providing approximate values for finding homologous image regions will be discussed in sections 5.3 and 5.4.

5.2.2.1 Generation of hypotheses of correspondence

If no other information were available, each feature from image I_1 could correspond to each of the features from the other image I_2. As this obviously would lead to too great a number of possible matches, methods for the restriction of the number of possible matches have to be searched for. First of all, the number of possible matches can be considerably reduced by geometric constraints:

- *Epipolar constraints:* A feature in I_2 homologous to a certain feature in image I_1 has to be situated along the epipolar line (cf. section 4.5.3). Thus if the orientation parameters of the images were known exactly, only points along the epipolar line would be possible candidates. If the orientation parameters are only known approximately, search space is still restricted to a band centered at the epipolar line, its width depending on the quality of the approximations. If only two images are available and if the orientation parameters are known sufficiently well, the images can be resampled so that the rows of the sensor coordinate system are identical to the epipolar lines. Using these *epipolar images* can speed up matching considerably because the number of parameters of $\mathbf{T}_{12}$ as well as the dimension of search space is reduced. The feature extraction techniques described in section 5.1 can be simplified considerably because only the grey level variations in direction of the image rows have to be considered [Krzystek, 1991]. However, epipolar images cannot be used if more than two images are to be used for matching simultaneously.

- *Approximations for the object:* Approximate values reduce search space along the epipolar lines. They can either be specified by the user, e.g. in the form of limits for the object's distance from the images, or

they can be derived automatically by hierarchical procedures (cf. section 5.4.1).

In this way, the search space for one feature is reduced to a more or less small rectangular area (right image in figure 5.14). The remaining possible hypotheses have to be assigned weights based on a measure of similarity S of the corresponding images. Depending on the contents of the symbolic image information and on the feature type, there are several possibilities for assigning weights:

- *Similarity of grey levels:* The cross correlation coefficient of the grey levels from small image patches (equation 5.23) can be used as a similarity measure for point features from different images, e.g. [Tsingas, 1992]. In a similar way, the grey levels on both sides of two possibly homologous image lines can be compared, e.g. [Baillard et al., 1999].

- *Similarity of neighbourhood:* A similarity measure can also be derived from the topological relations [Lang, 1999].

- *Similarity of curvature of image edges:* Edges can be parameterized by their length l, and a measure for similarity can be derived from a comparison of the curvatures $\Psi_1(l), \Psi_2(l)$ [Li et al., 1991].

In any case, the number of hypotheses can further be reduced by excluding hypotheses failing to meet a certain threshold for their similarity measure.

For each possible feature match, i.e., for each pair of features $(f_1, f_2)_i$ of images I_1 and I_2, a cost function C_i can be evaluated which might look as follows [Lang, 1999]:

$$C_i = \frac{1}{2} \cdot (C_{1i}^I + C_{2i}^I) + C_i^G + C_i^S \tag{5.31}$$

In equation 5.31, C_{1i}^I and C_{2i}^I are cost functions describing the quality of the features f_1 and f_2 in the images I_1 and I_2, respectively. Each of these terms describe, for instance, the distinctness of an extracted point. It is closely related to texture strength W (cf. section 5.1.2) because a feature is the more reliably located the stronger the texture is in its neighbourhood: $C_{ji}^I = C_{ji}^I(W)$. The term C_i^G in equation 5.31 measures how well the geometrical constrains fit with respect to the feature pair. It depends from the distance of feature f_2 in image I_2 from the epipolar line corresponding to feature f_1 in image I_1. Finally, the term C_i^S evaluates the similarity measure S, thus $C_i^S = C_i^S(S)$.

As a result, we get a set of correspondence hypotheses i weighted by C_i. However, these hypotheses are not consistent for two reasons:

1. The hypotheses set still contains false matches

2. The hypotheses set still contains multiple matches, i.e., a feature from image I_1 can still have two or more possibly homologous features from image I_2.

These problems have to be solved using knowledge about the object. Additional information can also be provided by using more than two images. For instance, if two features from different images have been found to be possible matching candidates, an epipolar line corresponds to each of the features in a third image, thus, the position of a corresponding feature in the third image is given by the intersection of these epipolar lines. This method can be used for further feature verification by adding additional terms to the cost function in equation 5.31 [Faugeras et al., 1992, Lang, 1999].

5.2.2.2 Evaluation of hypotheses of correspondence

As stated above, those correspondence hypotheses which contradict to the mathematical model of the image transformation $\mathbf{T}_{12}$ have to be eliminated. As the epipolar lines are related to the observation co-ordinate systems $(u_i, v_i), i \in \{1, 2\}$ of the images I_i and the features are originally extracted in the sensor co-ordinate systems $(r_i, c_i), i \in \{1, 2\}$, the features first have to be transformed to the observation co-ordinate systems using equation 4.26 (cf. section 4.2.2). In addition to the feature co-ordinates, the variance-covariance matrices describing the stochastic properties of the features have to be transformed, too, using the law of error propagation. Assuming that homologous image points are searched for, the parameters of $\mathbf{T}_{12}$ can be estimated by least squares adjustment (cf. section 4.1). Each correspondence hypotheses i gives two observation equations based on the mathematical model in equation 5.28. The co-ordinates of the feature from image I_2 are the observations (u_{i2}, v_{i2}), and the parameters c_{kl} of $\mathbf{T}_{12}$ are the unknowns to be determined in the adjustment. The co-ordinates (u_{i1}, v_{i1}) of the feature from image I_1 are considered to be free of errors:

$$\begin{aligned} E(u_{i2}) = \quad u_{i2} + \tilde{v}_{u_{i2}} = \quad c_{00} + c_{11} \cdot u_{i1} + c_{12} \cdot v_{i1} \\ E(v_{i2}) = \quad v_{i2} + \tilde{v}_{v_{i2}} = \quad c_{01} + c_{21} \cdot u_{i1} + c_{22} \cdot v_{i1} \end{aligned} \tag{5.32}$$

The weight p_i of the observation equation pair 5.32 for point i is chosen to be indirectly proportional to the cost function C_i, thus, using an appropriate a priori r.m.s. error of the weight unit σ_o:

$$p_{u_{i2}} = p_{v_{i2}} = p_i = \frac{\sigma_o^2}{C_i} \tag{5.33}$$

The more distinct two features from different images are, the better they fulfil the geometrical constraints, and the more similar they are in the two images, the greater is their influence on the determination of the parameters c_{kl} in equation 5.32. However, as we have seen in section 4.1, least squares adjustment is not robust with respect to gross errors in the data. In order to eliminate false matches, ML-type robust estimation techniques based on re-weighting the observations can be applied (cf. section 4.1.1.1). Matching problems are characterized by the fact that the number of gross errors in the data (i.e., the number of false matches) is relatively high. [Förstner, 1986] gives a procedure which successively uses two weight functions (equation 4.14) in order to eliminate false matches. First, a similar weight function $w(d_{i,k})$ as the one from equation 4.15 is used for observation pair i in iteration $k + 1$:

$$w_1(d_{i,k}) = \frac{1}{\sqrt{1 + d_{i,k}^2}} \tag{5.34}$$

where $d_{i,k}$ is the normalized discrepancy of observation i in iteration k (equation 4.12, cf. section 4.1.1.1). Weight function w_1 from equation 5.34 has rather heavy tails and thus does not eliminate gross errors too soon. According to [Förstner, 1986] and [Krzystek, 1995], 3 to 6 iterations are performed using w_1 which should eliminate large gross errors. After that, global convergence of the process should be guaranteed, and some (2-5) iterations are performed using another weight function w_2:

$$w_2(d_{i,k}) = e^{-d_{i,k}^2} \tag{5.35}$$

w_2 has steeper slopes than w_1, it will eliminate observations more easily and is used to eliminate "smaller" gross errors. Robust estimation should not be used too rigorously in the context described above if the mathematical model, e.g. an affine transformation $\mathbf{T}_{12}$ is not a very exact one: the image patches are greater than those used for LSM, and the object is not necessarily planar. However, this shows also one of the strengths of feature based matching compared to raster based techniques: it works even if the mathematical model is not rigorously fulfilled. Thus, feature based matching is, to a certain extent, less sensitive to occlusions than raster based matching.

After the iteration scheme has been finished, there may still be multiple matches contained in the data. They can be resolved by accepting the match with the best fit, i.e. the one receiving the smallest residuals. Remaining

false matches will be propagated to object space, and robust estimation techniques as have to be applied in object space in order to eliminate false 3D features [Gülch, 1994]. This can, for instance, be performed by robust estimation of the grid points of a 2.5D DEM (cf. section 2.2.1) based either on the finite elements model and the application of the iteration scheme using weight functions w_1 and w_2 from equations 5.34 and 5.35 [Krzystek, 1991] or on linear prediction, e.g. [Kraus and Pfeifer, 1998], using a weight function similar to the one in equation 4.17.

Comparing feature based matching techniques to raster based matching techniques, some differences can be observed [Gülch, 1994]:

- Raster based matching techniques offer a higher potential for accuracy, especially LSM. The accuracy of feature based matching is limited by the accuracy of feature extraction.
- Raster based matching techniques directly use the actual observations (i.e. the grey levels), which is more satisfactory from a theoretical point of view, even though the selection of a reference image with grey levels free from errors appears to be somewhat critical. This theoretical problem can be overcome by applying matching a second time, this time the second image being the reference image, and averaging results.
- Feature based matching techniques are less sensitive with respect to the quality of approximate values. LSM has a very small radius of convergence. Even though it is greater for cross correlation, the similarity condition has to be fulfilled, which might be critical in the presence of large rotation and / or scale differences between the two images.
- Feature based matching techniques are also less sensitive with respect to occlusions because the mathematical model $\mathbf{T}_{12}$ needs not necessarily to be fulfilled exactly.

5.3 Image pyramids

By the expression *image pyramid*, usually the representation of a digital image in different resolution levels is meant [Kropatsch, 1991, Ackermann and Hahn, 1991]. Figure 5.15 shows an example for such an image pyramid of an aerial image. The idea of image pyramids as a stack of digital images actually depicting the same scene with decreasing resolution is closely related to the concept of scale space [Yuille and Poggio, 1986], in which the scale s is introduced as an additional (continuous) dimension of a digital image. A continuous reduction of image resolution with respect to the original image can be achieved by smoothing the image with a Gaussian smoothing kernel $\mathbf{G}_\sigma$, the scale parameter corresponding to the standard deviation σ. In this context, image pyramids in the sense of the first definition are just a logarithmically sampled version of scale space (cf. figure 5.16).

Figure 5.15: A regular image pyramid of an aerial image.

The base level I_0 (the level having the best resolution) is the image at the original resolution. An image at level I_{i+1} is created from image I_i, the one at the next lower level, by two steps [Kropatsch, 1991]:

1. Apply a low pass filter, e.g. a Gaussian filter $\mathbf{G}_\sigma$ to the image at level I_i: $I_{i+1} = \mathbf{G}_\sigma \star I_i$. The standard deviation σ controls the degree of smoothing. In many applications, $\mathbf{G}_\sigma$ is approximated by an $m \times m$ binomial filter. Typically, $m = 3$ is chosen, which corresponds to $\sigma = 0.71$. The size of the filter $\mathbf{G}_\sigma$ is called *reduction window* [Kropatsch et al., 2000a].

2. Select the surviving elements of I_{i+1}, e.g. every n^{th} pixel. The *reduction factor* n is the second control parameter of image pyramid creation. Typically, $n = 2$ is chosen.

The *structure* of an image pyramid is determined by the "horizontal" neighbourhood relations within the levels of the pyramid and by the "vertical" *father-son* relations between adjacent levels [Kropatsch et al., 2000a]: Each cell (except those at the base level) of an image pyramid has a set of children (sons) at the level below which provide input to the cell. On the same level it has a set of neighbours, and each cell (except those at the apex) has one parent cell (father) at the level above (figure 5.16). For image pyramids in the sense of a stack of images at decreasing resolution, the cells are identical to the pixels, and all the above relations are given implicitly by the pixel indices in the grey level matrices, the reduction factor n and the reduction window which depends on the selection of σ. These image pyramids are often called *regular image pyramids* [Kropatsch et al., 2000a].

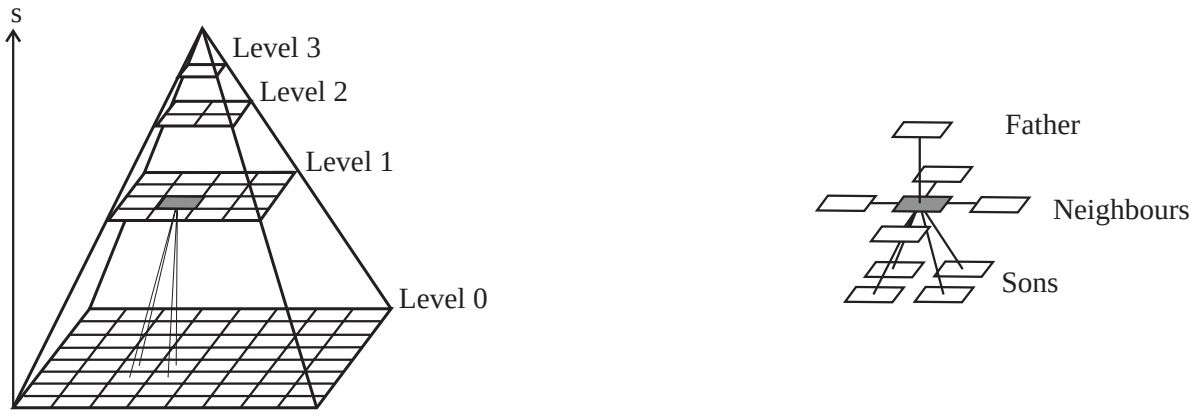


Figure 5.16: Structure of a regular image pyramid. Left: the discrete levels and scale s; right: a particular cell. Adopted from [Kropatsch et al., 2000a].

A more general view on pyramids is given by the concept of *irregular pyramids*. Irregular pyramids differ from regular ones by several items [Kropatsch et al., 2000a]:

1. The pixels are not necessarily the cells of irregular pyramids. The content of a cell can be anything from an original grey level via some more general numeric property to symbolic information, e.g. a cell can represent an image edge extracted by some feature extraction algorithm.

2. Corresponding to the generalization of the cell contents, the neighbourhood and father-son relations are generalized: all these relations are contained in a graph. The nodes of that graph correspond to the cells of the pyramid, and the edges correspond to any of the relations. The edges are labelled so that the neighbourhood relations can be distinguished from the vertical ones.

3. The creation process described above has to be replaced by a more general one. Still, the principle that smoothing is followed by a selection of the surviving cells can be applied. However, new criteria have to be found. For instance, an edge pyramid could be created by first assigning an "edge strength" to each cell (e.g. the length) and then checking which edges have a "stronger" neighbour. After that, the locally "strongest" elements are selected to be survivors and thus become cells in the adjacent upper pyramid levels, the "weaker" neighbours being assigned to that cell as children.

As already indicated by the example, features extracted from a digital image can be represented in irregular pyramids in a hierarchical way, and irregular pyramids of features could be referenced to as "feature pyramids". However, in digital photogrammetry, the term *feature pyramid* is used in another sense: It means that feature extraction is applied to all levels of an iconic (regular) pyramid, and all features extracted from the same level of the image pyramids represent a level of the feature pyramid [Krzystek, 1991, Ackermann and Hahn, 1991]. In that sense, these feature pyramids can be seen as a hierarchical representation of the digital image on a symbolic level, too. The difference between such feature pyramids and irregular pyramids is mainly given by the fact that feature pyramids contain less relations than irregular pyramids: Even if the neighbourhood relations within each level are represented in feature adjacency graphs, there are no explicit vertical relations contained in feature pyramids. Thus, in irregular pyramids, it is clear which features from level i are sons of a certain feature in level $i+1$, which is not the case for feature pyramids because the features from adjacent pyramid levels were created independently from each other. This can be seen as a drawback of feature pyramids, because in irregular pyramids this information is preserved. On the other hand, this can also be seen as an advantage. Smoothing in the creation of an iconic pyramids causes feature extraction to detect the most apparent image structures in the upper pyramid levels. Even if the reduction process in the creation of an irregular pyramid favours the "strongest" features and even though these pyramids preserve the vertical relations, there is no guarantee that all relevant features which would be detected by feature extraction in the corresponding image of the iconic pyramid are actually available at a certain pyramid level of the irregular one. In our work, we use the concept of feature extraction applied successively to all levels of iconic pyramids (cf. section 5.4).

	high resolution	low resolution
data amount	huge	small
details	rich and many	very few
overview	bad	good
precision	high	low

Table 5.1: Qualities of images at different resolutions. Taken from [Kropatsch et al., 2000a].

Image pyramids combine the advantages of both high and low resolutions of digital images (cf. table 5.1) without increasing the demand for disk space too much: even with the smallest possible reduction factor $n = 2$, the amount of data storage is only increased by 30%. The lower levels of an image pyramid provide detailed information, but a great amount of data, whereas the higher levels contain less information but give an overview and require a smaller amount of data. Iconic pyramids are essential for visualization of and navigation in great image data files: an appropriate level can be selected for visualization depending on the size of the window which is to be displayed in a certain region of a screen. If a user wants to see another region of the lowest image level on the screen, he or she can first view at a greater region of the image in a lower resolution (which can be visualized in a considerably shorter time), mark the region to be inspected more closely, and then display that region at full resolution.

In the context of automation of photogrammetric plotting, image pyramids are used for coarse-to-fine (hierarchical) methods in object reconstruction and image matching. Image pyramids have the following merits [Ackermann and Hahn, 1991, Kropatsch et al., 2000a]:

- The influence of noise is reduced in the lower resolution images by smoothing.
- In the low resolution images, the regions of interest for correspondence analysis in levels of higher resolution can be found at low cost because irrelevant details are no longer available there.
- This reduces computational cost as the divide-and-conquer principle can be applied: in high resolution images, the region of interest can be split into several patches which can temporarily be handled individually.

In section 5.4, the application of image and feature pyramids for object reconstruction is described in closer detail.

5.4 A general framework for object surface reconstruction

As we have already seen in chapter 2, depending on the class of object which is to be reconstructed, different object modelling schemes have to be used. In section 4.5 we discussed that, again depending on the object class, but also on the level of detail which is to be achieved, different sensors have to be used, different configurations of photographs are necessary. In dependence on the modelling technique, the results have to be represented in different ways. From this point of view, photogrammetric plotting tasks which are to be automated can be categorized as follows:

- *Small scale topographic mapping:* Techniques for the automatic reconstruction of object surfaces for small scale topographic mapping have already been described in sections 5.2.1 and 5.2.2. Most algorithms aim at a 2.5D DEM (section 2.2.1) which can be computed from a 3D point set. The object surface can be assumed to be smooth. In order to improve the quality of the DEM, surface discontinuities can be searched for in the DEM grid [Wild and Krzystek, 1996, Rieger et al., 1999], but at that scale this is only important in rugged terrain. The image configuration is usually close to the stereo case with two images covering each surface patch (figure 5.17). The images are analogous aerial images which have to be scanned off-line.

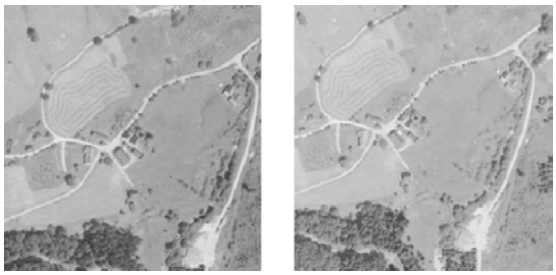


Figure 5.17: Two homologous image patches used for small scale topographic mapping.

- *Large scale topographic mapping:* A 2.5D DEM will no longer be sufficient to describe the earth surface, especially if we think of built-up areas. 3D modelling techniques useful for man-made objects have already been described in section 2.3. If topographic objects such as houses are to be reconstructed, a point cloud will no longer be sufficient to derive an object description; this task requires also surfaces and lines as well as topological information. As it is obvious from the image patches in figure 5.18, the object surface can no longer be assumed to be smooth so that we have to deal with occlusions and surface discontinuities. This is the reason why two images might no longer be sufficient for the automation of this task. Again, the images are scanned analogous aerial photographs.

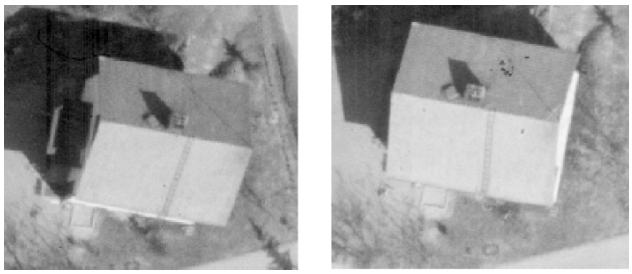


Figure 5.18: Two homologous image patches from a large-scale photo flight.

- *Close range and industrial applications:* Depending on the actual object class, close range and industrial applications have to face different problems. Flat building facades can more or less be treated similar to DEMs in small scale topographic mapping; in other applications, surface discontinuities and occlusions must be handled. Especially in architectural and industrial applications, non-stereo (bundle) configurations are typically used in order to increase accuracy and reliability (section 4.5). Some classes of objects might be better described by points and others by lines (e.g. figure 5.19). In most cases, 3D representations will be required.

Figure 5.19: Three images for the reconstruction of a car's door.

Considering the great variety of objects which can be reconstructed by photogrammetric techniques, it appears to be impossible to find one algorithm capable of handling all possible cases. However, it does make sense to investigate common strategies for object reconstruction and common structures for object modelling, in order to create a framework for object surface reconstruction into which specific algorithms can be inserted easily rather than to try to find one single algorithm capable of handling all possible cases. We have developed such a framework which is characterized by the following items [Rottensteiner, 1996, Rottensteiner, 1998]:

- *Hierarchical object reconstruction:* In order to make the algorithms work with quite coarse an approximation, a hierarchical coarse-to-fine strategy using image pyramids has to be applied.
- *Multi-image solutions:* Two images might be sufficient for topographic applications, but in case of tasks dealing with more complex shapes, occlusions will enforce the usage of more, e.g. four or six, images, [Fuchs, 1998, Faugeras et al., 1992, Baillard et al., 1999].
- *Feature based matching:* As already stated in section 5.2.2, feature based matching is more flexible with respect to occlusions and surface discontinuities than raster based matching.
- *Consistent object modelling:* A consistent way of object modelling in the reconstruction process provides a powerful tool for treating different applications in a similar way in a framework based on the hypotheses generation / verification paradigm.
- *Considering object space:* On the basis of these object models, object space is to be integrated directly into the reconstruction process. We do not aim at pure image-to-image matching, thus directly seek for image-to-object correspondences.
- *Robust hybrid photogrammetric adjustment:* All the above items are realized by integration of the system *ORIENT* for hybrid photogrammetric adjustment (cf. chapter 4). The mathematical model of that adjustment system provides a very sophisticated model of sensor geometry (i.e., the mapping functions, cf. section 4.2) and can be directly used for the formulation of the object models. Robust estimation is used to eliminate false matches.
- *Hybrid representation of results:* In chapter 2, we have seen that different classes of objects have to be represented in different ways. By supporting different object modelling schemes, this fact is considered in our framework. Note that the final representation of the object may differ from the way it is represented in the reconstruction process.

In this section, we first want to describe this framework in detail in section 5.4.1. After that, an example will be given showing the application of that framework to DEM generation for topographic mapping in section 5.4.2. Another example for the application of the framework is given by the automated modules of our system for semi-automatic building extraction. It will be described in chapter 6.

5.4.1 Hierarchical object reconstruction

Figure 5.20 presents the work flow of hierarchical object reconstruction in our framework. The goal of object reconstruction is the derivation of a description of the object surface. We support two modelling schemes in our framework:

1. 2.5D grid DEMs (cf. section 2.2.1)
2. Boundary representation of solid objects (cf. section 2.3.1)

The input data are:

1. Image pyramids of all digital images available in a photogrammetric block
2. The orientation parameters of all images in the block
3. Coarse approximate values for the object parameters.

The approximate values can be obtained in various ways, for instance:

- An approximate 2.5D DEM grid derived from the control and tie points of the photogrammetric block, or the terrain can be assumed to be a flat surface represented by an average terrain height provided by the user.
- A region of interest for the reconstruction of more complicated objects provided by an object detection algorithm or by human interaction.
- Approximate parameters of a solid object in boundary representation provided by the user in a semi-automatic environment.

It is the idea of hierarchical (coarse-to-fine) techniques in matching and object reconstruction to apply matching algorithms to one image pyramid level after the other, starting from the upper level. The results of pyramid level $i + 1$ are used to reduce search space in level i so that the whole region of interest can be split into smaller sub-regions which can be reconstructed independently from each other at level i. This strategy is iteratively applied until the base level of the pyramid ($i = 0$) is reached, e.g. [Ackermann and Hahn, 1991, Krzystek, 1991, Tsingas, 1992, Gülch, 1994]. This strategy is applied in our framework as depicted in figure 5.20 independently of the object class. Starting at pyramid level $i = N$, task-dependent object reconstruction techniques are iteratively applied. Post-processing might be necessary to estimate the parameters of the actual surface representation in pyramid level i from the reconstruction results. These results are back-projected to the digital images in the next lower level using the orientation parameters. Depending on the way the object is represented, back-projection may mean

- The back-projection of the grid-mesh points of a 2.5D grid DEM to derive homologous image patches
- The back-projection of the vertices of a solid object in boundary representation.

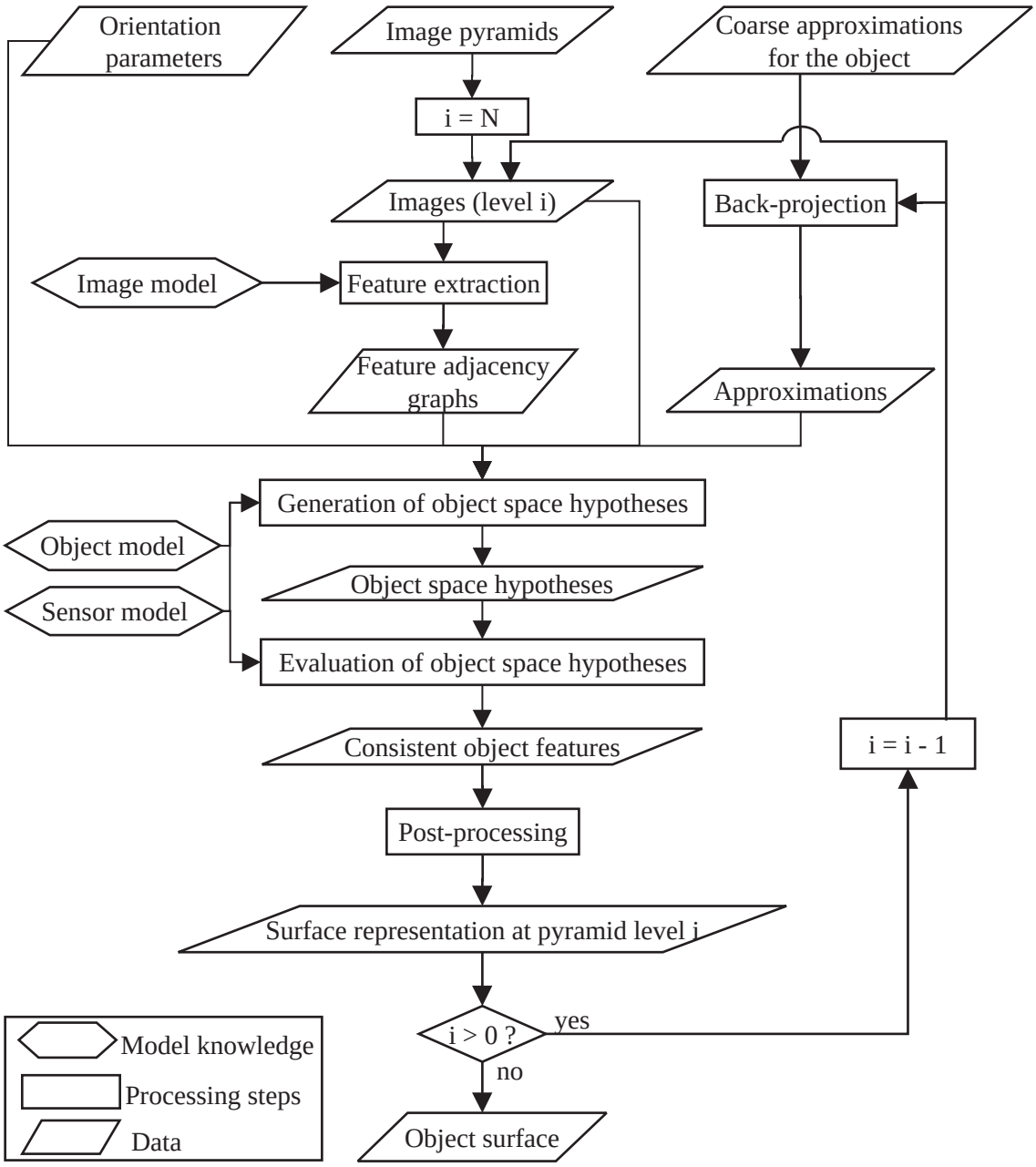


Figure 5.20: A flowchart of hierarchical object reconstruction

The process is terminated as soon as the lowest level of the image pyramids (i.e. the level with the highest spatial resolution; $i = 0$) has been reached. The surface representation at the base level of the pyramid is identical to the final representation of the object surface (figure 5.20).

Feature based matching techniques are applied for object reconstruction at a given pyramid level i. In our framework, these matching techniques rely on the data structures, the mathematical model, and the parameter estimation procedures of the hybrid photogrammetric adjustment system *ORIENT* which has been described in chapter 4. The matching process is guided by *model knowledge* in three ways (figure 5.20):

1. *Image model:* A model for the properties of digital images is required as the basis for feature extraction. We use the image model described in section 5.1 because we use a variation of the concept of polymorphic feature extraction in our framework. The image model also contains the properties of the abstract image representation, i.e., the properties of both the feature adjacency graphs and the (point and line) features.

2. *Sensor model:* A model for the geometrical properties of the imaging sensor is required both for the reduction of search space by epipolar lines in the way described in section 5.2.2.1 and for back-projection in the iteration process. In our framework, the sensor model is given by the mathematical model of *ORIENT* (cf. section 4.2). Each digital image corresponds to an "observation room" in the *ORIENT* data base (cf. section 4.3) by which access to the mapping parameters (the orientation parameters) is given. Note that there is no restriction with respect to the observation type (as long as it is compatible with the idea of a digital image being attached to it): in our framework all imaging sensors supported by *ORIENT* may be used. The current restriction to perspective photos is only one of implementation because the C++ interface for the mapping functions (cf. section 4.3.4) is not yet available for other observation types.

3. *Object model:* In contrast to the feature based matching techniques described in section 5.2.2, in our framework, the object model is made explicit in object space. The object model is a domain-specific one, but a unique way of representing model knowledge about the object in the reconstruction process is used, which makes our approach a framework rather than a collection of independent object reconstruction tools.

As depicted in figure 5.20, three major steps have to be performed at each pyramid level:

1. *Feature extraction:* points and edges have to be extracted from the images. The specific properties of our variation of the concept of polymorphic feature extraction as it is used in our framework (cf. section 5.1) will be described in section 5.4.1.1.

2. *Correspondence analysis:* Correspondences between homologous features from multiple images and / or between image and object features have to be found. The core of our framework is the representation of the object in the reconstruction process. It will be described in section 5.4.1.2. After that, section 5.4.1.3 is dedicated to correspondence analysis itself.

3. *Post processing:* The parameters of the final object representation have to be estimated from the results of the correspondence analysis. This step may include filtering in case the data delivered from the previous steps are distributed too densely. It is especially necessary in case the final representation differs from the object representation in the matching process. As stated above, we support 2.5D raster DEMs and boundary representations as final object representations in our framework. In the first case, post processing consists in the estimation of the heights of the grid points from a 3D point cloud. In the second case, no post-processing is required.

5.4.1.1 Feature extraction

The first step required at each pyramid level is feature extraction. In our variation of polymorphic feature extraction, points and edges are simultaneously extracted from the digital images. The concept of [Fuchs, 1998] as described in section 5.1.2 is simplified slightly in several ways:

1. The Gaussian filter matrix $\mathbf{L}$ in equation 5.5 is replaced by a binomial filter of size $m \times m$.

2. The threshold W_{min} for texture strength is chosen as a multiple of the median of W (equation 5.9).

3. Only edges and points, no homogeneous regions are extracted.

4. With respect to edges, no difference is made between edges in the sense of grey level discontinuities and "lines", i.e. linear features of some (small) spatial extent in gradient direction.

5. The feature adjacency graph is created as described in section 5.1.5. The neighbourhood relations are extracted depending on the sensor co-ordinates of the extracted points and edge vertices, not on an analysis of the extracted regions in texture classification.

The parameters for our implementation of the algorithm are:

- The size m_e of the binomial filter $\mathbf{L}$ in equation 5.5 for texture classification. Default: $m_e = 3$
- The size m_p of the binomial filter $\mathbf{L}$ in equation 5.5 for re-computing the texture strength for the pixels classified as "point" pixels. Default: $m_p = 3$
- The multiplication threshold j for the texture threshold in equation 5.9. Default: $j = 2.5$
- The threshold Q_{min} for "roundness" Q. Default: $Q_{min} = 0.7$
- The minimum distance $d_{p_{min}}$ between adjacent points in non-maxima suppression. This means that a pixel in a point region is supposed to be a relative maximum if there is no greater value of W in a pixel inside a square window of side length $s = (2 \cdot d_{p_{min}} + 1)$ pixels centered at that pixel. Default: $d_{p_{min}} = 3\ [pixel]$
- The minimum length l_{min} of an edge pixel chain. All edge pixel chains shorter than l_{min} are discarded. Default: $l_{min} = 3\ [pixel]$
- The maximum distance $d_{l_{max}}$ of an edge element from the approximating polygon in edge thinning (cf. section 5.1.4). Default: $d_{l_{max}} = 1\ [pixel]$

For each image on each pyramid level, a feature adjacency graph (FAG) is derived by the feature extraction module. The FAG describes the image on a symbolic level. The feature adjacency graph contains nodes of two types

1. The extracted image points with both their geometrical and their stochastic properties as described in section 5.1.3
2. The polygon vertices of the extracted image edges with both their geometrical and their stochastic properties as described in section 5.1.4.

It also contains two types of edges:

1. "plain" edges connecting two nodes of which at least one is an extracted image point
2. "polygon" edges connecting two adjacent polygon vertices of an extracted image edge.

5.4.1.2 Mathematical formulation of the object models

It has been stated above that model knowledge about the object is to be used in correspondence analysis for eliminating false matches. In our framework, we use explicit object models in object space, and the way these models are formulated is closely connected to parameter estimation. Remembering the discussion about different modelling techniques for topographic objects in chapter 2, we can make the trivial notion that all objects have surfaces, and it is the surfaces we want to reconstruct. If we have a closer look at that, we see that in all modelling techniques, the object surface at least consists of a set of faces and their mutual relations. These relations can be given implicitly or explicitly:

1. Implicit relations: The 2D domain of object surface is split into rectangular regions. The surface patch of each region corresponds to a face. The region borders are given by grid lines of the object co-ordinate system. There is no object edge at these face borders. This is the way the surface is represented in 2.5D grid-based DEMs (cf. section 2.2).

2. Explicit relations: In this case, the object has edges and corners. The object edges are the intersections of two neighbouring object faces, and corners are the intersection points of three or more such faces. Boundary representation (cf. section 2.3.1) is an appropriate modelling technique for modelling such objects, even though the object needs not necessarily be a closed solid.

We use the concept of surface observations of *ORIENT* to represent the faces of our object models (cf. section 4.2.3). Each face in the surface model corresponds to a an observation room of type "*GESTALT*" in the *ORIENT* data base (cf. section 4.3.3). Each face equation is formulated in the (local) observation co-ordinate system (u, v, w). In this co-ordinate system, the face is described by a set of coefficients a_{jk}, b_{ik}, c_{ij} in the three equations 4.27, and the local co-ordinate system is described by the position of the exterior reference point $\mathbf{P}_0$, the three rotation angles (ω, ϕ, κ) and the three diagonal elements of the mirror matrix $\mathbf{M}$, i.e. (m_u, m_v, m_w) (cf. section 4.2). Assigning a point to a face (be it an object corner or just a "bulk point" giving support to that face) is performed by inserting that point into the corresponding *GESTALT* room, which means that in a later adjustment, one of the observation equations 4.27 will be inserted. As stated above, (corner) points and edges may be a part of the object model or not. An object edge is, basically, just considered to be the intersection of two faces. If a point is assigned to an object edge, this means that the point will be inserted into both *GESTALT* rooms corresponding to the two faces intersecting at the object edge, which means that two observation equations 4.27 will be inserted into a later adjustment for that point. In adjustment, no more assumptions about the edge are required. The object parameters (i.e., the surface parameters a_{jk}, b_{ik}, c_{ij}, the rotation angles (ω, ϕ, κ), the object co-ordinates of $\mathbf{P}_0$ and, eventually, the object co-ordinates of object vertices) are to be determined in hybrid adjustment. In order to make them determinable, image features have to be assigned to object features in the sense described above, i.e., they are inserted into the respective *GESTALT* rooms, and additional observation equations are inserted into adjustment. The connection between image and object features is given by the point identifier only.

For each of the faces of its object model, a specific application has to

1. Create a *GESTALT* room.
2. Define the observation co-ordinate system by creating and initializing the according parameter rooms (*ROT*, *ERP*, *ADP*; cf. section 4.3.3) and by correctly initializing the references in the *GESTALT*'s header. The application is free to let several rooms refer to the same transformation parameters.
3. Select which of the three observation equations 4.27 shall be inserted into adjustment for all points contained in the *GESTALT* room.
4. Select an appropriate subset of all possible coefficients (either a_{jk}, b_{ik}, or c_{ij}, depending on which observation equation has been chosen) for that *GESTALT* and insert the according points into the room containing the additional parameters. Alternatively, the face can be declared to be symmetrical to another one.
5. Insert all those points of the object model which are assigned to the face into the *GESTALT* room.
6. Insert these points into the reference system, too.

As the object usually consists of more than one face, a system of *GESTALT* rooms is created. This is usually done in a setup phase before correspondence analysis is started, but it is also possible to create these rooms when hypotheses are generated if this makes sense for an application. Additional information (object corners, object edges) has to be considered, too, if necessary, but as described above, with respect to adjustment, these items are not as relevant as the faces are (leaving aside the fact that, using information about object edges or corners, the number of observation equations is increased). In some cases it is also necessary to define *parameter observations* for regularization if singularities are known to appear in adjustment.

Figure 5.21 shows two examples for the way object models can be handled by specific applications:

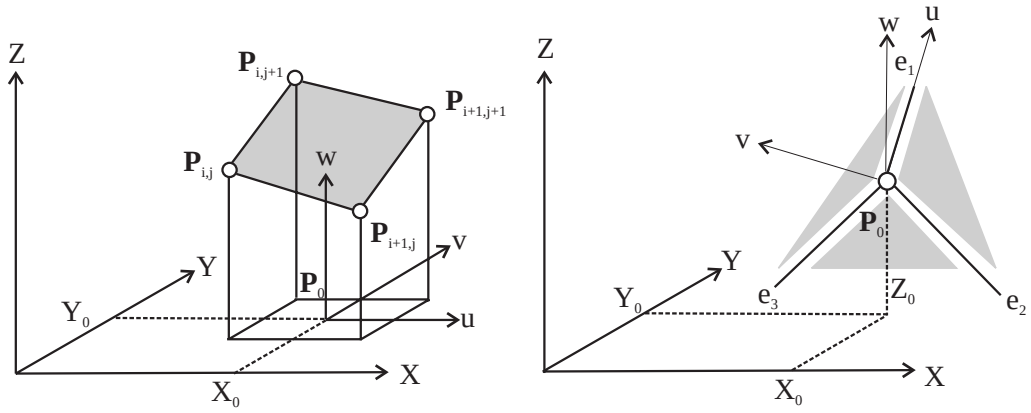


Figure 5.21: Left: A grid mesh in a 2.5D DEM represented by face ε and the observation co-ordinate system (u, v, w) of the *GESTALT*. Right: A trihedral corner as the intersection of three surfaces ε_1, ε_2, and ε_3 and the observation co-ordinate system. The surfaces mutually intersect at the edges e_1, e_2, and e_3.

- One mesh in a 2.5D raster DEM (left part in figure 5.21; cf. also figure 2.3). In order to represent a grid mesh like this, an application has to

  1. Create the *GESTALT* room corresponding to the face ε.
  2. Insert the points $\mathbf{P}_0$, $\mathbf{P}_{i,j}$, $\mathbf{P}_{i,j+1}$, $\mathbf{P}_{i+1,j}$, and $\mathbf{P}_{i+1,j+1}$ into the reference system. Initialize the co-ordinates of the grid points. The reference point $\mathbf{P}_0$ is assigned the co-ordinates of the centre of the grid mesh. This point will be kept fixed in adjustment, thus it will be de-activated in the reference system. The observation co-ordinate system is thus defined to have its origin in the centre of the grid mesh.
  3. Create a room of rotation parameters, initialize the rotation angles by 0 and deactivate them. The observation co-ordinate system is thus defined to be parallel to the object co-ordinate system.
  4. Create a room of additional parameters for a ficticious observation of w (third line in equation 4.27). Insert the coefficients c_{00}, c_{10}, c_{01}, and c_{11} into that room. Thus, for each point contained in the *GESTALT* room, an observation equation

     $$\tilde{v}_w = Z - Z_0 + c_{00} + c_{10} \cdot (X - X_0) + c_{01} \cdot (Y - Y_0) + c_{11} \cdot (X - X_0) \cdot (Y - Y_0) \qquad (5.36)$$

     will be inserted into adjustment. The formulation of the surface equation corresponds to equation 2.1 in section 2.2.1.
  5. Insert the grid points $\mathbf{P}_{i,j}$, $\mathbf{P}_{i,j+1}$, $\mathbf{P}_{i+1,j}$, and $\mathbf{P}_{i+1,j+1}$ into *GESTALT* room. Thus, for each of these points, the Z co-ordinate is determinable from the corresponding observation equation 5.36.
  6. Create a room of type "observed control points" and insert the grid points into that room. Deactivate the Z-equation of these points. These control point observations are required to prevent adjustment from becoming singular in the planimetric co-ordinates of the grid points as they are not determined by surface observations.

  The unknowns to be determined in adjustment are the object co-ordinates (X, Y, Z) of all grid points and the surface parameters c_{00}, c_{10}, c_{01}, and c_{11}. The transformation parameters are kept fixed.

- A trihedral corner (right part in figure 5.21). In the example in the figure, three surfaces (ε_1, ε_2 and ε_3) intersect at a corner point. The surfaces intersect at three edges e_1, e_2 and e_3, which again intersect at the corner point. The object edge e_1 is supposed to be horizontal, and the surfaces ε_1 and ε_2 have the same tilt. In [Lang, 1999], such corners are reconstructed in order to be used for automatic building extraction. In this case, it makes sense to define the local co-ordinate system to be centered at the corner point, thus

$\mathbf{P}_0$ is the corner point. Again, axis w of the observation co-ordinate system is vertical, but u is defined as being in direction of the horizontal edge. Thus, the rotation angles ω and ϕ are 0, but κ is defined by the direction of e_1. In order to represent such trihedral corner in our modelling scheme, an application has to:

1. Insert $\mathbf{P}_0$ into the reference system and assign approximate values to its co-ordinates.
2. Create a room of rotation angles *ROT*. Initialize the angles by $(\omega, \phi, \kappa)_0^T = (0, 0, \kappa_0)^T$, where κ_0 is derived from the approximate direction of e_1. Also create a room of observed rotation angles. Activate the observations for ω and ϕ. These observations will keep the (u, v) plane of the observation co-ordinate system horizontal.
3. Create *GESTALT* rooms for ε_1, ε_2, and ε_3. Declare $\mathbf{P}_0$ and *ROT* to be the exterior reference point and the rotation angles for all these surfaces.
4. Create rooms of additional parameters for surfaces ε_1 and ε_3. ε_1 contains the u-axis. In the observation equation system, it is described by just one parameter, i.e. its tilt in direction v. Thus, the additional parameter room will be an additional parameter room for w equations, and only one coefficient $c_{01}^{\varepsilon_1}$ has to be inserted into it. ε_3 is supposed to be in the (v, w)-plane. Its additional parameter room will be an additional parameter room for u equations, but no additional parameter will be added to it. The mirror matrix elements are declared to be $+1$.
5. As ε_2 is supposed to be symmetrical to ε_1 with respect to the (u, w)-plane of the observation co-ordinate system, it is declared to share its surface coefficients (its additional parameters) with ε_1. The elements of its mirror matrix are declared to be $(m_u, m_v, m_w)^T = (+1, -1, +1)^T$. Thus, for a point $\mathbf{P}$ on one of the three surfaces, one of the following three observation equations will be inserted into adjustment, using the shorthand $\mathbf{p}_R = (u_R, v_R, w_R)^T = \mathbf{R}^T \cdot (\mathbf{P} - \mathbf{P}_0)$ for the right-hand side in equation 4.22:

$$\begin{aligned} \varepsilon_1 &: \tilde{v}_w = w_R + c_{01}^{\varepsilon_1} \cdot v_R \\ \varepsilon_2 &: \tilde{v}_w = w_R - c_{01}^{\varepsilon_1} \cdot v_R \\ \varepsilon_3 &: \tilde{v}_u = u_R \end{aligned} \tag{5.37}$$

The object parameters of a trihedral corner are the co-ordinates of $\mathbf{P}_0$, the three rotation angles (ω, ϕ, κ) of which ω and ϕ are fixed by observations of a great weight[1], and the tilt $c_{01}^{\varepsilon_1}$ of the symmetrical planes ε_1 and ε_2. Assigning a point to $\mathbf{P}_0$ means that the point will be inserted into all three *GESTALT* rooms, and three observation equations 5.37 will be inserted into adjustment. Assigning a point to one of the edges means that the point will be inserted into the two *GESTALT* rooms corresponding to the surfaces intersecting at the edge, so that two observation equations 5.37 will be inserted into adjustment for that point.

To sum up, in order to create a domain-specific object model, an application of the framework has to decompose the object into faces, and it has to map the specific properties of these faces to *ORIENT* surfaces (*GESTALT* rooms). As *ORIENT* gives an application every freedom to choose the observation co-ordinate system and the parameterization of the surface equations, by carefully designing the system of *GESTALT* rooms the application ends up with having to determine a minimum set of parameters, as we have seen in the example of the trihedral corner. In section 5.4.2 and in chapter 6, more elaborated examples for domain specific object models will be given.

5.4.1.3 Correspondence analysis

Having detected features in two or more images, the correspondence problem has to be solved. On the contrary to section 5.2.2 where the relation between two images was given by a functional model $\mathbf{T}_{12}$ for a transformation between two images, in our more general approach we will seek correspondences in object space and

[1] This is due to the fact that in *ORIENT* it is not possible to declare a single rotation angle constant in adjustment. An application either has to determine all three angles or none (cf. section 4.3.1).

replace $\mathbf{T}_{12}$ by the perspective transformations of all images because we consider these methods to be more flexible with regard to handling occlusions and surface discontinuities. Instead of the functional model $\mathbf{T}_{12}$, a domain-specific local model of the object surface is provided using the modelling principle described in the previous section, and false correspondences are detected from bad fits to that model in object space. The selection of an appropriate local surface model is either done implicitly by the task which has to be solved, or it is the user who selects a model from a task-specific model data base if such data base is available. In some applications, model selection can be automated.

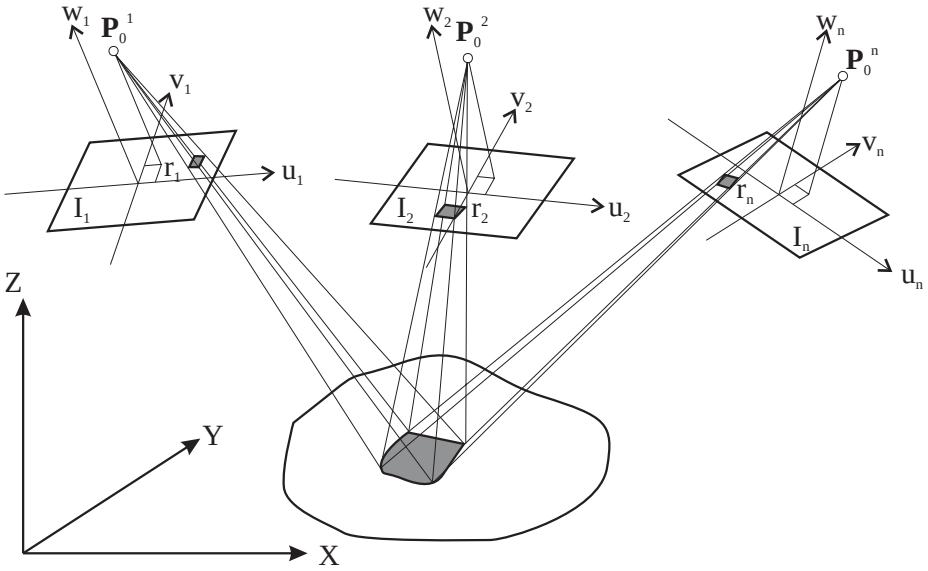


Figure 5.22: A patch of the object (grey) is projected to n images, where it corresponds to the (thus: homologous) image patches $r_i, i \in \{1..n\}$.

We assume that from the approximate values, a region of interest can be derived in object space. Large objects can be split into patches which are treated independently from each other in correspondence analysis. The results of the individual patches have to be united in post-processing. Each patch is projected into all digital images where it corresponds to homologous image patches $r_i, i \in \{1..n\}$ (figure 5.22; n is the number of digital images the patch is visible in). Before correspondence analysis starts, the object models are generated by providing a system of *GESTALT* rooms in the way described in section 5.4.1.2. From the approximate values, approximate parameters of the object models can be estimated. In the correspondence analysis, the same principle of hypotheses generation / verification as described in section 5.2.2 can be used, with some modifications:

1. The generation of correspondence hypotheses makes use of approximate values and the orientation parameters in order to reduce search space. Depending on the object class, different algorithms can be used for that step. In any way, two cases can be distinguished which are both covered by our framework:

   (a) *Data driven analysis:* Correspondences between feature points / edges from different images have to be found using the techniques described in section 5.2.2. In contrast to the procedure described there, in our framework, a correspondence is propagated to object space where it is assigned to a feature of a local object model. It is not known a priori how many object features will be found.

   (b) *Model driven analysis:* The model features have a specific meaning, and it is exactly the model features that are searched for in the images. No image-to-image correspondences have to be searched for. A correspondence is either a correspondence between an extracted feature point and a vertex of the object model or between an extracted image edge and an edge of the object model.

2. The evaluation of these hypotheses is performed under the assumption of the local surface model in object space: Only hypotheses consistent with the model will be accepted. Using the uniform mathematical formulation of these local surface models described in section 5.4.1.2 makes this step independent of the object class.

In the following paragraphs, we will describe these steps of our framework in general terms. An example for the application of the framework for a specific object class will be given in section 5.4.2.

Data driven generation of correspondence hypotheses: A data driven process starts with searching for homologous features in all images. It is an expansion of the hypotheses generation task as it is described in section 5.2.2 to the multi-image case. It may involve a search for corresponding points and/or corresponding edges, depending on the task which is to be solved. For instance, starting from a correspondence of two point features in two images, the corresponding point in a third image has to be located at or at least close to the intersection point of two epipolar lines [Faugeras et al., 1992]. The sensor model is exploited by using the epipolar constraint for restricting search space. No semantic meaning is attributed to the features. In some applications, the actual structure of the object might not be known exactly in advance, but has to be selected from a set of possible object shapes depending on what has been detected in the images.

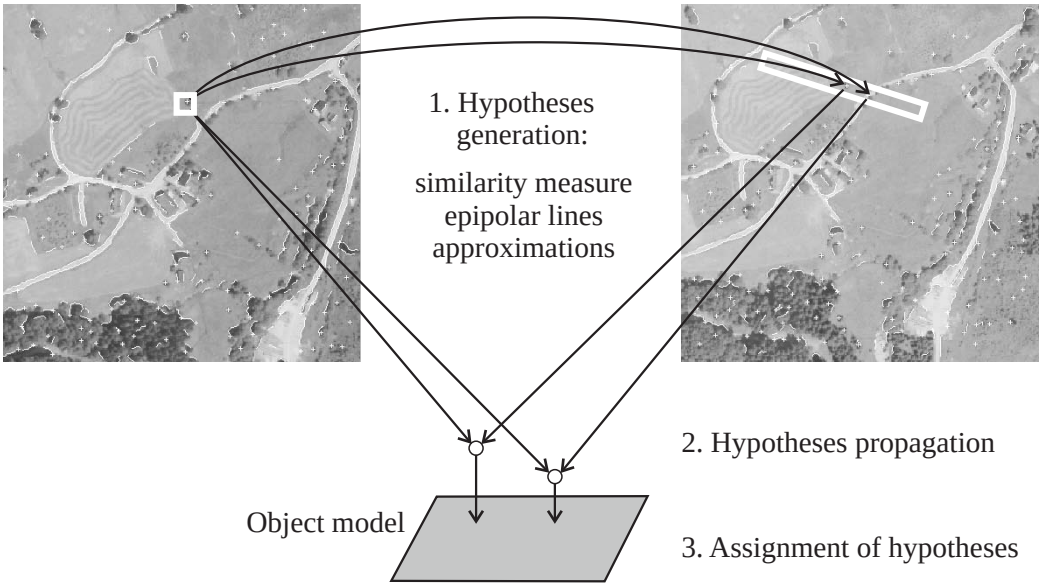


Figure 5.23: Data driven generation of correspondence hypotheses (two homologous image patches only).

After the initial image-to-image correspondences have been established, in contrast to the procedure described in section 5.2.2, in our framework, these correspondences are propagated to object space where they are assigned to features (faces, edges, or points, in the sense of section 5.4.1.2) of a local object model. In adjustment, correspondence is declared by assigning identical identifiers to points in different observation rooms. Let us have a look at two examples:

1. homologous image points assigned to an object face, and
2. homologous image edges assigned to an object edge.

Homologous image points: If *homologous image points* are assigned to an object face, n feature points $\mathbf{p}_i, i \in \{1..n\}$ from n images are supposed to correspond to the same object point $\mathbf{P}$. This means that:

1. A new object point $\mathbf{P}_{ID}$ with identifier ID has to be created in the reference system.

2. For each of the feature points $\mathbf{p}_i$, a point is inserted into the *ORIENT* observation room corresponding to image i. All these points receive the same identifier ID as the object point. The camera co-ordinates (u_i, v_i) of the points can be derived from the sensor co-ordinates (r_i, c_i) of the features by applying the affine transformation from equation 4.26. The stochastical properties of the image points can be derived from the variance-covariance matrix $\mathbf{C}_{rc_i}$ by error propagation. Note that in the current version of *ORIENT*, the off-diagonal elements of $\mathbf{C}_{uv_i}$ are neglected. The a priori variances of the image points (i.e. the diagonal elements of $\mathbf{C}_{uv_i}$) are additionally multiplied by a function of the cost function C_i (e.g., from equation 5.31) in order to give "similar" hypotheses a greater a priori influence than less similar ones.

3. A point with identifier ID is inserted into the *GESTALT* room corresponding to the object face the hypotheses is assigned to. The a priori r.m.s. error of this observation is a measure for how rigidly the object model has to be fulfilled. This r.m.s. error has to be specified by the user.

Thus, for this kind of hypotheses, the following observation equations are inserted into adjustment:

1. Two camera co-ordinate observations per image point (equations 4.24), the weight depending on $\mathbf{C}_{uv_i}$ and the cost function C_i.

2. One surface equation (one of equations 4.27).

In addition to the surface parameters, three new unknowns have to be determined in adjustment (the point's object co-ordinates).

Homologous image edges: If *homologous image edges* are assigned to an object edge, n edge features from n images are supposed to correspond to the same object edge e. As described in section 5.1.4, image edges are represented by the polygon vertices and their variance-covariance matrices $\mathbf{C}_{rc}$, the elements of $\mathbf{C}_{rc}$ depending on the length of the polygon edge: vertices being connected by long edges receive smaller variances than those connected by short edges. Note that there are only homologous lines, no homologous polygon vertices. This means that *for each polygon vertex* of an edge feature e_i in image i:

1. A new object point $\mathbf{P}_{ID}$ with identifier ID has to be created in the reference system.

2. A point receiving the identifier ID is inserted into the *ORIENT* observation room corresponding to image i. The camera co-ordinates (u, v) of the points can be derived from the sensor co-ordinates (r, c) of the features by applying the affine transformation from equation 4.26. The stochastic properties of the image points can be derived from the variance-covariance matrix $\mathbf{C}_{rc}$ by error propagation. In this context, the similarity values S for corresponding image edges as indicated in section 5.2.2.1 can be used to give edges being more "similar" to the edges from the other images a higher weight. Note that in the current version of *ORIENT*, the off-diagonal elements of $\mathbf{C}_{uv}$ are neglected.

3. A point with identifier ID is inserted into the *GESTALT* rooms corresponding to the object faces intersecting at the object edge e the hypotheses is assigned to. Again, the a priori r.m.s. error of this observation is a measure for how rigidly the object model has to be fulfilled, and it has to be specified by the user.

Thus, for this kind of hypotheses, the following observation equations are inserted into adjustment:

1. Two camera co-ordinate observations per image edge vertex (equations 4.24), the weight depending on $\mathbf{C}_{uv}$ (and, should it turn out to be feasible, similarity S).

2. Two surface equations (two of equations 4.27).

In addition to the surface parameters, three new unknowns per image edge vertex have to be determined in adjustment (three object co-ordinates per vertex).

Model driven generation of correspondence hypotheses: In a model driven analysis, the model features have a specific meaning, and it is exactly the model features that are searched for in the images. No image-to-image correspondences have to be searched for: the model is projected to all images, and image features are assigned to object features. Search space is not reduced by epipolar constraints, but by the approximate positions of the object features back-projected to the images (figure 5.24). Two types of correspondences can be created:

1. a correspondence between an extracted feature point and a vertex of the object model, and
2. a correspondence between an extracted image edge and an edge of the object model.

The way these correspondences are treated in the *ORIENT* data base is similar to the one described above for the data driven technique. The only difference is that in the first case, if the object vertex is defined by the intersection of three or more faces, a point having the same identifier as the new hypothetical point has to be inserted into all these faces, which yields three surface equations 4.27 being inserted into adjustment (cf. section 5.4.1.2).

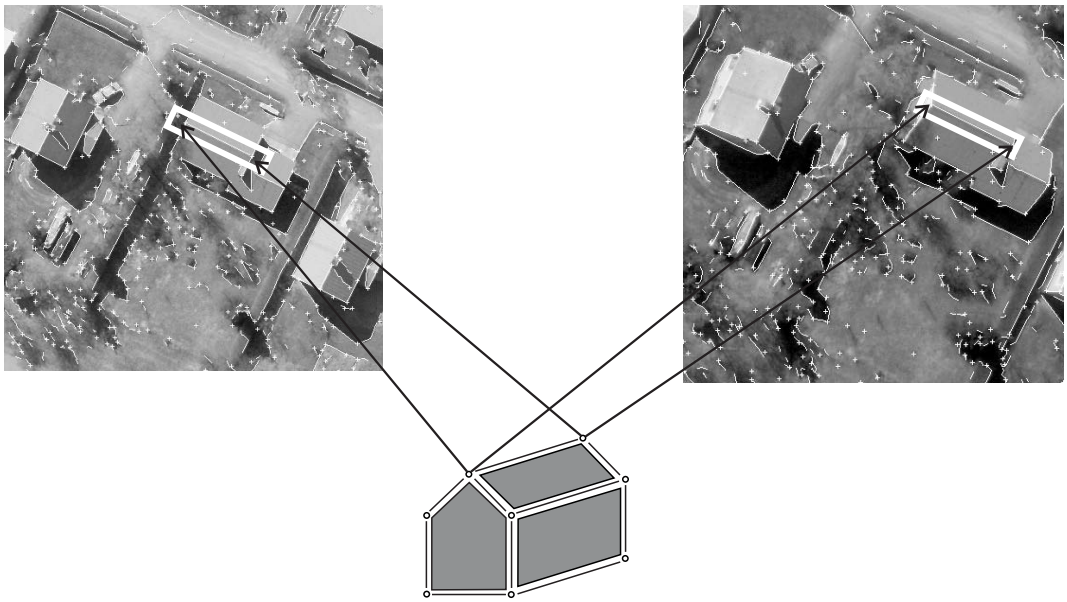


Figure 5.24: Model driven generation of correspondence hypotheses (two homologous image patches only).

Evaluation of correspondence hypotheses: Evaluation of correspondence hypotheses is performed by robust hybrid parameter estimation in the way described in chapter 4 using *ORIENT*. Three types of observations are used:

1. *Image co-ordinates:* For each image point and for each image edge vertex which is one of the partners of a correspondence hypothesis, two perspective observation equations 4.24 are inserted into the adjustment. The stochastical properties of these observations are determined during feature extraction and may be modified according to the similarity measure required for hypotheses generation. As the orientation parameters are assumed to be known, only the co-ordinates of the object point $\mathbf{P} = (X, Y, Z)^T$ are unknown.
2. *Surface observations:* For a point on a surface one and for each image edge vertex attached to an object edge two equations 4.27 are inserted into the normal equation system. In these observation equations a task-dependent subset of all possible parameters ($\mathbf{P}, \mathbf{P}_0, \theta = (\omega, \phi, \kappa)^T, a_{jk}, b_{ik}, c_{ij}$) is unknown

(cf. section 5.4.1.2 for details on parameterization of the surfaces and constant/unknown transformation parameters). The stochastical properties of these observations are controlled by the a priori r.m.s. error σ_s of a surface observation. σ_s describes the "rigidity" of the object models. It has to be provided by the user.

3. *Observed parameters* are required either for regularization purposes or for enforcing conditions, e.g. parallelity of co-ordinate planes of observation and object co-ordinate systems, respectively. In the first case, these observations will be given rather low weights as they should not influence adjustment too much, but just make some parameter determinable if required. In the second case, the weights of these observations will be very high as they are just meant to express that some unknown should be constant.

Starting from coarse approximate values, all unknown parameters are determined by iterative simultaneous adjustment of all observations. Most critical for convergence are the rotation angles. The approximations for these angles should be known with an accuracy of about 10-20 [gon]. After convergence has been achieved, robust estimation is applied to the observed camera co-ordinates only in order to eliminate false matches. We use the weight function from equation 4.18 and the strategy described in section 4.1.1.1: parameter h of that weight function is iteratively reduced depending on the size of the n largest normalized discrepancies still contained in adjustment. The observation equations corresponding to false matches, i.e. hypotheses contradicting the surface model, are thus eliminated from adjustment one after the other. The result of hybrid robust adjustment is given by an estimation for the surface parameters, the object co-ordinates and (if desired and possible) of $\mathbf{P}_0$ and (ω, ϕ, κ). In addition, the blunders are marked in the images so that the false correspondences can be found after adjustment. For that strategy, the approximate values have to be good enough to ensure convergence of the process, which means that the number of false correspondence hypotheses should not exceed 30%.

Together with the unknowns, *quality measures* have to be derived so that the algorithm can give the user a hint whether it was successful or not. This is not a simple task because usually, the measures derived from adjustment are far too optimistic. Here are some possible quality measures:

- The a posteriori r.m.s. error of the weight unit $\hat{\sigma}_o$ (equation 4.8). If it is significantly greater than its a priori value, the stochastical model of the adjustment was wrong, which might happen in case too many outliers were contained in the data.

- The a posteriori r.m.s. errors of the unknowns (equation 4.9) can be analyzed. If an unknown was at the edge of not being determinable, this will be reflected in its r.m.s. error. Such a situation might indicate that too few correspondences were found for a certain object feature.

- The redundancy of adjustment after elimination of the false observations: robust estimation only works if the redundancy is great enough. If this is not the case before error detection, robust estimation might fail. If it is not the case afterwards, this again might indicate that too many outliers were contained in the original hypotheses: in this case, it is to be suspected that "good" hypotheses were erroneously eliminated, too.

- The variance components of the observation groups [Kraus, 1997]: These components also might indicate a false stochastical model for a group of observations. For instance, a great variance component for surface observations would be an indicator for a bad fit of the model to the data.

- A coverage analysis for the object features might indicate whether there are object features which do not obtain support from image features.

Even though these quality numbers may give a hint on whether the process was successful or not, up to now, we cannot do without a visual inspection of the results by the human operator because we can see no guarantee that a failure of the process is actually reflected in these indicators.

5.4.1.4 Implementation aspects

Our framework for object reconstruction is implemented as a class library in C++. It is a typical application of the *ORIENT* data base interface (cf. section 4.3.4). The core of the framework is implemented in two C++ classes:

- *class MatchImage:* This class resembles the properties of one of the homologous image patches in the reconstruction process. It consists of the following members:
  - *imgRoom:* A reference to the observation room corresponding to the image in the *ORIENT* data base. It is of type *rasterRoom* (cf. section 4.3.4) so that it gives access not only to the image pyramid and to the image data files containing the individual levels of the image pyramid, but also to the point list and the mapping parameters of the observation room.
  - *fag:* The feature adjacency graph of the image. It is the symbolic description of the image.
  - *roi:* The region of interest in the image described by a rectangular window in the sensor co-ordinate system.
  - *dataBuffer:* The image data buffer containing the grey levels of the region of interest.
  - *setRoi(newRoi):* change the region of interest to *newRoi*. This method checks whether *newRoi* is inside the digital images or not.
  - *Read():* Read the contents of the image data files inside the region of interest to *dataBuffer*.
  - *extractFeatures(fexPar):* Extract the features inside the region of interest using the grey levels from the image data buffer *dataBuffer*. The parameters of feature extraction (cf. section 5.1) are passed in *fexPar*.
- *class MatchObject:* This class resembles the properties of an object patch in the reconstruction process. It is an abstract base class in which the object model is not yet defined. Class *MatchObject* consists of the following members:
  - *roi:* The region of interest as a prism parallel to the co-ordinate planes. It describes the maximum extensions of the object patch
  - *adjuster:* A reference to an object capable of adjustment. In this data member, *ORIENT* is wrapped up.
  - *imgList:* A list of instances of class *MatchImage*: all images in which the object patch is visible. This is checked by projecting the region of interest to all images using the access to the mapping functions of the observation rooms via *MatchImage*'s member *imgRoom*.
  - *setupModel():* A purely virtual function for model setup, i.e. the creation of the set of *GESTALT* rooms described in section 5.4.1.2.
  - *insertApprox(approxDescriptor):* A purely virtual method for providing approximate values for the object parameters. The approximate values are passed to that method in *approxDescriptor* which can be a point list or a polygon or the name of a file where the approximate values are stored. *imgList* will be updated after that procedure.
  - *extractFeatures(pyrLevel):* Extract features in all images contained in *imgList*.
  - *generateHypotheses(pyrLevel):* A purely virtual method for generating correspondence hypotheses at pyramid level *pyrLevel*.
  - *evaluateHypotheses(pyrLevel):* A method for evaluating correspondence hypotheses at pyramid level *pyrLevel*. It uses member *adjuster* for performing robust hybrid adjustment.
  - *reconstruct(pyrLevel):* Reconstruct the object at pyramid level *pyrLevel*. It successively calls *extractFeatures(pyrLevel)*, *generateHypotheses(pyrLevel)*, and *evaluateHypotheses(pyrLevel)*.

– *reconstructAllLevels(startLevel):* This method just provides a loop calling method *reconstruct(...)* in all pyramid levels starting at level *startLevel*.

In order to implement a new application of the framework, a class corresponding to an object patch has to be derived from class *MatchObject* in which the features of the object model have to be contained. In this class, the purely virtual methods *setupModel()*, *insertApprox(approxDescriptor)*, and *generateHypotheses(pyrLevel)* of the virtual base class have to be implemented using the domain specific knowledge about the object structure. This work has already been completed for the automation modules of semi-automatic building extraction (cf. chapter 6). It is currently performed for the task of DEM generation for topographic mapping (cf. section 5.4.2).

5.4.2 Example: DEM generation for topographic mapping

For the purpose of small scale topographic mapping, the earth surface can be assumed to be smooth. That is why a 2.5D grid DEM which can be derived from a point cluster in object space (cf. section 2.2.1) is well-suited for modelling. In order to apply our framework to the task of deriving such a DEM from digital images, we first have to think about what the object model in the reconstruction process has to look like. We have already described in section 5.4.1.2 how a mesh of a 2.5D grid can be modelled making use of our concept of surface observations (see also the left part of figure 5.21). In the reconstruction process at each pyramid level i, the whole area of interest in the terrain which is to be reconstructed is split into patches of $n_x \times n_y$ such grid meshes of grid sizes $\Delta X_i = r^i \cdot \Delta X_0$ and $\Delta Y_i = r^i \cdot \Delta Y_0$, where $(\Delta X_i, \Delta Y_i)$ are the grid sizes of the DEM from pyramid level i and r is the reduction factor of the pyramid, i.e., $r = 2$ in most cases (figure 5.25). The grid sizes $(\Delta X_0, \Delta Y_0)$ have to be provided by the user as they are equal to the grid sizes of the result he or she wants to obtain. The number of grid meshes n_x and n_y can also be specified by the user.

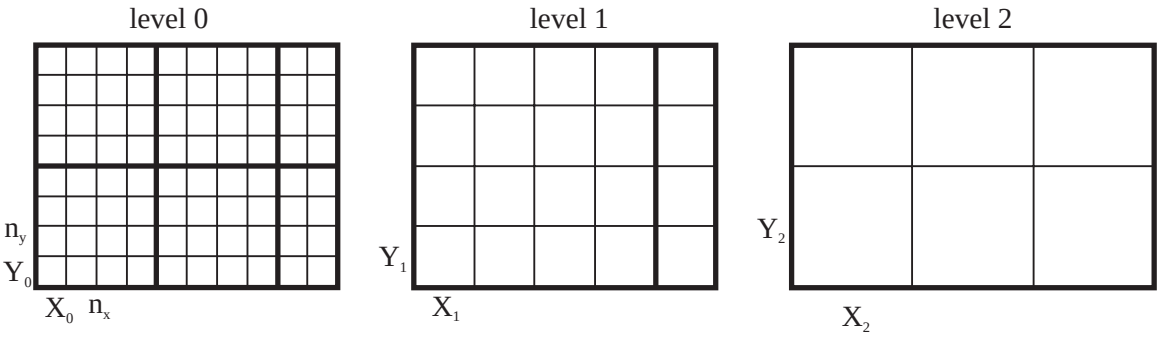


Figure 5.25: The structure of the DEM in the reconstruction process for three pyramid levels with $r = 2$ and $n_x = n_y$ = 4.

The grid meshes must not be treated individually because that would lead to steps in the terrain at the grid mesh borders. In our formulation of the problem, for each grid mesh, a *GESTALT* room as described in section 5.4.1.2 is created, and the four corner points of the mesh are inserted into that room. This means that each grid point (except those at the border of the current patch) is contained in four such *GESTALT* rooms because it is a corner of four grid meshes. Thus, for each of those grid points, four equations 5.36) are inserted into adjustment, each giving support to the four parameters of one of the grid meshes. This is the way how the grid meshes are forced not to be separated by steps in the terrain heights. In order to make the planimetric co-ordinates of the grid points determinable, control point observations have to be created for them. Thus, after the model setup phase we obtain:

- $4 \cdot n_x \cdot n_y$ observation equations 5.36 (one for each grid point of each mesh). The stochastic properties of these observations are given by the r.m.s. error σ_s of a surface observation. As described in section 5.4.1.3, σ_s describes how rigidly the mathematical model of the surface has to be fulfilled, and the

decision which points are considered outliers is heavily influenced by that parameter. It has to be chosen in dependence of the expected standard deviation of the terrain which can be estimated from the image configuration [Kraus, 1993].

- $2 \cdot (n_x + 1) \cdot (n_y + 1)$ observation equations 4.28 (one for X and one for Y of each grid point). These observations have to prevent singularities and thus can be assigned a very small a priori r.m.s. error σ_c, e.g., $\sigma_c = \pm 0.001$ m
- $4 \cdot n_x \cdot n_y$ unknown surface parameters $(c^j_{00}, c^j_{10}, c^j_{01}, c^j_{11})$ (4 parameters per grid mesh j)
- $3 \cdot (n_x + 1) \cdot (n_y + 1)$ unknown co-ordinates of the grid points.

Our formulation of the problem is similar to the formulation of DEM estimation by finite elements (cf. section 2.2.1, [Krzystek and Wild, 1992]). Our method has the drawback that more unknowns are to be determined as in the classical finite element approach, where the surface parameters are expressed as functions of the grid points and the planimetric co-ordinates of the grid points are kept constant. On the other hand, our approach is more flexible with respect to extensions of the representation of the surface within a grid mesh, and it uses the primary observations (the camera co-ordinates) for adjustment. In the classical finite element method, additional observations for regularization are used which enforce the interpolated grid to have smooth transitions at the borders of the meshes: in grid point $\mathbf{P}_{i,j}$, the curvatures of the connecting lines between its two neighbours in both co-ordinate directions should be identical, thus the ascensions of the straight lines $\overline{\mathbf{P}_{i,j-1}\mathbf{P}_{i,j}}$ and $\overline{\mathbf{P}_{i,j}\mathbf{P}_{i,j+1}}$ should be identical. A similar observation can be made for the straight lines $\overline{\mathbf{P}_{i-1,j}\mathbf{P}_{i,j}}$ and $\overline{\mathbf{P}_{i,j}\mathbf{P}_{i+1,j}}$. [Krzystek and Wild, 1992] additionally introduce observations for the torsion across the diagonals of the grid meshes. The r.m.s. error σ_r of these observations which their weights depend on have to be selected carefully as they influence the degree of smoothing of the DEM. In our framework, such observations can be inserted, too: in each (interior) grid point, two additional *GESTALT* rooms can be created with constant rotation angles which are all zero. A room of additional parameters for w-observations is created for each of these *gestalt* rooms. $\mathbf{P}_0$ is selected to be $\mathbf{P}_{ij}$. For observing the curvature in X direction, the points $\mathbf{P}_{i-1,j}$ and $\mathbf{P}_{i+1,j}$ are inserted into adjustment, and a linear coefficient in u direction c^{ij}_{10} is chosen as the parameterization. Similar considerations can be performed for the curvature in Y direction, so that for each (interior) grid point $\mathbf{P}_{ij}$, four observation equations for regularization can be inserted into adjustment:

$$\begin{aligned}
\tilde{v}^X_w &= Z_{i-1,j} - Z_{i,j} + c^{ij}_{10} \cdot (X_{i-1,j} - X_{i,j}) \\
\tilde{v}^X_w &= Z_{i+1,j} - Z_{i,j} + c^{ij}_{10} \cdot (X_{i+1,j} - X_{i,j}) \\
\tilde{v}^Y_w &= Z_{i-1,j} - Z_{i,j} + c^{ij}_{01} \cdot (Y_{i-1,j} - Y_{i,j}) \\
\tilde{v}^Y_w &= Z_{i+1,j} - Z_{i,j} + c^{ij}_{01} \cdot (Y_{i+1,j} - Y_{i,j})
\end{aligned} \tag{5.38}$$

Equations 5.38 make our object model complete. Note that for the two *GESTALT* rooms created for each grid point $\mathbf{P}_{ij}$, two additional unknowns have to be determined: the linear coefficients c^{ij}_{10} and c^{ij}_{01}. The a priori r.m.s. error σ_r of these observations which influences the degree of smoothing can be specified by the user.

The reconstruction process starts at the upper pyramid level N. The coarse approximations from the flow chart in figure 5.20 are derived from a DEM of grid widths $(\Delta X_{N+1}, \Delta Y_{N+1})^T = 2^{N+1} \cdot (\Delta X_0, \Delta Y_0)^T$ which can, for instance, be interpolated using the control points of the aerial block, or it can be derived under the assumption that the terrain is a tilted plane. At each pyramid level i, the grid points of the DEM corresponding to pyramid level i can be interpolated in the DEM from pyramid level $i+1$. The interpolated grid can be projected to all images. The projected grid points define a distorted grid in all images, and meshes of that distorted grid can be assumed to be homologous image patches corresponding to the same grid mesh in object space (grey patches in figure 5.26). Within these patches, hypotheses for corresponding image points are searched for in the way described in section 5.4.1.3 These hypotheses are supposed to give support to the corresponding grid mesh, so that a surface observation equation 5.36 can be inserted into adjustment in addition to the two

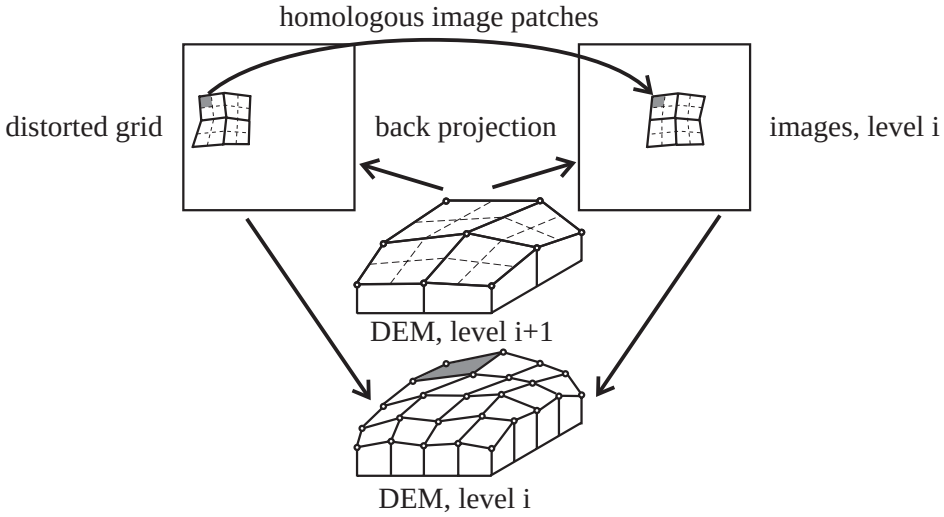


Figure 5.26: Iterative generation of a DEM. Grey: homologous image patches and the corresponding object patch. Broken lines: the DEM grid corresponding to pyramid level i interpolated in pyramid level $i + 1$.

perspective observation equations 4.24 per image feature. As described in the paragraph on hypotheses about corresponding image points in section 5.4.1.3, each hypotheses involving points from k images increases the redundancy of the normal equation system by $2 \cdot k - 2$. Thus, under the assumption that enough hypotheses are found, parameter estimation can be performed on the basis of a great redundancy, an important prerequisite for robust estimation.

The surface patches of pyramid level i are united in a post processing step, where a 2.5D DEM covering the whole region of interest is computed from the estimated grid points of the individual patches (cf. the work flow in figure 5.20). This application of our framework is currently implemented in the course of a diploma thesis.

Chapter 6

Semi-automatic extraction of buildings

Our system for semi-automatic building extraction is based on the integration of CAD and photogrammetry in the sense of figure 1.2. A general overview on the properties of our system and a comparison to other work in the field of semi-automatic building extraction has already been given in section 1.3. In the current chapter we want to describe our system in detail. We will start with an overview on the work flow and the components of our system in section 6.1. Section 6.2 is dedicated to our specific way of representing domain-specific model knowledge about buildings on the basis of the principles of the framework for object surface reconstruction described in section 5.4. The way this model knowledge is used for a topology-based extraction of buildings in an interactive work flow is presented in section 6.3. Automatic fine measurement of building primitives is implemented as a special application of our framework for object reconstruction. It will be described in section 6.4.

6.1 System overview

Our system for semi-automatic building extraction has been integrated into the program *ORPHEUS* for digital photogrammetry (cf. section 4.4). It uses the principles of the framework described in section 5.4 for modelling the building or, to be more precise, building parts, in the reconstruction process. We start from the following input data:

1. *Digital images and their orientation parameters*: As our system is integrated into *ORPHEUS*, these data can be accessed via the C++ interface to the *ORIENT* data base described in section 4.3.4. For the digital images, image pyramids are available. The mathematical model for the mapping functions and thus both numbers and interpretation of the orientation parameters of the images are those described in section 4.2. In the current version, we are restricted to handling perspective images, i.e., to the mathematical model of section 4.2.2. Nevertheless, this is a restriction of the state of implementation of the C++ data base interface only. As soon as new observation types, i.e., new sensor models, can be accessed via that data base interface, they can be immediately used for semi-automatic building extraction, too.

2. *Data base of known building shapes*: This data base contains the domain specific model knowledge about buildings in the form of a set of parametric building primitives and several generic building types. The building primitives (as we will from now on call both the parametric primitives and the generic building types because they are treated in the same way in our system) are considered to be data so that the data base of building primitives can be expanded by a human operator. The internal representation of the building primitives is given by B-rep on the basis of the principles described in section 5.4. More details on that will be given in section 6.2.

3. *Digital terrain model* (optional): In our system, only the building roofs can be measured automatically. This means that the floor height of a building cannot be determined by the automated tool. If a DTM is

available, it can be used to determine the floor height. We use a hybrid 2.5D grid DTM containing geomorphological data as it is created by the program system *SCOP* for that purpose (cf. section 2.2.1). If no DTM is available, the remaining height parameter can either be determined by interactive measurement of a floor point, or it can be kept fixed at a certain initial value. In the latter case, its determination is postponed.

Our system can be considered to be a hybrid solid modeller based on B-rep in the way presented in section 2.3.5 (cf. especially figure 2.19). This means that

1. internally, the buildings are modelled by boundary representation
2. a CSG interface is provided for the operator so that a building can be successively reconstructed by combining basic building primitives (in the sense defined at the beginning of this section) using Boolean operators.

Our system was developed with the explicit goal of providing tools making it possible for a user to obtain 3D descriptions of buildings which are to be inserted into and managed in a TIS. The part of the TIS is taken over by the program *SCOP.TDM* in our system. From that point of view, a building is an instance of a 3D topographic object in the sense depicted in figure 3.5. The class *building* is derived from class *topographicObject3DwithBrep* in figure 3.5. The integration of the relational topographic data base *SCOP.TDM* and our system is based on the principles for hybrid geometrical modelling described in section 3.2.2: as soon as the reconstruction of a building is finished, the building is handed over to the topographic object manager which is responsible for inserting the building's meta data into *SCOP.TDM*, and the detailed description of the building is treated as a BLOB. At any instance, the buildings can be queried from the data base according to some (thematic or geometrical) attributes. From the results of the SQL query, the topographic object manager will create instances of class *building* which can be handed over to an application, for instance for visualization or for a conversion to another data format (cf. figure 3.6).

In our system, in the sense of object oriented programming, a *building* consists of :

1. *Meta data*: the meta data of the building are the attributes which are stored in the topographic data base. The most important attributes are the building identifier, the bounding box (i.e., the prism parallel to the co-ordinate axes containing the whole building), and the name of the binary file where the building is made persistent.
2. *Boundary representation*: a full winged-edge data structure in the way described in section 2.3.1.1. This is the central representation of the building geometry. It is realized by the integration of the *VRaniML*^{TM} library [Great Hill Corporation, 2000]. Additional semantic information is added to the geometrical description of the building:
   - the faces are marked as being either a wall, the floor, or a part of a roof
   - the edges are marked as being either eaves, ridges, or no special edges.

   The boundary representation of a building can be made persistent on a binary data file. We do not assign semantic information to building parts in order to distinguish the main body of a building from features such as dormers or chimneys in the way it is done, e.g., in [Koehl, 1997].
3. *CSG tree*: During building extraction, the human operator de-composes the building into parts which can be modelled by the primitives of the building model data base. One primitive after the other is reconstructed and then added to the building by applying a Boolean operator. As long as the building is still extracted, the CSG tree of the building is stored as a list of primitives and operator types ordered by the order of their insertion into the building. This information is only available during building extraction, and it is not made persistent.

4. *Tools for adding primitives to the building*: As soon as a primitive has been reconstructed, it is added to the whole building. The primitive itself is also modelled by B-rep, thus, when the primitive is added to the building, a Boolean operator (either a union, an intersection or a difference) has to be applied to the B-rep describing the current state of the building and the new primitive. This means that the B-rep of the building is updated immediately when a new primitive is added to it, and the results are super-imposed to the images. We use the method by [Mäntylä, 1988] for applying Boolean operators to B-reps in the way described in section 2.3.1.3. This method is implemented in the *VRaniML*^{TM} library.

5. *Tools for making the data persistent*: Finally, a building needs tools for writing its relevant data (i.e., the B-rep and the semantic attributes of the faces and the edges) to and reading them from disk.

From this description, it is not yet clear whether in our system a "building" is a single building in the sense of a building that can be identified by a postal address or whether it is a block comprising a continuously built-up area. In fact, this decision is left to the operator: in densely built-up areas it is often very difficult to distinguish "buildings" in the sense of the first definition from each other just from the aerial images, and on the other hand, in a TIS one would like to be able to select buildings by an attribute such as the postal address. We suggest that the operator should split up building blocks into smaller buildings wherever he or she can do so by interpretation of aerial images, and otherwise he or she should consider groups of buildings to be one object. In the latter case, additional information (e.g. existing TIS data or information from the cadastre) has to be used to split the building blocks into "buildings" in the sense of the first definition. This cannot be done yet in our system.

Our system for semi-automatic building extraction is integrated into *ORPHEUS* as a specific tool for measurement. Its structure is presented in figure 6.1. In comparison to the structure of a measurement tool as described in section 4.4 (cf. also figure 4.12), it additionally contains an interface to *SCOP.TDM* via the topographic object manager in the way described in section 3.2.2. This architecture enables our system to use the functionality of both *ORPHEUS* and *SCOP.TDM* to fulfil the requirements described in section 1.2.3 with respect to visualization and data management:

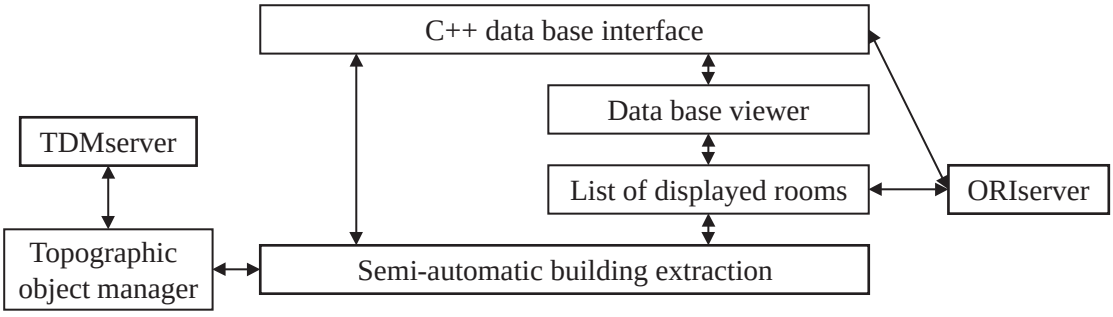


Figure 6.1: The architecture of semi-automatic building extraction as a measurement tool in *ORPHEUS*.

- As soon as a project is loaded in *ORPHEUS*, the project specific data are loaded and thus available to the application via the data base interface.
- Via the list of displayed rooms, the images which are currently displayed on the screen are accessible, e.g., for visualization.
- As soon as semi-automatic building extraction is activated, it has to query all buildings which are visible in at least one image from the server version of *SCOP.TDM*. This is performed via the topographic object manager. The buildings are back-projected and super-imposed to all displayed images as wire frames.
- As soon as semi-automatic building extraction is finished, these super-imposed wire frames have to be erased.

- Whenever the user chooses to display another image, the tool for semi-automatic building extraction gets a notice of it from the data base viewer. It will query all the buildings visible in the new image from *SCOP.TDM* via the topographic object manager and display them also in the new image.
- In the same way as the buildings from the data base are displayed, the currently active building and the currently active primitive are super-imposed to the images as wire frames.

The work flow for the extraction of one building will be described in section 6.1.1. After that, we will describe some practical aspects of our system in section 6.1.2.

6.1.1 Work flow for building extraction

The work flow for the extraction of one building is presented in figure 6.2. It consists of several steps:

1. *Building initialization*: Upon user request, a new instance of a building is created. This instance does not yet contain geometrical information.
2. *Adaptation of a single primitive*: As stated above, a building is created primitive by primitive, the primitives being added to the building by Boolean operators via a CSG interface. Again, several steps have to be performed:
   (a) *Select primitive and Boolean operator*: This has to be done by the user. The type of primitive the user wants to reconstruct can be selected from all available primitive types in the data base of known building shapes. A wire frame sketch of the selected primitive type is displayed on the GUI for an easy interpretation (cf. figure 6.3). In addition, the user has to choose by which Boolean operator the primitive has to be added to the current building. The contents of the data base of known building shapes will be described in detail in section 6.2.
   (b) *Primitive initialization*: As soon as a primitive type and a Boolean operator have been selected, the user can decide to initialize a primitive. Primitive initialization depends on the primitive type: parametric primitives can be completely initialized without any additional information using some default values for the building parameters. For generic types, the user has to digitize a polygon in one of the images. From that polygon, a prismatic building "primitive" can be initialized. The way the primitives are modelled and how they are initialized will be described in section 6.2.
   (c) *Interactive editing of the primitive*: As soon as the primitive has been initialized, it is super-imposed to all currently displayed images it is visible in. The user can adapt the parameters of the building by interactively measuring building vertices in the digital images. These measurements need not necessarily be accurate if the user intends to perform fine measurement automatically. Interactive adaptation of the building parameters will be described in section 6.3.
   (d) *Automatic fine measurement of the primitive*: If the user selects to do so, the parameters of the primitive will be automatically detected by feature based matching on the basis of our framework for object surface reconstruction (cf. section 5.4). Note that the user can do so at any instance when he or she decides that the parameters of the building primitive are close enough to their actual values so that matching will succeed. Automatic fine measurement will be described in detail in section 6.4
   (e) *Interactive post-editing of the primitive*: If despite of the user's expectations the matching algorithm fails, the parameters of the building primitive can be re-adapted interactively in the same manner as before automatic measurement.
3. *Addition of the primitive to the building*: At any time the user can decide that the current building primitive fits to the image data well enough so that it can be accepted. As soon as the user accepts a primitive, the primitive is added to the current building. In case a DTM is available, the heights of the floor vertices

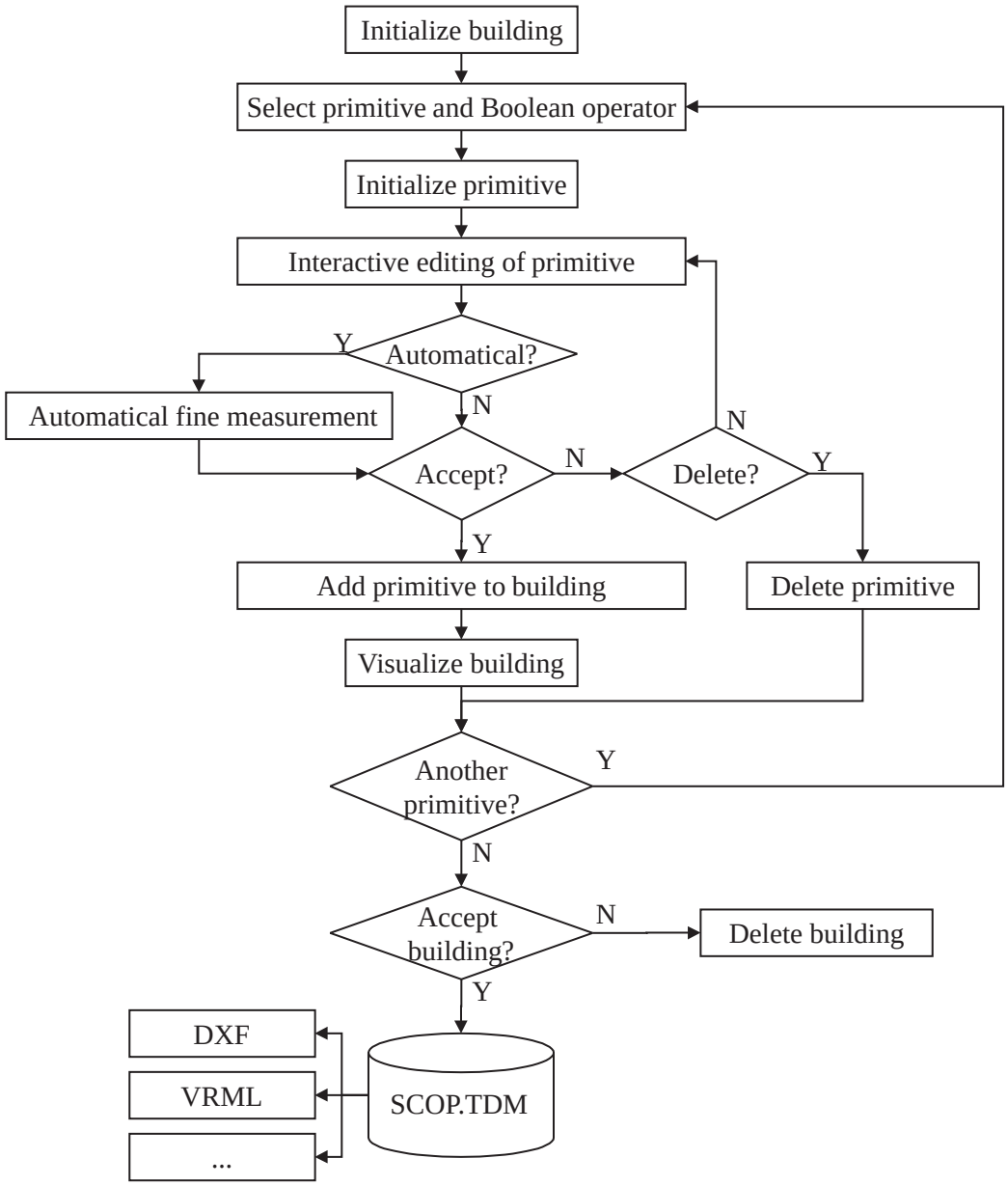


Figure 6.2: The work flow for the reconstruction of one building.

of the primitive are interpolated from the DTM and the building floor is shifted to the interpolated height of the lowest floor vertex of the primitive if this height is smaller than the height of the compound building. If it is not, the floor of the primitive is shifted to the height of the compound building. After adapting the floor height, the Boolean operator selected by the user will be applied to the two B-reps, the B-rep of the current building being updated immediately. The new structure of the current building will also be super-imposed to the digital images. As long as the current building is still reconstructed, the CSG tree will be preserved. Of course, the user also can choose to discard the current primitive at any instance, e.g., when he or she detects that a false primitive type has been chosen.

4. *Accept the building and transfer it to* SCOP.TDM: If the user thinks that the building is not yet completely reconstructed, he or she can choose another primitive from the data base of known building shapes and repeat the previous steps. At any instance, the user can decide that the building is completely reconstructed. In this case, he or she can accept the building, and the building will be transferred to *SCOP.TDM*. At this

instance, the CSG tree is discarded. Of course, the buildings contained in *SCOP.TDM* can be transferred into another data format, e.g., *DXF* for data exchange with standard CAD software or *VRML* for creating animations for the internet. If the user decides at a certain instance that he or she is not satisfied with the current building, the building can also be deleted.

By the work flow for building extraction, work is saved in many ways compared to conventional analytic photogrammetry:

- The primitives are already consistent 3D models in B-rep containing all the topological information required. In conventional photogrammetric systems, only the wire frame model can be defined, and the face information has to be added off-line, for instance in a standard CAD program, which is very tedious work.

- By the specific way the primitives are modelled in the reconstruction process, additional geometrical information such as information about edges being horizontal or vertical or information about symmetries or orthogonality of faces is exploited. Thus, even if automatic fine measurement fails, work is saved because only a few primitive vertices actually have to be measured in the digital images.

- If automatic fine measurement succeeds, interactive measurement of an even smaller number of primitive vertices is sufficient to provide approximate values. In addition, these vertices only have to be measured approximately, which leaves the tedious pointing process to the computer. As automatic fine measurement is based on line matching, the building corners are determined from intersection of the faces only, and the faces will be adjusted to the image lines, which are far more representative for the building primitive than the vertices. Thus, we think that the resulting object co-ordinates of the primitive vertices will be better determined than those derived from interactive measurement, especially if the primitive corners are only poorly visible or blurred in the images.

- Finally, the Boolean operations involve the computation of the intersection points of the edges of the primitive and the faces of the current building and vice versa. In a conventional photogrammetric system, these intersection points would have to be measured interactively, too, or they would have to be generated off-line using some intersection tool of a CAD program.

6.1.2 Practical aspects

Currently, the system is working on all *WINDOWS* platforms starting from *WINDOWS 95*. Building extraction is activated by opening the respective window hidden behind a button on the *ORPHEUS* main window. Figure 6.3 shows the control window for semi-automatic building extraction. Note that at a certain instance, some of the fields of the GUI are de-activated in order to prevent the user from inconsistent actions such as accepting a building without any primitive. The fields on the GUI have the following meaning:

- *New building*: By selecting this button, a new building is initialized.

- *Accept building*: By selecting this button, the current building is accepted, and it will be transferred to *SCOP.TDM*. The button becomes active if at least one primitive has already been added to the current building.

- *Remove building*: By selecting this button, both the current primitive and the current building are discarded.

- *Building primitive*: This group of GUI fields describes the current primitive type and the Boolean operation by which it is to be added to the current building. It consists of three fields:

  - Selection of the primitive type: In this field, the user can select the type of the next primitive to be added to the building by its name. This field is only active if no current primitive exists, i.e., before initialization and after acceptance or removal of a primitive.
  - Graphical sketch of the primitive type: This sketch shall support the user in the interpretation of the primitive type. In addition, during interactive editing, the user can snap the vertex he or she wants to digitize next.
  - Selection of the Boolean operator: In this field, the user can select the type of the Boolean operator by which the next primitive is to be added to the building. This field is only active if no current primitive exists. Of course, the first primitive has to be added to the building by a "union" operator.

- *New primitive*: By selecting this button, a new primitive of the type currently displayed in the *Building primitive* group will be initialized.
- *Remove primitive*: By selecting this button, the current primitive will be deleted.
- *Accept primitive*: By selecting this button, the current primitive is accepted, and it will be added to the current building as described in section 6.1.1.
- *Reconstruct*: Selecting this button will activate automatic fine measurement.
- *Undo*: Only the last modification of the current primitive can be undone in the current version. This includes the results of automatic fine measurement.
- *Options...*: Behind this button, another window for the specification of parameters both for interactive editing and automatic fine measurement is hidden. These control parameters will be described in sections 6.3 and 6.4.
- *VRML...*: A *VRML 2.0* export of all buildings.
- *Identifier*: The identifier of the current building. This number is just displayed for informative purposes.

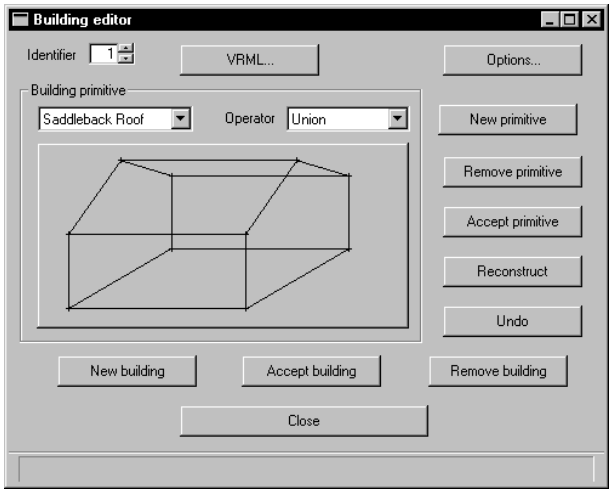


Figure 6.3: The GUI for building extraction.

As soon as building extraction is activated, the contents of the topographic data base will be super-imposed to all displayed images. Typically, two images will be displayed on the screen. Figure 6.4 presents a typical situation in the work flow of building extraction. Two images are displayed using the image visualization tools of *ORPHEUS*. As discussed in section 4.4, for each image, an overview and a zoom window are provided. Note that *ORPHEUS* offers the possibility of choosing a zoom factor freely for both windows. The zoom window can be positioned by click on the central mouse button. Typically, the zoom window will be used for interactive

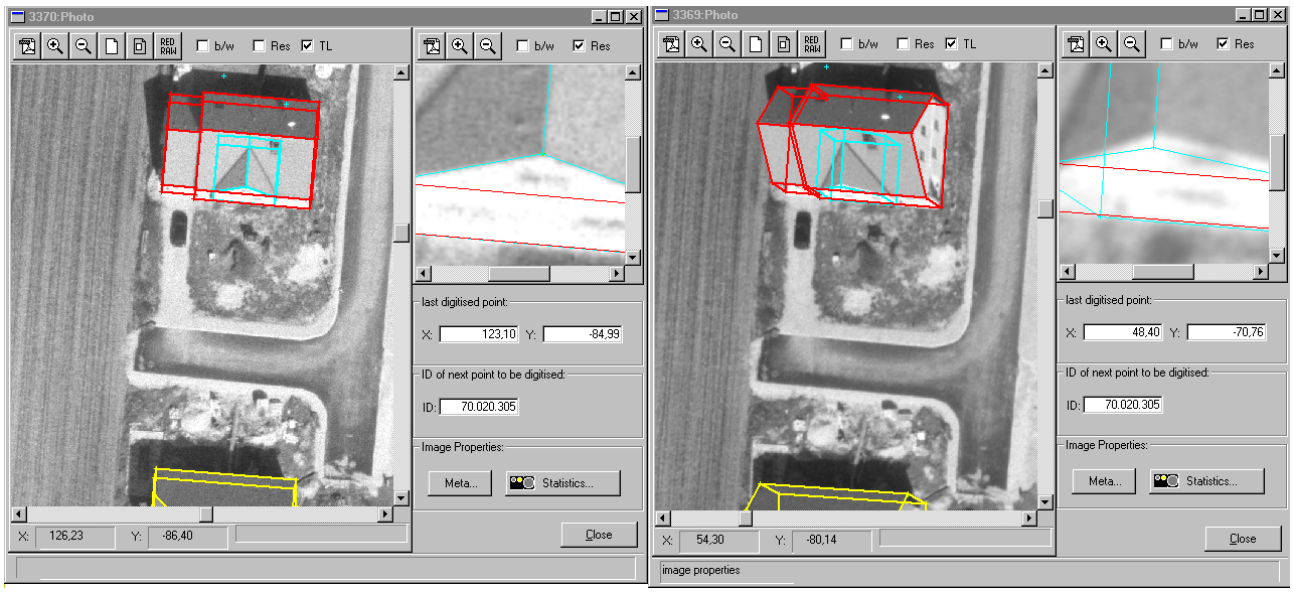


Figure 6.4: An example for the working environment of building extraction. Two images are displayed on the screen. The buildings which have already been reconstructed are displayed in yellow. The current building already consisting of the Boolean union of two saddle back roof primitives is displayed in red. A third primitive, also of type “saddle back roof” is still adapted; it is displayed in cyan.

fine measurement if it turns out to be necessary. However, in some situations it might be useful to zoom into the overview window in order to obtain a better overview for an appropriate scene interpretation. If the user wants to inspect the building from another viewing angle (if such an image is available), he or she can open another image at any instance, and both the current and the already reconstructed buildings will be super-imposed to that image.

Figure 6.4 also shows the way the buildings are displayed in the images:

- The buildings which are already contained in the topographic data base are displayed in yellow
- The current building is displayed in red. Note that in figure 6.4, the current building consists of two primitives of type “saddle back roof” which have already been intersected. For that project, a DTM was used to determine the height of the building floor.

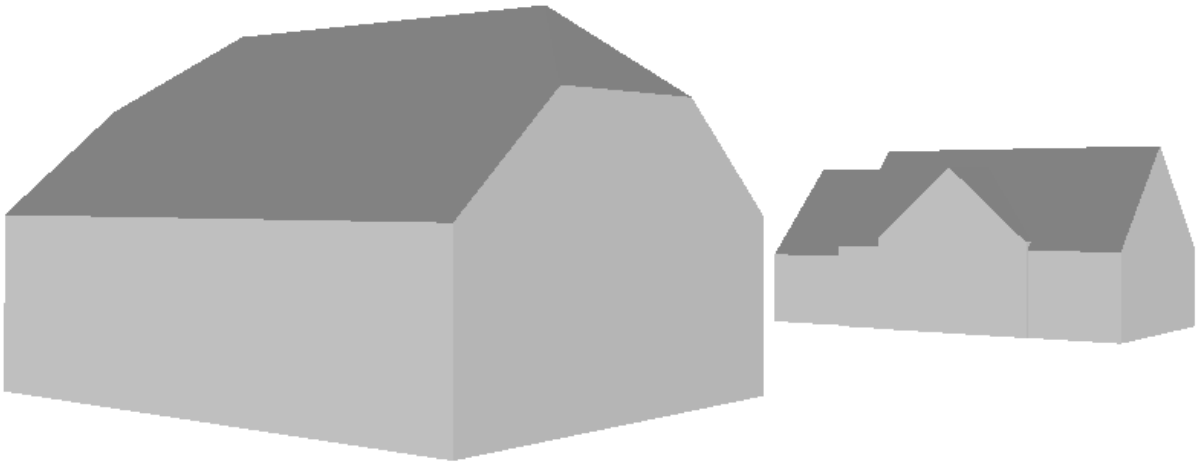


Figure 6.5: A *VRML* visualization of the two buildings displayed in figure 6.4.

- The current primitive, again a saddle back roof, is displayed in cyan. It has already been partly adapted to the images. As soon as the user would decide from the super-imposed wire frame that the primitive fits well to the image data, he or she could accept the primitive, which would then be added to the current building. In this case, the updated building wire frame would be displayed in red, the cyan parts being erased.
- As soon as the user accepts the current building, its colour will be changed to yellow.

Figure 6.5 shows a visualization of the buildings from figure 6.4 in the *VRML* format.

6.2 Modelling building primitives using the concept of surface observations

In section 6.1.1 we have described the work flow for the reconstruction of a building. We have shown that the building is reconstructed primitive by primitive, and it is the primitive parameters that are adapted to the image data both by points digitized by the operator and by the matching tools. During the reconstruction of the primitives, parameter estimation is performed by *ORIENT*. That is why the requirements for parameter estimation have to be fulfilled by the technique used for modelling the building primitives. As described in section 6.1, the primitives, just as the buildings, are modelled by B-rep using a full half-edge data structure. The B-rep consist of faces, loops, edges, and vertices. However, with respect to the primitives, there are some modifications in the way described in section 5.4.1.2 so that they can be used for object reconstruction:

- Each face of the B-rep of a building primitive corresponds to a *GESTALT* room in the *ORIENT* data base.
- The primitive is described in a local co-ordinate system which is supposed to be the observation co-ordinate system for the estimation of the surface parameters. It is linked to the object co-ordinate system by the exterior reference point $\mathbf{P}_0$ and three rotational angles (ω, ϕ, κ), of which ω and ϕ are zero. The remaining four parameters (X_0, Y_0, Z_0) and κ describe the primitive's position and orientation in the object co-ordinate system.
- The parameterization of the plane corresponding to a face refers to that local co-ordinate system. It can be chosen so that symmetry constraints are fulfilled automatically. The shape of the building in the local co-ordinate system is described by the surface parameters.
- All vertices of all loops belonging to a face are inserted into the respective *GESTALT* room. This implies that one vertex is contained in all *GESTALT* rooms corresponding to the faces intersecting at the vertex.
- For each vertex contained in one *GESTALT* room, one surface equation will be inserted into adjustment in the way described in section 5.4.1.2.
- From the point of view of parameter estimation, a building primitive is a system of planes described in a unique observation co-ordinate system. The topology of the building is used implicitly in parameter estimation as each building vertex is contained in several *GESTALT* rooms. The parameters to be estimated are:
  - The position and orientation parameters of the primitive
  - The surface parameters of the primitive
  - The vertex co-ordinates.

  As each vertex is contained in at least three surfaces, its co-ordinates can always be estimated from surface observations. The remaining unknowns describing the building position, orientation and shape have to be determined from additional information, i.e., from observations either provided by the operator or by the automated tools.

- As long as no additional information is available, for all unknowns not determined from the surface observations, parameter observations are added to adjustment.

This representation of a primitive for parameter estimation is a special case of the modelling technique used in our framework for object surface reconstruction (cf. section 5.4.1). It is used both for interactive editing and for automatic fine measurement.

We have already stated in the previous sections that a data base of known building primitive shapes is provided by our system. From what we have said up to now it is clear that for each known primitive type, that data base has to provide all the information required for the construction of a B-rep model and for the set-up of all the data relevant for parameter estimation, i.e., the full half-edge data structure, the description of the parameterization of all faces, and initial values for the surface parameters. In the phase of initialization of the primitive, the B-rep model is created and the *ORIENT* data base is modified so that it contains all observation rooms and parameters required for the primitive. As stated in section 6.1.1, two groups of primitive types are provided by our system which differ by the way their initialization has to be performed:

- *Parameterized primitives*: Simple building shapes such as hip roof or saddle back roof buildings. The topology of these primitives is completely described in the data base so that the primitives can be initialized even if no additional information is available. Only the geometrical parameters have to be adjusted to the image data. This part of our data base of known building primitive shapes can be adapted by the user. It will be described in detail in section 6.2.1.

- *Generic primitives*: Our system provides two generic primitive types, too. For these types, a construction rule has to be provided rather than a fixed topology: For instance, a prismatic building is characterized by two horizontal planes (roof and bottom) and a set of n vertical planes (walls). Even though the terms "generic" and "primitive" are somewhat contradictory, we use the term "generic primitive" because with the exception of initialization, these primitive types are treated in exactly the same way as the parametric types are. The difference is that both the number of geometrical parameters and the topology of generic primitives is not known a priori because the number of walls is unknown. Note that this part of the data base of known building primitive shapes cannot be adapted by the operator because the construction rules have to be programmed. We will have a closer look at the generic primitive types offered by our system in section 6.2.2.

6.2.1 Parametric building primitives

As stated above, parametric building primitives describe common simple building shapes such as hip-roof or saddle back roof buildings. (cf. figure 6.6). The data required for the initialization of a parametric building primitive have to be contained in the data base of known building primitive shapes.

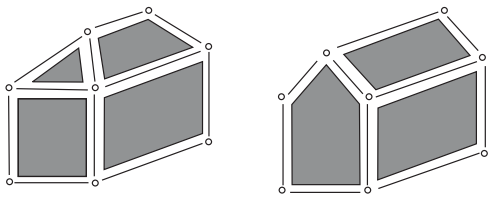


Figure 6.6: Surface Models for building reconstruction: A hip-roof building (left) and a saddle back roof building (right).

In order to add a new primitive to the knowledge base, the user has to provide a set of building corner points v, a set of faces f and a set of edges e. For each edge, its starting and end vertices and the neighbouring surfaces have to be defined. For each surface, the mirror matrix $\mathbf{M}$ and either a subset of the coefficients a_{jk}, b_{ik}, c_{ij} from equation 4.27 defining its mathematical formulation or alternatively a reference to a symmetrical surface

have to be provided. In addition, the outer boundary loop of the surface has to be defined by an ordered set of vertices. Note that all surfaces are formulated in the same observation co-ordinate system. Thus, in all surface equations, the same exterior reference point $\mathbf{P}_0$ and the same rotations θ are used. Taking the saddle back roof as an example, the following formulation would be adequate (figure 6.7): The reference point $\mathbf{P}_0$ is situated in the centre of the floor, which is enforced implicitly by the way the faces are formulated. The facades are vertical, thus only one rotation κ around the Z-axis is required to determine the orientation of the building in object space. The model consists of seven surfaces:

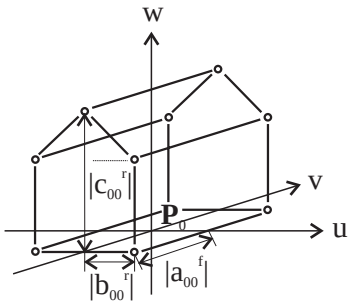


Figure 6.7: Parameterization of a primitive representing a saddle back roof building.

1. Floor: $w_0 = 0; m_u = m_v = m_w = 1$
2. Front facade: $v_0 = a_{00}^f; m_u = m_v = m_w = 1$
3. Back facade: $v_0 = a_{00}^f; m_u = m_w = 1, m_v = -1$: the same parameters as (2), but mirrored with respect to the uw-plane.
4. Right facade: $u_0 = b_{00}^r; m_u = m_v = m_w = 1$
5. Left facade: $u_0 = b_{00}^r; m_u = -1, m_v = m_w = 1$ the same parameters as (4), but mirrored with respect to the vw-plane.
6. Right roof plane : $w_0 = c_{00}^r + c_{10}^r \cdot u_R; m_u = m_v = m_w = 1$
7. Left roof plane: $w_0 = c_{00}^r + c_{10}^r \cdot u_R$; $m_u = -1$, $m_v = m_w = 1$: the same parameters as (6), but mirrored with respect to the vw-plane.

6.2.1.1 Initialization of parametric building primitives

In order to initialize a parametric primitive from the data base, the data relevant for parameter estimation have to be added to the *ORIENT* data base:

1. A room of rotation angles *ROT* has to be created. The angles are initialized by $(\omega, \phi, \kappa)_0^T = (0, 0, \kappa_0)^T$, where κ_0 can be assumed to be identical to the orientation of the previous primitive. A room of observed rotation angles is also created and initialized by the same values. As ω and ϕ are actually to be kept at 0 in order to keep the (u, v) plane of the observation co-ordinate system horizontal, the weights of the respective observations are chosen to be great, e.g. according to $\sigma_\omega = \sigma_\phi = \pm 0.1$[mgon]. The weight of the observation for κ has to be chosen somewhat smaller.

2. The vertices and the reference point $\mathbf{P}_0$ are added to the reference system. $\mathbf{P}_0$ can be initialized by the exterior reference point of the previous primitive, a constant offset being added to the planimetric co-ordinates. The object co-ordinates of the vertices can be initialized by transforming the default values of the local system into the object co-ordinate system using the initial values for $\mathbf{P}_0$ and θ.

3. For each face, a *GESTALT* room has to be created, and $\mathbf{P}_0$ and *ROT* have to be declared to be the exterior reference point and the rotation angles for all these faces.

4. For each face not being declared a symmetric one, a room of additional surface parameters has to be created, and that room has to be declared to contain the additional parameters for either u, v, or w equations of the corresponding face, depending on the type specified in the data base. Each parameter required for the face is inserted into the room of additional surface parameters. All elements of the mirror matrix are declared to be +1.

5. Each face declared to be symmetrical to another face it is declared to share its surface coefficients (its additional parameters) with that face. One or two elements of its mirror matrix are declared to be -1.

6. Each vertex contained in at least one loop of a face has to be added to the corresponding *GESTALT* room. Thus, the topology of the primitive is implicitly introduced into parameter estimation because each vertex is contained in several loops belonging to different faces.

7. At this stage, the unknown building parameters can be estimated from the default co-ordinates of the vertices.

8. Create rooms of type "observed parameters" for all surface parameter rooms and add observations for all unknown parameters. The observed values are copied from the current values of the parameters as they were estimated in step 7.

9. Create a room of type "control points" and add $\mathbf{P}_0$ to it using its default position from step 2.

As each vertex is contained in at least three surfaces, its co-ordinates can always be computed from the intersection of these surfaces. The remaining parameters (three co-ordinates of $\mathbf{P}_0$, the building orientation κ and the surface parameters, in case of the saddle back roof a^f_{00}, b^r_{00}, c^r_{00} and c^r_{10}) describe the shape of the building. For these parameters, the user has to successively provide approximate values as described in section 6.3. As long as no such information is available, the parameter observations are used in order to make these parameters determinable in adjustment.

Although the building primitive is formulated in B-rep, it is described by a minimum set of parameters which, in addition, can be interpreted easily: the building length $l = 2 \cdot |a^f_{00}|$, the building width $w = 2 \cdot |b^r_{00}|$, the building height $h = |c_{00}|$, and for the obliquity angle α of the roof we get $tan(\alpha) = |c_{10}|$. Thus, using our way of formulation, we get rid of the over-parameterization usually associated with B-rep without losing the flexibility of that form of modelling.

6.2.1.2 Parametric primitives in the data base of known primitive types

The data describing a primitive can be provided in an ASCII file. The section of that file describing the saddle back roof building in figure 6.7 is presented in table 6.1. The description of the primitive starts with the

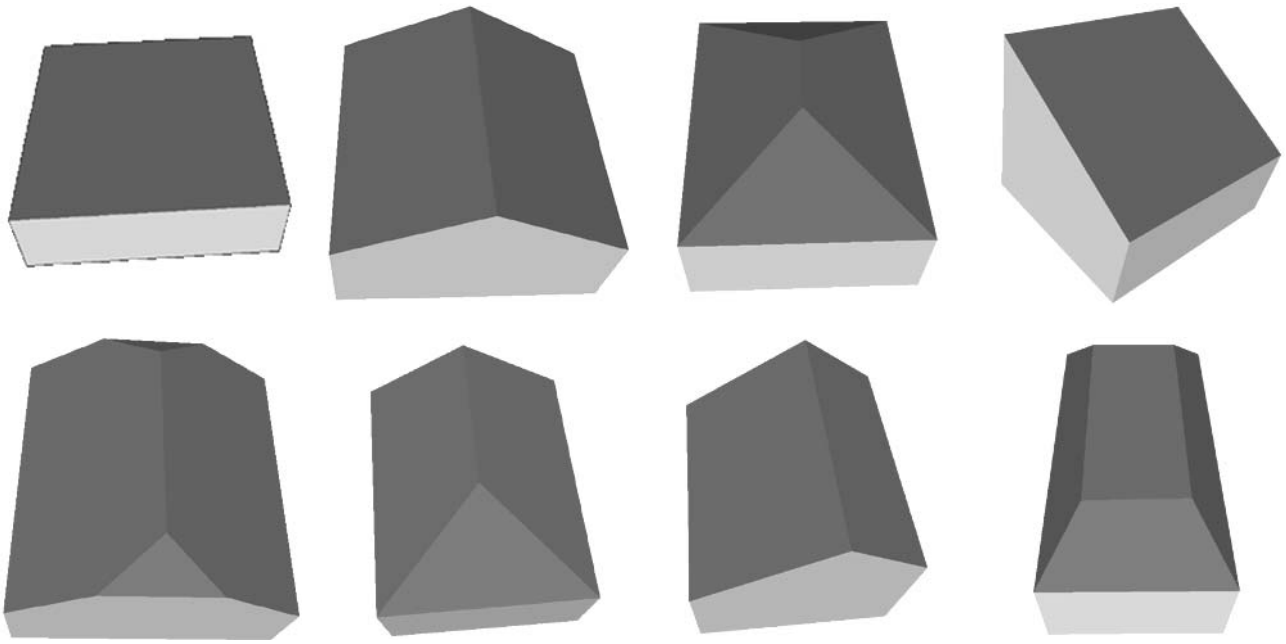


Figure 6.8: A selection of primitives. Upper row, from left to right: flat roof, saddle back roof, hip roof, tilted roof. Second row, left to right: saddle back roof with "cut-off gables", semi-hip roof, saltbox roof, mansard roof.

```
PRIMITIVE(Saddle back roof)
VERTICES (10)
 1 -4 -6  0
 2 -4  6  0
 3  4  6  0
 4  4 -6  0
 5 -4 -6  5
 6 -4  6  5
 7  4  6  5
 8  4 -6  5
 9  0 -6  8
10  0  6  8
FACES (7)
FLOOR 1 W 0 0 0           LOOPS (4;  1  2  3  4 )
WALL  2 U 1 0 0           LOOPS (4;  1  5  6  2 )
WALL  3 V 1 0 0           LOOPS (5;  2  6 10  7  3 )
WALL  4 U SYMMETRIC (U 2) LOOPS (4;  3  7  8  4 )
WALL  5 V SYMMETRIC (V 3) LOOPS (5;  4  8  9  5  1 )
ROOF  6 W 1 1 0           LOOPS (4;  5  9 10  6 )
ROOF  7 W SYMMETRIC (U 6) LOOPS (4;  7 10  9  8 )
EDGES (15)
 1  1  2  1  2
 2  1  3  2  3
 3  1  4  3  4
 4  1  5  4  1
 5  5  2  1  5
 6  2  3  2  6
 7  3  4  3  7
 8  4  5  4  8
 9  2  6  5  6 EAVES
10  4  7  7  8 EAVES
11  5  6  5  9
12  5  7  8  9
13  3  6  6 10
14  3  7  7 10
15  6  7  9 10 RIDGE
END PRIMITIVE
```

Table 6.1: A section of an ASCII file containing the data base of known building shapes. This section shows the description of the saddle back roof building in figure 6.7.

key word `PRIMITIVE` followed by the name of the primitive that is displayed to the user, and it ends with `END PRIMITIVE`. First, there is the list of vertices (10 in this case). For each vertex, a numeric identifier and default co-ordinates in the local co-ordinate system are provided. The list of vertices is followed by the list of (in this case, 7) faces. Each line corresponding to a face starts with a key word describing the semantic attribute of a face: a face is either a wall, the floor, or it belongs to the roof. This key word is followed by a numeric identifier and the type of surface observation which is to be created for each point in the corresponding *GESTALT* room. Depending on this type, this equation will be a first (`U`), second (`V`), or a third (`W`) equation 4.27. The next numbers describe the parameterization of the surface equation: three flags corresponding to the constant and the two linear parameter in the respective observation equations. If such a flag is `1`, the corresponding parameter will be used for parameterization, otherwise it will not be used. Faces which are symmetrical to another face

are marked by the key word SYMMETRIC, followed by the coefficients in the mirror matrix which are to be set to -1 and the identifier of the face to which the current face is symmetric. Each line is finished by a list of loops contained in a face. Each loop is described by the number of vertices followed by a semi-colon and the ordered list of vertices belonging to that loop. Finally, there is a list of (in this case, 15) edges. Each line describing an edge starts with the edge identifier, followed by the identifiers of the two faces intersecting at the edge and the identifiers of the starting and end vertices of the edge. The edges belonging to a building ridge or to the eaves are marked by a key word. Similar descriptions have to be provided for all parametric primitive types. Thus, adding a primitive to the data base can be performed by just editing an ASCII file.

Even though an operator is thus free to adapt the data base of known building primitives, the number of primitive types contained in that data base should not be too great. If the number of available primitives becomes too great, the data base will become difficult to survey. Many building shapes can be created by Boolean operations and face glueing of simple primitives. On the other hand, there are no reasons against adding primitives typical for a certain region to the data base if this will save work. When a new primitive type is added to the data base, care has to be taken on a sensible parameterization of the building faces. Especially with respect to roof faces, the decision whether to declare two such faces to be symmetrical has to be taken with care. Currently, our data base contains about fifteen building primitives. The most important ones are presented in figure 6.8.

6.2.2 Generic building types

Two generic building primitive types have been implemented in our system:

- *Prismatic type*: This type is considered to be a vertical prism consisting of two horizontal faces (floor and roof) and n vertical walls.
- *Rectangular prismatic type*: Another prismatic type. In addition to the specifications of the previous type, consecutive walls of the rectangular type are supposed to be orthogonal to each other.

From the definitions of these types we can see that these prisms cannot be initialized without additional information provided by the user in the form of a closed polygon digitized in one of the digital images. As soon as the polygon has been digitized, it can be propagated to object space using the orientation parameters of the image and, e.g., a default image scale. After that, its shape can be analyzed in order to create the B-rep of the primitive. As soon as this B-rep has been created, the prismatic primitives are treated in exactly the same way as the parametric primitives by our system. From the point of view of the modelling techniques described in section 2.3, the creation of a prismatic primitive can be seen as a translational sweep of the (roof) polygon by a vertical vector $\mathbf{h}$, the length of the vector being a default building height.

Again, the prism is described in a local co-ordinate system. The w-axis is vertical, and the u-axis can be chosen to be parallel to one of the polygon segments. The external reference point $\mathbf{P}_0$ can be chosen to be in one of the polygon vertices of the prism floor (cf. figure 6.9). We obtain the following formulations for the faces of the prismatic model:

1. Floor: $w_0 = 0$
2. Roof: $w_0 = c^r_{00}$
3. Wall f_1: $v_0 = 0$. This is the wall containing the u-axis.
4. Wall f_n: $u_0 = a^n_{10} \cdot v$. This is the second wall containing $\mathbf{P}_1$ and thus $\mathbf{P}_0$.
5. Wall $f_i, i \in \{2, .., n-1\}$: depending on the largest component of the normal vector $\mathbf{n}$ in the local co-ordinate system, f_i will be formulated as:
   (a) $u_0 = a^i_{00} + a^i_{10} \cdot v$ if u is the largest component

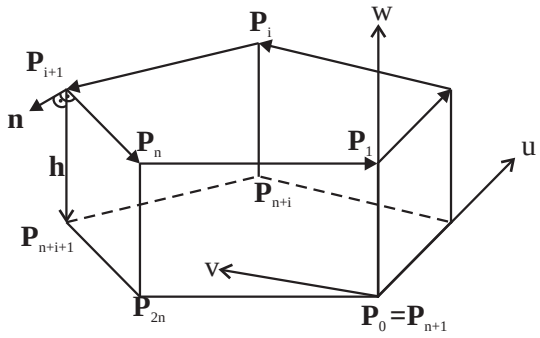


Figure 6.9: A prismatic primitive consisting of five walls.

(b) $v_0 = b^i_{00} + b^i_{10} \cdot u$ if v is the largest component.

For the rectangular prism the same rules for the definition of the local co-ordinate system can be applied, but the faces are described in a simpler form in order to enforce orthogonality. We obtain the following formulations for its faces:

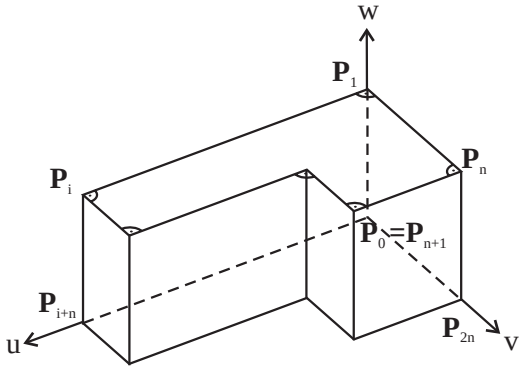


Figure 6.10: A rectangular prism consisting of six walls.

1. Floor: $w_0 = 0$
2. Roof: $w_0 = c^r_{00}$
3. Wall f_1: $v_0 = 0$. This is the wall containing the u-axis.
4. Wall f_n: $u_0 = 0$. This is the second wall containing $\mathbf{P}_1$ and thus $\mathbf{P}_0$.
5. Wall $f_i, i \in \{2, .., n-1\}$: again, there are two possible formulations:
   (a) $u_0 = a^i_{00}$ for $i = 2 \cdot k$
   (b) $v_0 = b^i_{00}$ for $i = 2 \cdot k + 1$.

6.2.2.1 Initialization of generic building types

A generic primitive can be initialized as soon as the roof has been digitized as a closed polygon in one of the displayed images. First, the polygon has to be propagated to object space and analyzed with respect to its shape:

1. Transform the polygon vertices to object space using the image's orientation parameters together with some information about the "local image scale" to determine an initial roof height.
2. Check the order of the polygon vertices and reverse order in case the polygon is ordered clockwise if seen from above.
3. Change the order of the vertices of the closed polygon by cyclic permutation of the vertices so that the first vertex $\mathbf{P}_1$ fulfils the criteria for the exterior reference point $\mathbf{P}_0$: It has to be a point of intersection of two long and almost orthogonal polygon edges.

After that, the B-rep has to be created:

1. The roof polygon is added to the B-rep first. The corresponding face is flagged to be the roof. The roof edges are flagged to be eaves.

2. The floor polygon is added to the B-rep. The corresponding face is flagged to be the floor. The floor vertex corresponding to roof vertex $\mathbf{P}_i$ is called $\mathbf{P}_{n+i}$, where n is the number of polygon vertices. The orientation of the floor loop has to be different from the orientation of the roof loop.

3. For each polygon segment $i, i \in \{1, .., n\}$, a face corresponding to a wall has to be added to the B-rep. The outer loop of that face consists of four vertices ordered by $(\mathbf{P}_i, \mathbf{P}_{n+i}, \mathbf{P}_{n+i+1}, \mathbf{P}_{i+1})$ for $i \in \{1, .., n-1\}$ and $(\mathbf{P}_n, \mathbf{P}_{2n}, \mathbf{P}_{n+1}, \mathbf{P}_1)$ for $i = n$. These faces are flagged as being walls.

4. For each wall face, determine the type of formulation (i.e, either a u- or a v- equation). The way this is done depends on the type of the primitive in the way described above.

As soon as the B-rep has been created, the data relevant for parameter estimation have to be added to the *ORIENT* data base. This is done in exactly the same way as for parametric types (cf. section 6.2.1.1), the differences being that $\mathbf{P}_{n+1}$ is used as the exterior reference point $\mathbf{P}_0$ instead of a new point not being a building vertex, and κ is initialized by evaluating the orientation of the vector $\overline{\mathbf{P}_1\mathbf{P}_2}$ in the object co-ordinate system.

6.3 Interactive determination of approximate building parameters

As soon as a primitive has been selected from the data base of known building shapes, approximations for the primitive parameters have to be determined interactively. Typically, two images will be open for visual determination of approximations, and a wire frame representation of the building will be super-imposed to the images using the current primitive parameters (cf. figure 6.4). In section 6.2, we have described how the primitives were initialized. We have seen that the co-ordinates of the vertices can be determined from surface observations, and additional information has to be provided by the user in order to determine the position, the orientation and the shape parameters of the building. In order to make these parameters also determinable in adjustment, an observation was added for each parameter.

The user can update the current values of the parameters by interactively identifying building vertices in the digital images. After each of these user interactions, the building parameters have to be updated taking into account the new piece of information provided by the user and using default values for the parameters which cannot yet be determined. The new values of the parameters will be estimated from:

- the surface observations of the building primitive for the vertices,
- the observations for all parameters, and
- the perspective observations for the camera co-ordinates of all vertices the user has already measured in the images.

In this context it is the main problem to find out which building parameter(s) can be determined by the new image point provided by the operator. This problem is solved by applying robust estimation to the parameter observations. The operations to be performed for updating the primitive parameters interactively is presented in figure 6.11:

1. The operator has to select the building vertex which is to be measured next. By default, one roof vertex after the other will be proposed for measurement by the system. This default can be overridden by selecting a vertex in the graphical sketch of the primitive type (cf. figure 6.3).

2. The selected vertex is identified interactively in one of the displayed images, which results in the two camera co-ordinates of the measured point to be inserted into the corresponding *PHOTO* room in the *ORIENT* data base. Two perspective observation equations will be inserted into adjustment for the new image point.

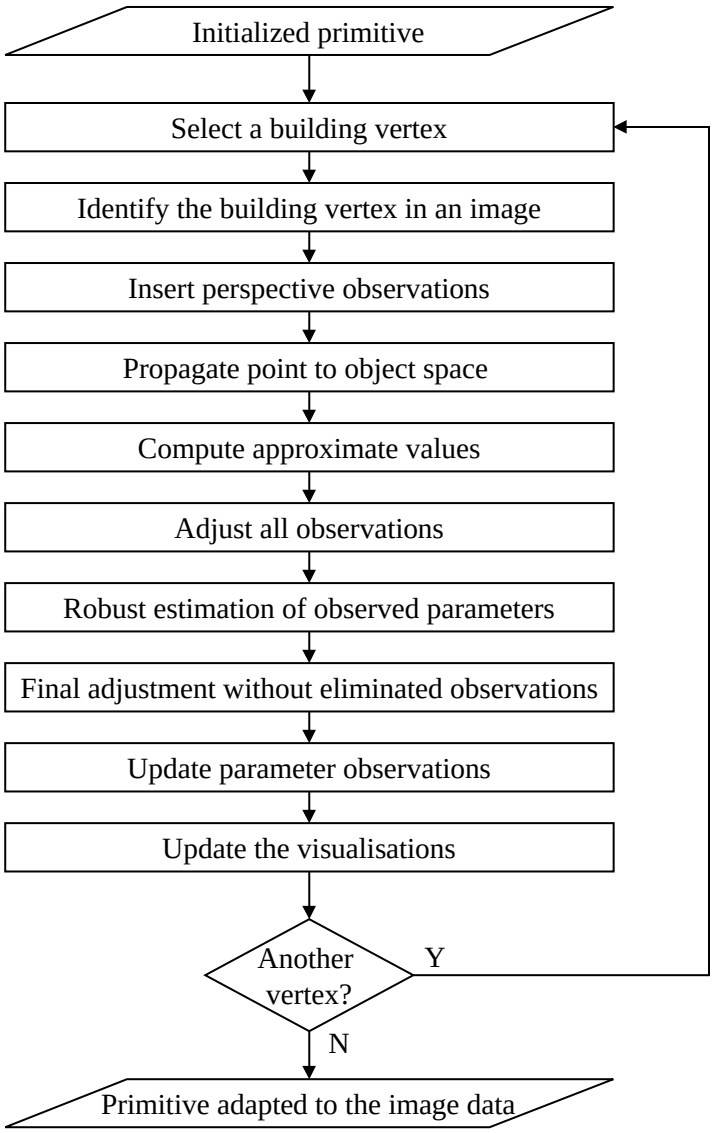


Figure 6.11: The work flow for interactive editing of the primitive parameters.

3. The image point is propagated to object space. If the respective vertex has not been measured in another image, this is done using a default local image scale, otherwise the object point can be computed from spatial intersection. In the latter case, the local image scales of the vertex are stored in order to use them for propagation of the next point.

4. The object co-ordinates of the vertex are used to determine approximate values for the unknown building parameters. The shape parameters are not critical in this context as long as reasonable defaults are used. The same holds true for the vertex co-ordinates because they can always be computed from the surface observations: if all the other parameters are kept fixed, the observation equations 4.27 are linear with respect to the object co-ordinates of the vertices. That is why the only critical parameters are the co-ordinates of $\mathbf{P}_0$ and κ. From the difference of the new object co-ordinates of the vertex and its old ones, a shift vector can be computed which is used to update $\mathbf{P}_0$ and the corresponding control point. In order to obtain an approximate value for κ, an "observation count" for each vertex is required. This count is increased by one as soon as the vertex is identified in a new image. Using this observation count, the number of vertices already having been measured in at least one image can be derived. If this number is greater than one, the rotational angle κ can be computed from a comparison of the vector between two such vertices in both the local and the object co-ordinate systems.

5. Several steps of adjustment have to be performed using the approximate values from step 4 in order to estimate building parameters and building vertices from all available observations. Note that at this stage, there will be contradictions between the observations for the primitive parameters and the new information by the user (the new perspective observations) which will result in some observations having great residuals. Only contradicting observations obtain large residuals in adjustment because observations not being checked by other ones are required to determine a certain parameter, and their residuals will be close to zero.

6. This behaviour is used to find out which building parameter(s) can be determined by the new point. Robust estimation is performed in the way described in section 4.1.1.1, but the re-weighting scheme is only applied to the observed parameters. After a few iterations, the observed parameters contradicting the information provided by the user are detected to be gross errors, and they will be eliminated from adjustment. After that, the new values of the primitive parameters are estimated from all observations without those observed parameters marked as gross errors.

7. The parameter observations are updated using the estimated values so that they can be used in a further estimation process. After that, the wire frame of the primitive is updated in all displayed images using the new parameter values.

8. This procedure is repeated until the operator considers the building to fit well enough to the image data. This decision can be made on the basis of a visual inspection of the super-imposed wire frames.

The main control parameters of the process of estimating the building parameters in the way described above are the a priori r.m.s. errors of all observations involved in parameter estimation as they define the stochastic model of adjustment. All these parameters can be set by the user. These parameters comprise:

- The r.m.s. error σ_s of a surface observation of a vertex. This parameter describes the statistical deviation of the actual building from the B-rep model. By default, σ_s is chosen to be $\pm$ 2 cm.

- The r.m.s. error σ_i of a perspective observation of a vertex. This parameter describes the measuring accuracy. By default, σ_i is chosen to be $\pm$ 1 pixel.

- The r.m.s. error σ_p of an observed parameter. This r.m.s. error depends on the parameter type. By default, it is chosen to be $\pm$ 0.1 m for control points and constant offsets, $\pm$ 10% for linear shape parameters, $\pm$ 0.1 mgon for ω and ϕ and $\pm$ 1 gon for κ. With the exception of ω and ϕ, these parameters are just required to prevent singularities. That is why they should be chosen in a way as not to give these observations too much influence.

The user is free to decide when to stop interactive editing. The quality of the measured image points can also vary: If the user just wants to provide approximate values for automatic fine measurement, the points can be measured rather coarsely. On the other hand, if automatic fine measurement is expected to fail, e.g., due to low contrast or due to noise, the building vertices can be measured quite accurately in the images, and the results of interactive editing can already be accepted as the final ones (cf. section 6.1.1).

Figure 6.12 shows an example for interactive determination of the parameters of a building. The upper row shows the right image, and the lower row shows the left one. The leftmost column shows the situation after the first image point has been measured in the left image. By these two image co-ordinates, the reference point $\mathbf{P}_0$ can be placed along the image ray, thus its planimetric co-ordinates can be determined, but not its height. $\mathbf{P}_0$ will be placed along the image ray, its Z co-ordinate being still determined by the default observation. As that default is not a very good one in the example, the wire frame is quite far away from the correct position in the right image. After measuring the homologous point in the right image, that point can be determined by spatial intersection. $\mathbf{P}_0$ can now be determined in object space (second column in figure 6.12). Measuring a second point in an image will determine κ and one of the building parameters, in this case d_{00}^f (third column in figure 6.12). As described above, in this case it is necessary to compute a better approximation

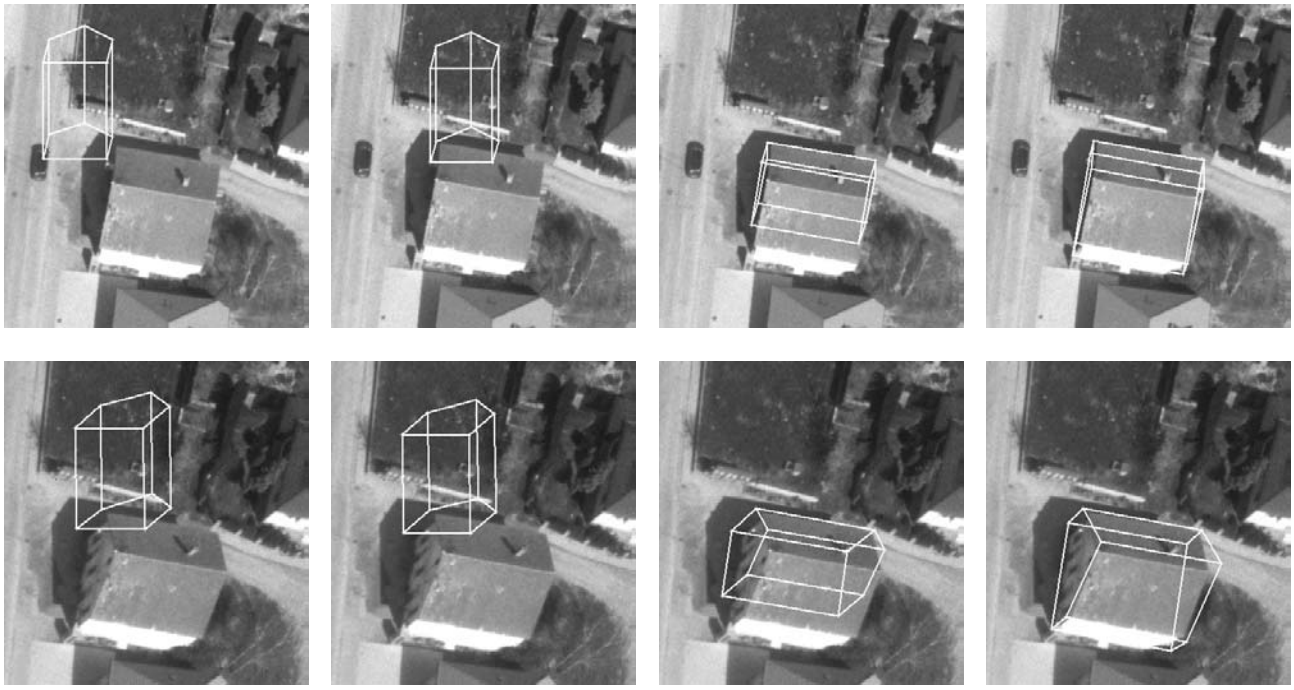


Figure 6.12: Determination of approximations. Upper row: right image, lower row: left image. First (leftmost) column: one vertex has been determined in the left image. Second column: the same vertex has been identified in the right image. Third column: a second vertex has been measured in the left image. Fourth column: a third vertex has been measured in the left image.

for κ first because otherwise adjustment would not converge. Finally, after measuring a third point in one of the images, another parameter (b'_{00}) can be determined (fourth column in figure 6.12). Note that this principle works independently of the order in which the points are measured by the operator. In addition, there are no restrictions with respect to the number of points which have to be measured interactively. However, in the example the model is already placed well enough so that the automated modules will work.

The principle of identifying building vertices in the images brings about problems in cases when these vertices are invisible, e.g., due to occlusions. The principle described in this section can be easily expanded to interactive measurement of points on the building edges. In this case, in addition to the perspective observations, two surface observations have to be inserted into adjustment in the work flow described above, and some rules for the determination of approximate values for κ have to be defined. This expansion will be implemented in the future.

6.4 Automatic fine measurement

As soon as it is requested by the user, fine measurement of the building parameters is performed automatically. Automatic fine measurement is a specific application of our framework for object surface reconstruction described in section 5.4. The object to be reconstructed is the building primitive, which is modelled in the way described in section 6.2, and the approximate values for the primitive parameters have been provided in the way shown in section 6.3. Object reconstruction is performed iteratively in several pyramid levels starting at an image resolution selected by the user. The work flow has already been described in section 5.4.1 (cf. especially the flow chart in figure 5.20). At each pyramid level, correspondence analysis has to be performed. As the topological structure of the building primitive is completely known, a model driven strategy in the sense described in section 5.4.1.3 can be applied for that purpose. The correspondence analysis is guided by the following principles:

- At each pyramid level, image edges in the sense described in section 5.1.4 are matched with the roof edges of the building model (i.e., all edges having at least one neighbouring face flagged as a roof).
- The generation of correspondence hypotheses is performed independently for all roof edges.
- For each roof edge, the generation of correspondence hypotheses is performed independently in all images.
- The evaluation of the correspondence hypotheses is performed in an overall robust hybrid adjustment in the way described in section 5.4.1.3.

The coarse-to-fine approach using image pyramids is necessary to improve the radius of convergence of the method: in the higher levels of the image pyramids, the approximate positions of the projected building edges are only a few pixels away from the actual image edges. In addition, by lowpass filtering which is used to produce the image pyramids, only the most salient image edges will survive whereas in the lower pyramid levels, more details and noise influences are available which are candidates for false matches. We also want to emphasize the benefits of using more than two images for matching. Due to shadow effects and occlusions, there are always object edges which are not visible in one or more images. Using a greater number of images helps to overcome that problem.

In this section, we will discuss the task-specific aspects of both the generation (section 6.4.1) and the evaluation of correspondence hypotheses (section 6.4.2). In section 6.4.3, we will sum up the control parameters and quality measures of automatic fine measurement. Examples for the application of automatic fine measurement and an evaluation of its performance will be presented in chapter 7.

6.4.1 Generation of correspondence hypotheses

In order to generate hypotheses of correspondence for one roof edge e in one image, its two adjacent vertices $\mathbf{V}_1$ and $\mathbf{V}_2$ are transformed to the camera co-ordinate system (u, v) using equation 4.24 applied to the approximated object co-ordinates of the vertices, and from there they are transformed to to the scanner co-ordinate system (r, c) using the inversion of equation 4.26. The approximated positions of the vertices in the scanner co-ordinate system are $\mathbf{v}_1^o = (r_1^o, c_1^o)^T$ and $\mathbf{v}_2^o = (r_2^o, c_2^o)^T$, and they are the limits of the approximate edge e^o in the scanner co-ordinate system (cf. figure 6.13). In the scanner co-ordinate system, the approximate edge e^o is then described by a straight line equation in analogy to equation 5.12, the only difference being that the co-ordinates need not be reduced to the co-ordinates of the centre of gravity. The normal vector $\mathbf{n}_e$ of e^o and the distance ρ of the straight line through $\mathbf{v}_1^o$ and $\mathbf{v}_2^o$ from the origin of the scanner co-ordinate system can be derived from:

$$\mathbf{n}_e = \frac{1}{\sqrt{(r_2^o - r_1^o)^2 + (c_2^o - c_1^o)^2}} \cdot \begin{pmatrix} c_2^o - c_1^o \\ r_1^o - r_2^o \end{pmatrix} = \begin{pmatrix} \cos\varphi \\ \sin\varphi \end{pmatrix} \qquad \text{and} \qquad \rho = \mathbf{n}_e \cdot \mathbf{v}_1^o \tag{6.1}$$

Using $\mathbf{n}_e$ and ρ from equation 6.1, the distance $d_{e,p}$ of a point $\mathbf{p} = (r_p, c_p)^T$ from the approximated edge e^o is given by:

$$d_{e,p} = \begin{cases} ||\overline{\mathbf{v}_1^o\mathbf{p}}|| & \forall \mathbf{p} \quad | \quad \overline{\mathbf{v}_1^o\mathbf{v}_2^o} \cdot \overline{\mathbf{v}_1^o\mathbf{p}} < 0 \\ |r_p \cdot \cos\varphi + c_p \cdot \sin\varphi - \rho| & \forall \mathbf{p} \quad | \quad 0 \le \overline{\mathbf{v}_1^o\mathbf{v}_2^o} \cdot \overline{\mathbf{v}_1^o\mathbf{p}} \le \left\|\overline{\mathbf{v}_1^o\mathbf{v}_2^o}\right\|^2 \\ \left\|\overline{\mathbf{v}_2^o\mathbf{p}}\right\| & \forall \mathbf{p} \quad | \quad \left\|\overline{\mathbf{v}_1^o\mathbf{v}_2^o}\right\|^2 < \overline{\mathbf{v}_1^o\mathbf{v}_2^o} \cdot \overline{\mathbf{v}_1^o\mathbf{p}} \end{cases} \tag{6.2}$$

where $||\overline{\mathbf{p}_1\mathbf{p}_2}||$ is the Euclidean distance between $\mathbf{p}_1$ and $\mathbf{p}_2$. The search space for image edges corresponding to the edge e is then reduced to all points $\mathbf{p}$ having a distance smaller than a user-defined threshold ε from the approximate position e^o of that edge in the image (the light grey area in figure 6.13). Let $L = \{l_1, l_2, \ldots, l_{n_l}\}$

be the set of extracted image edges at the current pyramid level. As we have seen in section 5.1.4, each image edge $l_k \in L$ consists of n_p polygon vertices $\mathbf{p}_{k,i}$: $l_k = \{\mathbf{p}_{k,1}, \mathbf{p}_{k,2} \ldots, \mathbf{p}_{k,n_p}\}$ and their variance-covariance matrices $\mathbf{C}_{rc_{k,i}}$ from equation 5.22. Using the approximate values, a subset $L^c \subseteq L$ containing the candidate image edges for correspondence can be found:

$$L^c = \Big\{ l_k \in L | \quad \exists \mathbf{p}_{k,i} \in l_k | \quad d_{e,p_{k,i}} \leq \varepsilon \Big\} \tag{6.3}$$

Equation 6.3 means that all image edges l_k having at least one polygon vertex $\mathbf{p}_{k,i}$ which is inside the search area in vicinity of the approximate position e^o of the edge in the image are considered to be candidate edges for matching. For all candidate edges $l_k^c \in L^c$, correspondence analysis is carried on further for the straight line segments $s_{k,i}$ which are the connections between polygon vertices $\mathbf{p}_{k,i}$ and $\mathbf{p}_{k,i+1}$ of l_k^c. A segment $s_{k,i}$ of a candidate edge $l_k^c \in L^c$ is supposed to correspond to edge e if two conditions are fulfilled:

1. $s_{k,i}$ is approximately parallel to e^o, thus the angle α between the normal vectors $\mathbf{n}_e$ of the edge and $\mathbf{n}_{k,i}$ of $s_{k,i}$ is smaller than a certain threshold ε_α:

$$\cos\alpha = |\mathbf{n}_{k,i} \cdot \mathbf{n}_e| \geq \cos\varepsilon_\alpha \tag{6.4}$$

   where $\mathbf{n}_{k,i}$ is normalized, thus $||\mathbf{n}_{k,i}|| = 1$ and $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$.

2. At least one of the endpoints $\mathbf{p}_{k,i}$ and $\mathbf{p}_{k,i+1}$ of $s_{k,i}$ has to be inside the search area, thus:

$$\Big(d_{e,p_{k,i}} \leq \varepsilon\Big) \vee \Big(d_{e,p_{k,i+1}} \leq \varepsilon\Big) \tag{6.5}$$

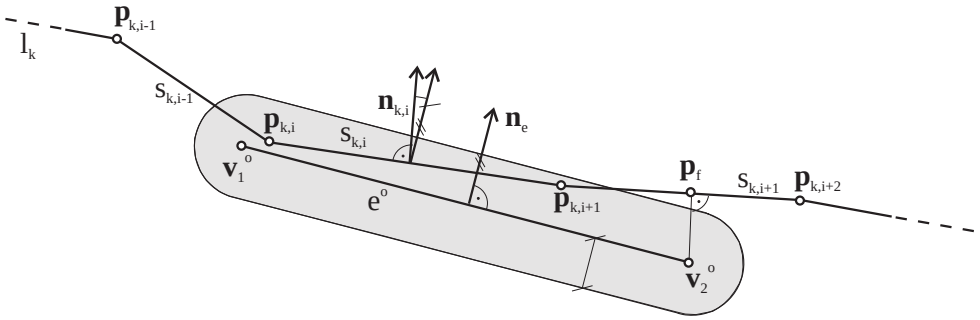


Figure 6.13: Generation of correspondence hypotheses for one edge in one image. e^o: approximate position of the edge in the scanner co-ordinate system. The light grey area gives the search area in the vicinity of e^o containing all points $\mathbf{p}$ with $d_{e,p} \leq \varepsilon$. l_k: a candidate image edge.

If the orthogonal projection of one or both of the vertices $\mathbf{v}_1^o$ and $\mathbf{v}_2^o$ to the straight line defined by the end points $\mathbf{p}_{k,i}$ and $\mathbf{p}_{k,i+1}$ of a candidate segment $s_{k,i}$ fulfilling these conditions is situated between $\mathbf{p}_{k,i}$ and $\mathbf{p}_{k,i+1}$, $s_{k,i}$ has to be reduced so that only the common parts of the projection of e^o and the original segment remain inside. For instance, for segment $s_{k,i+1}$, the end point $\mathbf{p}_{k,i+2}$ has to be replaced by the projection $\mathbf{p}_f$ of $\mathbf{v}_2^o$ to $s_{k,i+1}$. At this instance, the variance-covariance matrices of the segment end points have to be re-computed: As we have seen in section 5.1.4, the variance-covariance matrix $\mathbf{C}_{rc}$ of a polygon vertex of an image edge depends on the lengths of the neighbouring image edge segments. By the projection, the length of the segment is reduced, so that the elements of $\mathbf{C}_{rc}$ have to be re-computed.

For each end point $\mathbf{p}_e \in \{\mathbf{p}_{k,i}, \mathbf{p}_{k,i+1}\}$ of each (reduced) candidate segment $s_{k,i}$, the following procedure is applied:

1. A new identifier is created and assigned to $\mathbf{p}_e$.

2. A new point corresponding to $\mathbf{p}_e$ is inserted into the reference system in the *ORIENT* data base, its object co-ordinates being initialized by dummy values.

3. $\mathbf{p}_e$ is inserted into the two *GESTALT* rooms corresponding to the neighbouring faces of e. Thus, two surface observations for $\mathbf{p}_e$ will be inserted into adjustment.

4. $\mathbf{p}_e$ is transformed to the camera co-ordinate system (u, v), and the variance-covariance matrix $\mathbf{C}_{uv}$ has to be computed from $\mathbf{C}_{rc}$ by the law of error propagation. After that, $\mathbf{p}_e$ is inserted into the *PHOTO* room corresponding to the current image. Thus, two perspective observations for $\mathbf{p}_e$ will be inserted into adjustment. At this instance, the off-diagonal element of $\mathbf{C}_{uv}$ is neglected because *ORIENT* only can handle diagonal weight matrices.

This means that we get 4 observations for each end point of each candidate segment:

- 2 camera co-ordinates and
- 2 surface observations.

Each end point of a segment adds 3 unknowns to the adjustment, the point's object co-ordinates. As each end point adds 4 observations and 3 unknowns to the adjustment, it increases redundancy by 1. Note that an image edge vertex might be added to adjustment twice (e.g. $\mathbf{p}_{k,n+1}$ in figure 6.13). As in the stochastic model of an image edge vertex, the length of the segment used for computing the straight line approximation of the edge element chain (cf. section 5.1.4, especially equations 5.20 and 5.21), the end points of long image edge segments will obtain more influence on the adjustment results than short ones, which is desirable because it reduces the influence of small short image edges, e.g. the edges of windows which are by chance visible in some of the images. The stochastic properties of the surface observations is described by the a priori r.m.s. error σ_s of a surface observation. σ_s has to be selected with care as it describes the "rigidity" of the building model in reconstruction. If it is chosen too small, correct hypotheses will be eliminated from adjustment because the symmetry conditions of the building are enforced too strictly. If σ_s is chosen too great, wrong hypotheses might not be detected because their variation from the model can be explained by the stochastic properties of the model.

6.4.2 Evaluation of correspondence hypotheses

As stated above, the matching procedure is applied to all object edges in all images. Thus, we obtain an enormous redundancy which should render possible the elimination of false matches. The edges are not adjusted separately, so that they all influence each other. The number of observations used to determine one single building primitive may be high (up to several hundred). The number of outliers is kept small by hierarchical processing on the basis of image pyramids.

Robust estimation is applied for hypotheses verification as described in section 5.4.1.3, the re-weighting scheme being applied to the image observations only. Using the strategy described in section 4.1.1.1, in the current implementation, $n_s = 5$ observations are suspected to be gross errors at each adjustment iteration. All the observations inserted into the *ORIENT* data base are adjusted simultaneously. The observations are:

1. The camera co-ordinates and the surface observations of the end points $\mathbf{P}_i$, $i \in \{1, \ldots, n_h\}$, of the candidate image edge segments (cf. section 6.4.1).

2. The surface observations of the vertices $\mathbf{v}_1, \ldots, \mathbf{v}_{n_v}$ of the building primitive. These observations implicitly represent the topology of the primitive in adjustment.

3. Two observed rotational angles for ω and ϕ. These observations are given a very high weight as they are just a substitute for keeping these angles constant.

4. One control point observation for Z_0, i.e., the height of the exterior reference point $\mathbf{P}_0$. This observation is required because the building floor is not used for matching so that Z_0 cannot be determined from the matching results. Again, the weight of this observation is chosen quite great.

Figure 6.14 shows the structure of the normal equation matrix for adjustment. The unknowns are:

1. The object co-ordinates of the n_v vertices $\mathbf{v}_1, \ldots, \mathbf{v}_{n_v}$ of the building primitive. These unknowns correspond to a sub-matrix of non-zero elements of size $(3 \cdot n_v) \times (3 \cdot n_v)$.
2. The object co-ordinates of the end points $\mathbf{P}_i$, $i \in \{1, \ldots, n_h\}$, of the candidate image edge segments (cf. section 6.4.1). Each of these points corresponds to a sub-matrix of non-zero elements of size 3×3.
3. The three co-ordinates of the exterior reference point $\mathbf{P}_0$ of the building model.
4. The three rotational angles $\theta = (\omega, \phi, \kappa)^T$.
5. The unknown building shape parameters $\mathbf{ADP}$.

In figure 6.14, the non-zero elements of the normal equation matrix $\mathbf{N}$ are shown in grey. The total number of unknowns u is $u = 3{\cdot}n_v{+}3{\cdot}n_h{+}3{+}3{+}n_{ADP}$. In the case of the saddle back roof building (cf. figure 6.7), this evaluates to $u = 3 \cdot 10 + 3 \cdot n_h + 3 + 3 + 4 = 40 + 3 \cdot n_h$. As stated above, the number n_h of hypothesized edge points may be very high. However, using the *ORIENT*'s sparse-matrix technique [Gsandtner and Kager, 1988], the zero elements are eliminated from the solution of the normal equations so that only sub-matrices of 3×3 non-zero elements and the blocks combining these points with the other unknowns have to be considered which considerably reduces the computational costs.

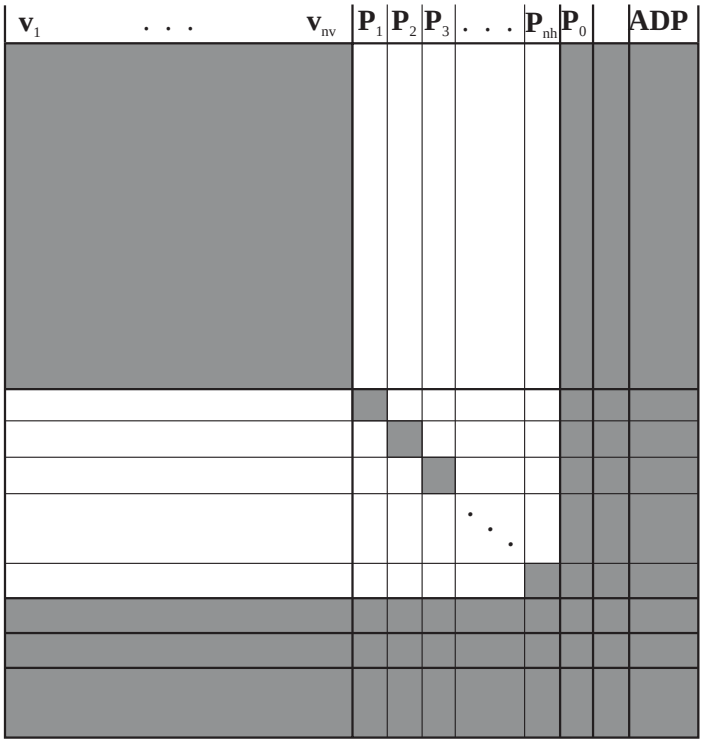


Figure 6.14: The structure of the normal equation matrix $\mathbf{N}$. White areas are filled by zeroes, grey areas are filled by non-zero elements.

In order to speed up computation, a preliminary elimination of the unknown object point co-ordinates is desirable. This could be achieved by using a *new observation type* for image edges in *ORIENT*. In its simplest type,

the object edge could be described as a straight line segment between two points $\mathbf{P}_1$ and $\mathbf{P}_2$ in object space. The normal vector $\mathbf{n}_{img} = (u_n, v_n, w_n)^T$ of the plane defined by $\mathbf{P}_1$, $\mathbf{P}_2$, and the projection centre $\mathbf{P}_0^{img}$ of the image is given by:

$$\mathbf{n}_{img} = \begin{pmatrix} u_n\left(\mathbf{R}^{img}, \mathbf{P}_0^{img}, \mathbf{P}_1, \mathbf{P}_2\right) \\ v_n\left(\mathbf{R}^{img}, \mathbf{P}_0^{img}, \mathbf{P}_1, \mathbf{P}_2\right) \\ w_n\left(\mathbf{R}^{img}, \mathbf{P}_0^{img}, \mathbf{P}_1, \mathbf{P}_2\right) \end{pmatrix} = \mathbf{R}^{img^T} \cdot \left[\left(\mathbf{P}_1 - \mathbf{P}_0^{img}\right) \times \left(\mathbf{P}_2 - \mathbf{P}_0^{img}\right)\right] \tag{6.6}$$

in the image co-ordinate system of the image img. In equation 6.6, $\mathbf{R}^{img}$ is the rotational matrix attached to that image, just as $\mathbf{P}_0^{img}$ is its projection centre. Using $\mathbf{n}_{img}$ from equation 6.6, an observation for the orthogonal distance d_p^{img} of an image point $\mathbf{p} = (u, v)^T$ from the image edge corresponding to the object edge between $\mathbf{P}_1$ and $\mathbf{P}_2$ can be formulated as:

$$d_p^{img} + v_d = 0 + v_d = \frac{u_n \cdot u + v_n \cdot v - u_n \cdot u_0^{img} - v_n \cdot v_0^{img} - w_n \cdot f^{img}}{\sqrt{u_n^2 + v_n^2}} \tag{6.7}$$

with $\mathbf{p}_0 = (u_0^{img}, v_0^{img}, f^{img})^T$ being the interior reference point of the image img. As the point is considered to be situated on the straight image edge corresponding to the object edge, the distance is fictitiously observed to be 0 in equation 6.7. Using this formulation, only one distance observation would be inserted for each end point of a candidate image edge segment. As in our formulation in section 6.4.1, redundancy would be increased by one, but no additional unknowns would have to be inserted into adjustment. Thus, for instance, in the case of the saddle back roof building, the number of unknowns would be reduced to 40. Fictitious distance observations are used in a similar way in [Ameri, 2000, Lang, 1999, Veldhuis, 1998]. In our system, this observation type is not yet available.

6.4.3 Control parameters and quality measures

To sum up the contents of the previous sections, the control parameters of automatic fine measurement are:

- The control parameters of feature extraction (cf. section 5.4.1.1). These parameters have to be specified by the user.
- The pyramid level l_{start} at which the hierarchical matching process is to start. l_{start} has to be specified by the user. By default, for images scanned at 15 μm, l_{start} is chosen to be 3.
- The stochastical model of adjustment:
  - The a priori r.m.s. errors of the camera co-ordinates of the end points of the candidate image edge segments. They are derived from error propagation applied to the results of straight line fitting in the way described in section 5.1.4.
  - The a priori r.m.s. error σ_s of the surface observations of both the end points of the candidate image edge segments and the vertices of the building primitive. σ_s can be specified by the user. The default value is $\sigma_s = \pm 2$[cm]
  - The a priori r.m.s. errors σ_ω and σ_ϕ of the observed rotation angles. These r.m.s. errors are chosen to be $\sigma_\omega = \sigma_\phi = \pm 0.1$[mgon].
  - The a priori r.m.s. error σ_{Z_0} of the control point observation for Z_0. It is chosen to be $\sigma_{Z_0} = \pm 0.1$[mm].
- The threshold ε delimiting search space for correspondence analysis at the current pyramid level (cf. section 6.4.1). This threshold can be specified by the user. By default, ε is chosen to be 2 pixels in pyramid level i, which corresponds to $\varepsilon \cdot 2^i$ pixels in pyramid level 0.

- The threshold ε_u delimiting search space for correspondence analysis in the upper pyramid level l_{start}. This threshold can be specified by the user; by default, it is chosen to be 5 pixels. The reason for using another threshold ε_u in pyramid level l_{start} than for the other pyramid levels is given by the fact that ε_u defines the overall search radius of the procedure. Thus, ε_u can be increased in order to increase the overall search radius without increasing ε in the other pyramid levels, which is not necessary in case automatic fine measurement converges. The overall size of search space corresponds to $\varepsilon_u \cdot 2^{l_{start}}$ pixels on level 0 of the image pyramids, which evaluates to 40 pixels for the default values.

- The threshold ε_α for the parallelity condition in correspondence analysis (cf. section 6.4.1). In the current version, a fixed threshold for $|\cos \varepsilon_\alpha|$ of 0.7 is used, which corresponds to $\varepsilon_\alpha \simeq 45°$.

- The threshold h_{min} for parameter h of the weight function in the iterative strategy for robust estimation described in section 4.1.1.1. This means that the largest residual of an observation still contained in adjustment after robust estimation will be h_{min} times the observation's a priori r.m.s. error. h_{min} is chosen to be 3.

All the quality measures described in section 5.4.1.3 are derived after adjustment. The most important ones for self-diagnosis are the a posteriori r.m.s. error of the weight unit $\hat{\sigma}_o$ (equation 4.8), the redundancy of adjustment after elimination of the false observations and the results of a coverage analysis for the building edges. For the coverage analysis, the percentage of support is computed for each roof edge of the building primitive. For each roof edge, an accumulator having 100 entries is created, and its elements are initialized by 0. For each candidate image edge segment with at least one end point not eliminated in robust estimation, the following procedure is applied: Let $l_e^n = \|\overline{\mathbf{V}_1\mathbf{V}_2}\|$ be the length of the roof edge n in object space, i.e., the Euclidean distance of its end points $\mathbf{V}_1$ and $\mathbf{V}_2$ in object space. Then, two indices $index_k, k \in \{1, 2\}$ are computed from the object co-ordinates of the end points $\mathbf{P}_1$ and $\mathbf{P}_2$ of the image edge segment:

$$index_k = 100 \cdot \frac{\overline{\mathbf{V}_1\mathbf{P}_k} \cdot \overline{\mathbf{V}_1\mathbf{V}_2}}{l_e^n} \tag{6.8}$$

$index_k$ gives the position of $\mathbf{P}_k$ along the roof edge in units of the edge length in [%]. After that, all elements of the accumulator having entries between $index_1$ and $index_2$ are incremented. If one of the end points of the image edge segment was eliminated in adjustment, its corresponding index will be set to $0.5 \cdot (index_1 + index_2)$, thus, if only one end point is eliminated, the remaining end point is considered to correspond to half the segment, which appears to be natural if we consider the way the stochastical properties of the corresponding image points were derived. As soon as the accumulator has been filled by all "surviving" image edge segments corresponding to edge n, the coverage c_n of the roof edge n is given by the number of accumulator cells containing a value greater than 0. After that, an overall coverage c_{all} can be computed from the weighted sum of all c_n:

$$c_{all} = \frac{\sum\limits_n l_e^n \cdot c_n}{\sum\limits_n l_e^n} \tag{6.9}$$

In chapter 7, examples for the behaviour of the automatic fine measurement tool in dependence of the control parameters will be given.

Chapter 7

Experiments

In this chapter, we want to present a test project which was carried out in the village of Stoitzendorf in Lower Austria. A small aerial block of two strips (image scale: 1:4500, camera constant: 150 mm) with 70% overlap and 50% side lap was photographed. The images were scanned at a resolution of 15 μm, which corresponds to about 7 cm in object space. The configuration of the block guarantees each part of the village to be visible in at least three images. Most buildings are even visible in six images.

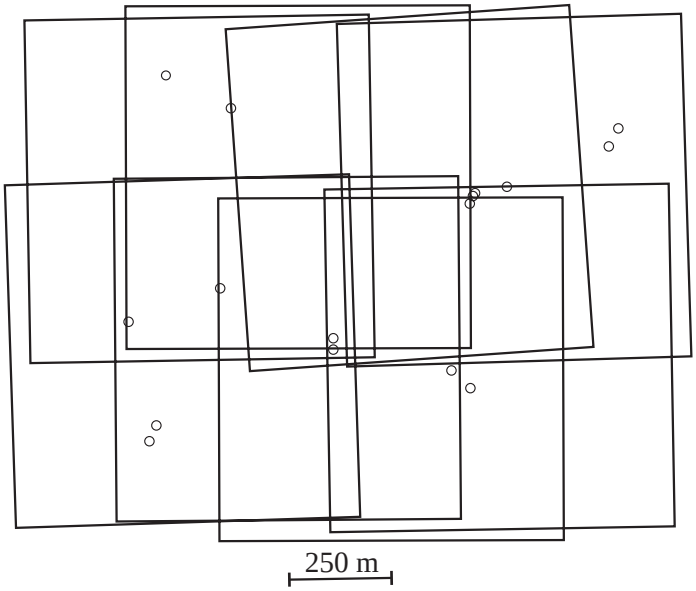


Figure 7.1: Flight overview over the block used for the evaluation of semi-automatic building extraction. The figure contains the footprints of eight photographs in two strips and the distribution of control points.

The orientation parameters of the images were determined by aerial triangulation using *ORPHEUS*. A total number of 16 control points was determined with an accuracy of $\pm$2 cm using GPS. These control points as well as 44 tie points were measured interactively in *ORPHEUS*. The theoretical accuracy a posteriori of an observed camera co-ordinate of these point categories was $\pm$6 μm and $\pm$8 μm, respectively.

Before we start with the evaluation of our system, let us consider which accuracy can be expected for the reconstruction of buildings in conventional photogrammetry, given the configuration of our test project. In a classical stereo reconstruction using two images only for the determination of each point, a standard deviation of σ_{XY_t} = $\pm$2.5 cm for the planimetric co-ordinates and a standard deviation of σ_{Z_t} = $\pm$4.0 cm for the height of a targeted point can be expected. Taking into account an uncertainty of definition of building vertices of $\pm$7-12 cm for the planimetric co-ordinates and of $\pm$8-15 cm for the height, the standard deviations for the vertex co-ordinates can be expected to be σ_{XY} = $\pm$7.5 - 12.5 cm and σ_Z - $\pm$9.0 - 15.5 cm using the empirical formulae

given by [Kraus, 1993]. Using more than two images will increase the accuracy of the results, whereas due to monoscopic measurement, accuracy will be deteriorated in comparison to stereoscopic measurement.

In this chapter, the system for semi-automatic building extraction will be tested in two ways. In section 7.1, we want to evaluate the performance of automatic fine measurement with respect to both accuracy and reliability of results, the requirements for the approximate values in order to achieve convergence and with respect to the influence of the control parameters on all these items. In section 7.2, we want to demonstrate the applicability of the overall process by presenting the relevant figures observed while creating a 3D model of a part of the village in our test project.

7.1 Evaluation of the performance of automatic fine measurement

In order to evaluate the performance of the automatic fine measurement tool, we have selected four buildings consisting of five objects which can be clearly related to primitives and will thus be referred to as “primitive 1” to “primitive 5” in this section (figure 7.2). The buildings were chosen according to the following criteria:

- They represent examples for the two most common roof shapes in our test area, i.e., saddle back roofs and saddle back roofs with cut-off gables.
- Each of both roof shapes is contained once in direction from north to south and once from east to west so that the test sample resembles different lighting conditions.
- With respect to primitive 2 there is the problem that two vertices are occluded by trees, and the drive close to the eastern wall of the building could bring about problems for the matching process as its edges could be mistaken to be building edges, even more so because the shadow border on the western side of the building is symmetrical to it. We can see whether the algorithm can handle such cases.
- One of the buildings consists of two primitives of different types which intersect each other so that for each primitive, one vertex is inside the other one and, thus, not visible. This building was selected to show that our algorithm can also handle such cases.

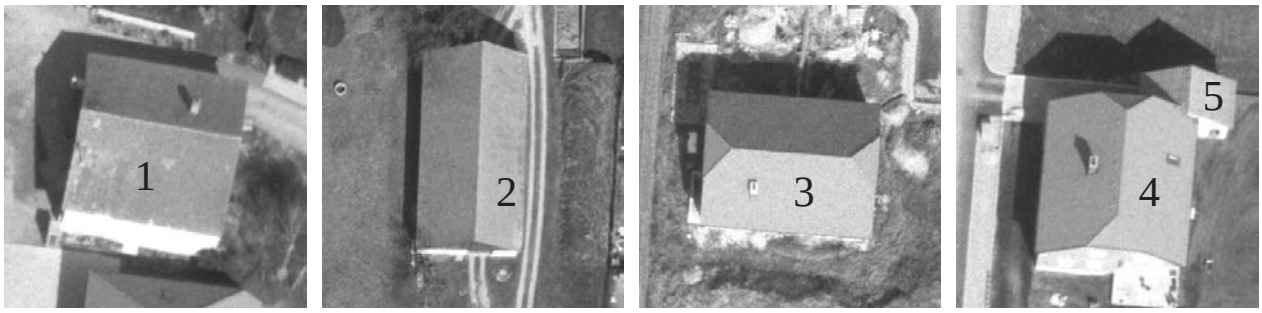


Figure 7.2: The four buildings used for testing. Note that the rightmost building consists of two primitives.

The parameterization of the building primitive resembling a saddle back roof was already described in section 6.2.1 (cf. also figure 6.7). The parameterization of primitives 3 and 4 is similar, but there are two additional faces of triangular shape at the gables. These faces are not considered to be symmetrical. They are described by:

1. Front gable: $w_0 = c^f_{00} + c^f_{01} \cdot v_R; m_u = m_v = m_w = 1$
2. Back gable: $w_0 = c^b_{00} + c^b_{01} \cdot v_R; m_u = m_v = m_w = 1$

In section 7.1.1, we will evaluate the performance of the automated tool for all five primitives using the default values for the parameters of the matching process. In section 7.1.2 we will investigate the influence of the most important parameters on the behaviour of the algorithm, and section 7.1.3 is dedicated to the influence of the quality of the approximate values on the results. In both sections, we will restrict ourselves to primitives 1 to 4. Finally, in section 7.1.4 we will discuss problem areas of automatic fine measurement.

7.1.1 Automatic fine measurement using default parameters

The default parameters for the matching process have already been listed in section 6.4.3. These defaults were chosen according to several criteria:

1. At level l_{start} = 3, the lengths of the building edges of 5-10 m correspond to 10-20 pixels, which is still long enough to make the building edges detectable in the images.
2. ε_u = 5 pixels at pyramid level 3 correspond to about 25% of the building dimensions, an accuracy which can be achieved easily by interactive measurement without being so coarse to prevent automatic fine measurement from success.
3. If matching succeeds at the upper pyramid level, the radius of convergence in the successive levels can be chosen to be small (ε = 2 pixels) in order to reduce the number of iterations required for robust estimation.
4. The building model is supposed to fit with an accuracy of $\sigma_s = \pm 2$ cm. This is in coherence with what we expect to be the accuracy of construction. It corresponds to about 1/3 of a pixel in image space and thus is supposed to be strict enough to make outliers determinable in the matching process.
5. $W_{min} = 2.5 \cdot \ median(W)$ was found to give good results in previous tests of the feature extraction modules.

The results of automatic fine measurement of the primitives resembling saddle back roofs (1, 2, and 5) and a comparison to parameters determined interactively is contained in table 7.1. With respect to the primitives resembling saddle back roofs with cut-off angles, the results are contained in table 7.2. Table 7.3 contains the significant figures of the matching procedure for all primitives.

The approximate values for automatic fine measurement were determined as follows:

- Saddle back roofs: Three building vertices belonging to the eaves were measured in one image coarsely, one of them also in a second image in order to determine Z_0. The procedure corresponds to the one depicted in figure 6.12. Note that the initial values for X_0, Y_0, a_{00}^f and b_{00}^r (i.e., the building position and the planimetric building extents) are thus determined relatively well whereas both roof height and tilt are quite off their actual values.
- Saddle back roofs with cut-off gables: A similar procedure as for saddle back roofs was applied. A fourth point (one of the tops of the triangular faces) was additionally measured in one image to provide better approximations for the small triangles. In fact, this is not really necessary. The quality of the approximate values is similar to the one for the saddle back roofs.

The manual measurements were carried out using the interactive tool described in section 6.3. All roof vertices were measured interactively in all images they were visible in. The parameters of the five primitives were determined the way described in section 6.3. Thus, as the surface observations were used in the measurement process, the results of interactive measurement cannot really be considered to be independent of those derived automatically. Still, these results are a product of human scene interpretation, and it does make sense to use them for a comparison. In the future, a comparison between the matching results and the results of a completely independent measuring process has to be carried out.

Interpreting tables 7.1, 7.2 and 7.3, we can make the following observations:

P	S/R [μm] r.m.s.	X_0 [m] [cm]	Y_0 [m] [cm]	κ [gon] [^{c}]	a^f_{00} [m] [cm]	b^r_{00} [m] [cm]	c^r_{00} [m] [cm]	c^r_{10} [%] [%]
1	I	-247.31	1180.98	289.41	6.35	-6.51	-8.00	-74.9
	120	-247.39	1180.71	289.66	6.78	-6.86	-6.79	-48.7
		$\pm$2.6	$\pm$2.6	$\pm$28.6	$\pm$2.6	$\pm$2.6	$\pm$11.0	$\pm$1.9
	60	-247.32	1180.60	289.29	6.62	-6.77	-6.54	-48.5
		$\pm$0.9	$\pm$0.9	$\pm$9.3	$\pm$1.0	$\pm$1.0	$\pm$2.4	$\pm$0.5
	30	-247.31	1180.59	289.04	6.61	-6.77	-6.47	-48.4
		$\pm$0.5	$\pm$0.5	$\pm$6.1	$\pm$0.6	$\pm$00.6	$\pm$1.6	$\pm$0.3
	15	-247.32	1180.60	289.15	6.62	-6.76	-6.46	-48.8
		$\pm$0.3	$\pm$0.3	$\pm$3.5	$\pm$0.3	$\pm$0.3	$\pm$0.9	$\pm$0.2
	M	-247.35	1180.56	289.20	6.59	-6.78	-6.52	-50.7
		$\pm$1.0	$\pm$1.0	$\pm$7.6	$\pm$1.1	$\pm$1.1	$\pm$2.3	$\pm$0.4
	$\|\Delta\|$	**0.03**	**0.04**	**0.05**	**0.03**	**0.02**	**0.06**	**1.9**
2	I	-590.92	975.68	197.84	6.62	-12.71	-8.00	-75.5
	120	-590.96	975.19	197.55	6.80	-13.00	-9.23	-105.8
		$\pm$2.6	$\pm$5.2	$\pm$142.0	$\pm$4.6	$\pm$5.2	$\pm$7.3	$\pm$2.1
	60	-590.92	975.44	197.55	6.63	-12.81	-9.26	-97.3
		$\pm$0.9	$\pm$1.6	$\pm$4.8	$\pm$1.2	$\pm$1.6	$\pm$2.7	$\pm$0.7
	30	-590.89	975.57	197.58	6.63	-12.71	-9.27	-94.7
		$\pm$0.4	$\pm$0.7	$\pm$2.7	$\pm$0.5	$\pm$0.7	$\pm$1.6	$\pm$0.3
	15	-590.88	975.62	197.54	6.64	-12.69	-9.26	-93.8
		$\pm$0.2	$\pm$0.4	$\pm$1.4	$\pm$0.3	$\pm$0.4	$\pm$0.5	$\pm$0.1
	M	-590.85	975.65	197.45	6.59	-12.69	-9.25	-94.5
		$\pm$1.0	$\pm$1.0	$\pm$4.5	$\pm$1.4	$\pm$1.0	$\pm$2.3	$\pm$0.5
	$\|\Delta\|$	**0.03**	**0.03**	**0.09**	**0.05**	**0.00**	**0.01**	**0.7**
5	I	-612.09	1058.00	-5.30	-4.00	2.04	-8.00	75.3
	120	-612.00	1057.93	-2.22	-3.95	1.98	-7.41	69.3
		$\pm$11.3	$\pm$8.2	$\pm$116.9	$\pm$11.4	$\pm$7.4	$\pm$23.4	$\pm$7.0
	60	-612.49	1057.87	-4.84	-4.17	2.04	-7.63	72.3
		$\pm$3.5	$\pm$2.2	$\pm$44.7	$\pm$5.1	$\pm$2.1	$\pm$5.9	$\pm$2.0
	30	-612.47	1057.89	-5.00	-4.13	2.05	-7.68	72.1
		$\pm$1.0	$\pm$0.8	$\pm$17.6	$\pm$1.5	$\pm$0.8	$\pm$2.1	$\pm$0.8
	15	-612.47	1057.89	-5.09	-4.13	2.06	-7.65	71.3
		$\pm$0.6	$\pm$0.6	$\pm$12.0	$\pm$0.8	$\pm$0.5	$\pm$1.5	$\pm$0.5
	M	-612.42	1057.89	-5.20	-4.03	2.05	-7.66	72.5
		$\pm$2.0	$\pm$2.0	$\pm$34.6	$\pm$2.6	$\pm$2.0	$\pm$3.6	$\pm$1.2
	$\|\Delta\|$	**0.05**	**0.00**	**0.11**	**0.10**	**0.01**	**0.01**	**1.2**

Table 7.1: Results of the matching process for the primitives 1, 2 and 5 (those resembling saddle back roofs) in figure 7.2. The planimetric co-ordinates of $\mathbf{P}_0$ are reduced by $(-34000.00, 5389000.00)$. The floor heights were kept fixed at $Z_0^1 = 270.257$, $Z_0^2 = 272.480$ and $Z_0^5 = 270.260$. P: Number of the primitive in figure 7.2. S/R: State/Resolution. r.m.s.: units of the r.m.s. errors. I: Initial values for automatic fine reconstruction. M: Manual determination of the building parameters. Beneath each parameter, there is its r.m.s. error. $|\Delta|$: The absolute value of the difference between the results of manual measurement and the results of automatic fine measurement in the lowest pyramid level.

Theoretical accuracy: The theoretical accuracy of the building parameters is estimated to be in a range between ± 0.2 and ± 0.5 cm in planimetric position and extent and between ± 0.5 and ± 1.5 cm in height. The errors of the rotational angle κ correspond to similar perpendicular distances (an angular error of $\pm 10^c$ corresponds to a perpendicular error of ± 1.6 cm at a distance of 10 m). The errors of the roof tilts of about $\pm 0.2\%$ to $\pm 1.5\%$ correspond to similar errors as those of the building heights ζ_{00}. Note that the r.m.s. errors of the heights of the eaves are influenced both by the r.m.s. errors of the building heights and the roof tilts: the influences of both parameters are accumulated according to the law of error propagation. We can assume the results of automatic fine measurement to be more accurate than those achieved in the conventional photogrammetric process because the roof edges are more representative structures than the roof vertices, and the influence of errors of definition of these edges is reduced by the approximating procedure applied to the image edges. Still, all the theoretical accuracy figures appear to be far too optimistic (the buildings for sure were not even constructed with an accuracy of a few millimeters). This is caused by the enormous redundancy numbers of adjustment: There are about 600-900 redundant observations for primitives 1 to 4 and about 200 for primitive 5, the parameters of which, consequently,

P	S/R [μm] r.m.s.	X_0 [m] [cm]	Y_0 [m] [cm]	κ [gon] [^{c}]	a_{00}^f [m] [cm]	b_{00}^r [m] [cm]	c_{00}^r [m] [cm]	c_{10}^r [%] [%]	c_{00}^f [m] [cm]	c_{01}^f [%] [%]	c_{00}^b [m] [cm]	c_{01}^b [%] [%]
3	I	-55.16	60.66	-5.41	-7.94	6.30	-6.00	-82.1	-9.00	75.0	-9.00	-75.0
	120	-54.92	60.39	-4.80	-7.84	6.18	-6.51	-79.1	-9.69	66.0	-11.05	-87.1
		± 3.0	± 2.4	± 18.3	± 2.8	± 2.9	± 11	± 2.1	± 38	± 5.2	± 48	± 6.5
	60	-54.83	60.41	-4.49	-7.93	6.24	-6.46	-79.4	-10.17	71.8	-11.32	-88.0
		± 1.1	± 0.9	± 7.0	± 1.1	± 1.1	± 2.4	± 0.5	± 11	± 1.6	± 16	± 2.3
	30	-54.83	60.44	-4.45	-7.93	6.19	-6.47	-79.2	-10.62	77.6	-11.09	-84.4
		± 0.5	± 0.4	± 3.2	± 0.5	± 0.5	± 1.2	± 0.3	± 6.0	± 0.8	± 6.4	± 0.9
	15	-54.82	60.43	-4.44	-7.92	6.18	-6.45	-79.0	-10.63	77.8	-11.08	-84.4
		± 0.3	± 0.3	± 2.3	± 0.3	± 0.3	± 0.8	± 0.2	± 4.9	± 0.7	± 5.0	± 0.7
	M	-54.84	60.44	-4.36	-7.91	6.18	-6.45	-79.7	-10.82	80.4	-11.08	-84.1
		± 1.0	± 0.9	± 5.8	± 0.8	± 1.2	± 1.8	± 0.4	± 9.4	± 1.3	± 9.9	± 1.4
	$\lvert\Delta\rvert$	**0.02**	**0.01**	**0.08**	**0.01**	**0.00**	**0.00**	**0.7**	**0.19**	**2.6**	**0.00**	**0.3**
4	I	-18.58	149.86	95.77	6.25	-5.99	-6.00	75.6	-9.00	-75.0	-9.00	75.0
	120	-18.64	149.91	93.60	6.60	-6.70	-6.00	79.0	-9.13	-70.9	-10.70	96.3
		± 6.3	± 4.4	± 43	± 4.3	± 19	± 14	± 3.7	± 58	± 9.4	± 68	± 11
	60	-18.59	150.05	94.66	6.69	-6.47	-5.98	76.9	-9.27	-76.8	-9.91	83.3
		± 1.0	± 1.1	± 9.5	± 1.1	± 1.3	± 3.1	± 0.7	± 19	± 3.3	± 19	± 3.0
	30	-18.58	150.08	94.69	6.69	-6.48	-6.00	75.9	-9.29	-77.1	-9.96	83.6
		± 0.5	± 0.6	± 4.7	± 0.5	± 0.6	± 1.6	± 0.4	± 7.0	± 1.2	± 8.2	± 1.3
	15	-18.58	150.10	94.80	6.69	-6.46	-6.01	76.3	-9.36	-78.1	-9.99	84.5
		± 0.3	± 0.4	± 3.6	± 0.4	± 0.4	± 1.1	± 0.3	± 5.1	± 0.9	± 6.6	± 1.1
	M	-18.57	150.15	94.71	6.64	-6.45	-6.00	76.1	-9.36	-77.6	-9.85	82.3
		± 0.9	± 0.9	± 7.5	± 0.9	± 1.5	± 1.8	± 0.5	± 8.5	± 1.4	± 0.1	± 1.8
	$\lvert\Delta\rvert$	**0.01**	**0.05**	**0.09**	**0.05**	**0.01**	**0.01**	**0.2**	**0.00**	**0.5**	**0.14**	**2.2**

Table 7.2: Results of the matching process for the primitives 3 and 4 (those resembling saddle back roofs with cut-off gables) in figure 7.2. The planimetric co-ordinates of $\mathbf{P}_0$ are reduced by $(-34600.00, 5389900.00)$. The floor heights were kept fixed at $Z_0^3 = 276.940$ and $Z_0^4 = 275.320$. P: Number of the primitive in figure 7.2. S/R: State/Resolution. r.m.s.: units of the r.m.s. errors. I: Initial values for automatic fine reconstruction. M: Manual determination of the building parameters. Beneath each parameter, there is its r.m.s. error. $|\Delta|$: The absolute value of the difference between the results of manual measurement and the results of automatic fine measurement in the lowest pyramid level.

have the highest r.m.s. errors (primitive 5 is relatively small, so that the number of observations is much smaller than for the other ones). Note that with respect to the parameters of the small triangular faces of primitives 3 and 4, the r.m.s. errors are much greater than those of the main roof faces: as these faces are rather small, there is only a small number of observations available for them. In addition, as the observation co-ordinate system is centered at the primitive center, the parameters of the small triangles are somewhat extrapolated in adjustment.

Convergence behaviour: Due to the effects of lowpass filtering in pyramid generation, only the most salient edges are detected in the images at pyramid level 3 (120 μm). This results in the fact that some edges, especially smaller edges or edges having low contrast, are not detected at all, which results in a smaller percentage of support c_{all} in the upper pyramid levels. As the number of parameters to be estimated is rather small, there is enough information available in all six images so that the procedure converges. The number of hypotheses grows considerably with resolution: With increasing resolution, more and more noise becomes available in the images which causes, for instance, the image edges to be broken. In addition, small object features such as windows or small structures on the roof might be extracted as well, and if they are within the search area for a roof edge, they are considered to be hypotheses, too.

If we have a look at the differences of parameters between consecutive levels, we see that in most cases, the changes are in the range of 1 to 2 cm between the resolutions 30 μm and 15 μm, the main exception being Y_0 of primitive 2. As about half the computational efforts are required for the lowest pyramid level, the question arises whether it makes sense to use a scanning resolution of 15 μm in this example. However, the differences $|\Delta|$ are small, too. Using a resolution of 15 μm reduces these differences by 30

P	Res. [μm]	n_h	n_o	n_x	n_{elim}	n_{it}	$\hat{\sigma_o}$	c_{all} [%]
1	120	82	361	286	14	10	1.1	88
	60	172	721	556	34	13	1.1	96
	30	248	1025	784	36	13	1.4	98
	15	634	2569	1942	57	13	1.6	98
2	120	104	449	352	18	11	1.3	85
	60	250	1033	790	82	20	1.3	93
	30	392	1601	1216	75	25	1.5	98
	15	910	3673	2770	129	29	1.4	89
3	120	150	653	506	11	10	1.1	79
	60	284	1181	908	29	12	1.0	94
	30	354	1461	1118	38	18	1.1	98
	15	694	2821	2138	58	19	1.3	96
4	120	134	581	458	12	10	1.1	70
	60	268	1119	860	29	13	1.1	92
	30	376	1549	1184	49	22	1.3	96
	15	632	2573	1952	96	26	1.7	95
5	120	66	301	238	13	9	1.4	73
	60	110	473	370	28	16	1.2	67
	30	136	577	448	20	13	1.3	80
	15	194	809	622	13	7	1.6	77

Table 7.3: Significant figures of the matching process for all primitives in figure 7.2. P: Number of the primitive in figure 7.2. Res.: Resolution of the current pyramid level in [μm]. n_h: number of hypothesized image edge vertices. n_o: number of observations. n_x: number of unknowns. n_{elim}: number of eliminated hypotheses. n_{it}: number of iterations in adjustment. $\hat{\sigma_o}$: estimated r.m.s. error of the weight unit, in units of σ_o a priori. c_{all}: overall support of the primitive.

to 50% for most parameters. More extensive tests are required to answer this question in a definite way.

Self diagnosis: The matching process converged in a satisfactory way for primitives 1 to 4. With respect to primitive 5, there is an error in parameter a^f_{00} of 10 cm which corresponds to a difference of the building widths of $2 \cdot 10 = 20$ cm. This is caused by the fact that the one of the eaves is in the shadow of primitive 4 so that it is not correctly matched. Looking at table 7.3 we see that the overall support c_{all} is considerably smaller than for the other primitives, but this is what has to be expected because one edge of the primitive is completely inside primitive 4. $\hat{\sigma_o}$ is rather high, but so it is with primitives 4 and 1 even though we consider these primitives to be correctly matched. We could take the small overall support and the relatively great value of $\hat{\sigma_o}$ as a warning and as a hint to look at the results for this primitive more closely, but in the case of our five primitives, we cannot decide whether the results are sufficiently correct or not. Perhaps, a variance component analysis taking the correspondence hypotheses of each roof edge in each image as the observation groups would give additional hints on badly determined edges. However, in the current version of *ORIENT* this is not possible due to restrictions in defining groups of observations.

Comparison to manual measurement: Comparing the lines denoted by $|\Delta|$ of tables 7.1 and 7.2, we see that the results of manual measurement differ slightly from the matching results in pyramid level 0. Apart from primitive 5, these differences are below or equal to 5 cm in planimetry which corresponds to less than one pixel in image space. These results are very satisfactory. In order to give more realistic measures for the accuracy of the results than the theoretical r.m.s. errors of the unknowns, the differences Δ can be evaluated because their true value is known to be zero. The r.m.s. error of the differences σ_Δ can be computed from

$$\sigma_\Delta = \pm\sqrt{\frac{\sum \Delta^2}{n}} \tag{7.1}$$

where n is the number of differences used for computing the sum of squared differences $\sum \Delta^2$. σ_Δ gives a realistic measure for the accuracy of the matching results with the limitations that in our case the test sample is small and that the manual and the automatic measurement procedures are not completely independent. However, the r.m.s. errors of the building parameters cannot be derived easily from σ_Δ because σ_Δ contains the r.m.s. errors of both measurement series. If both series were uncorrelated and if they were equally accurate, the r.m.s. error σ_p of a parameter p could be estimated from $\sigma_{\Delta p}$ by $\sigma_p = \sigma_{\Delta p}/\sqrt{2}$. We have determined σ_Δ for all groups of building parameters:

- *Position* (X_0, Y_0)*:* The positions of the primitives differ by 5 cm at most. $\sigma_{\Delta X_0, \Delta Y_0}$ = $\pm$3.1 cm, which corresponds to about half a pixel in image space.
- *Rotation* κ*:* $\sigma_{\Delta\kappa}$ = $\pm 8.5^c$. This corresponds to a perpendicular distance of 1.3 cm at a distance of 10 m.
- *Planimetric dimensions* (a^f_{00}, b^r_{00})*:* $\sigma_{\Delta a^f_{00}, \Delta b^r_{00}}$ = $\pm$4.1 cm including the parameters of primitive 5. If we consider these parameters to be outliers, we obtain $\sigma_{\Delta a^f_{00}, \Delta b^r_{00}}$ = $\pm$2.9 cm, which corresponds to $\sigma_{\Delta w}$ = $\pm$5.8 cm for the planimetric building extents w.
- *Building height* c^r_{00}*:* $\sigma_{\Delta c^r_{00}}$ = $\pm$2.8 cm.
- *Roof tilt* c^r_{10}*:* $\sigma_{\Delta c^r_{10}}$ = $\pm$1.1%. The r.m.s. error of the roof tilt propagates to the r.m.s. error of the heights of the eaves. An r.m.s. error of $\pm$1% of c^r_{10} corresponds to an r.m.s. error of $\pm$1 cm of the height difference between the eaves and the central ridge of a building which is 20 m wide.
- *Parameters of the small triangular faces* (c^f_{00}, c^f_{01}, c^b_{00}, c^b_{01})*:* The errors in these parameters are considerably greater than those in the other ones (up to 19 cm). This is partly due to the small size of the small triangular faces, but especially it is caused by effects of extrapolation because c^f_{00} and c^b_{00} are distances measured in the centre of the primitives, and the top vertices of the triangular faces are relatively close to the front and back faces, respectively. In the case of primitive 3, the difference of c^f_{00} of 19 cm corresponds to a difference of 6 cm in the height of the top vertex of the corresponding triangle.

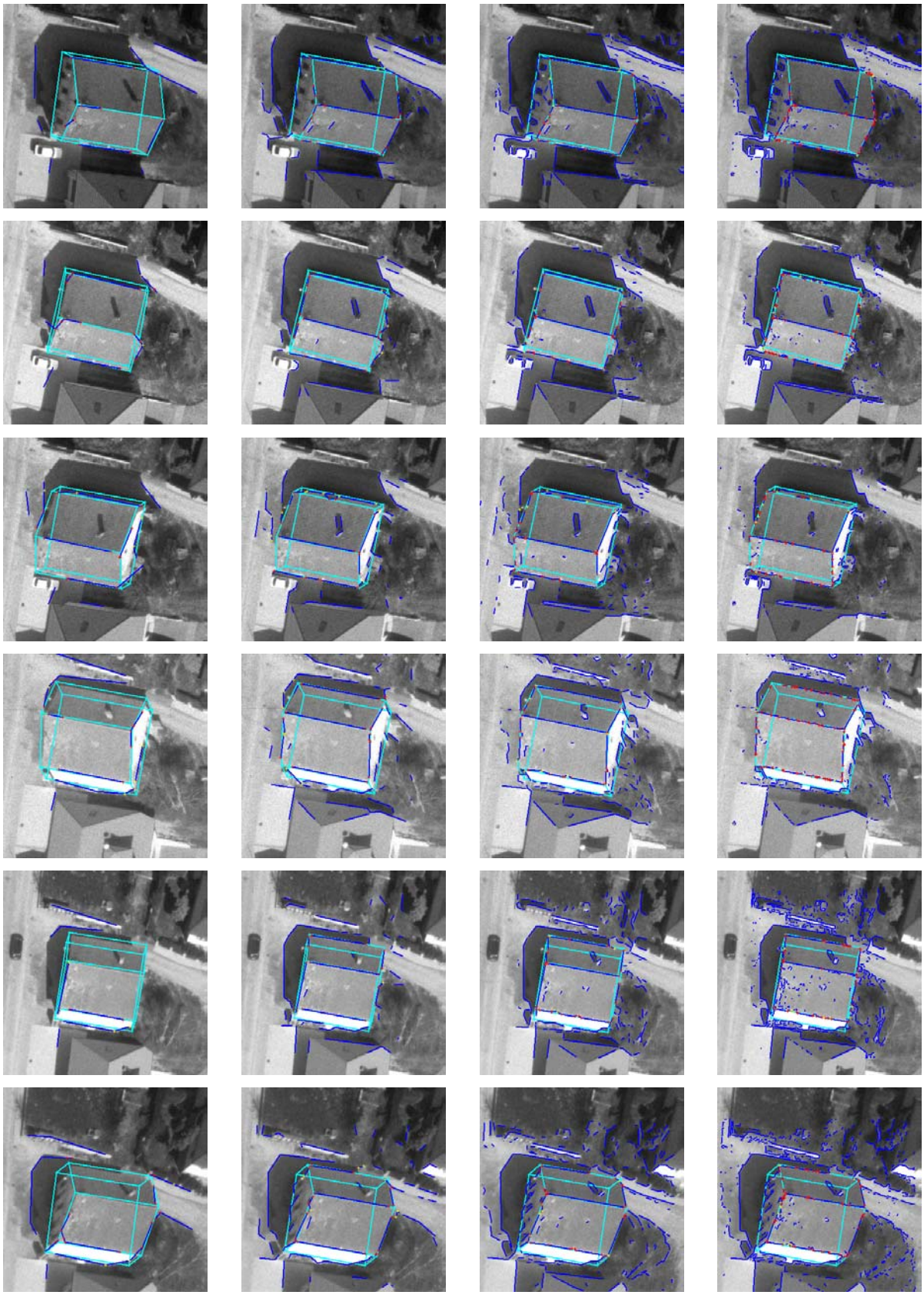


Figure 7.3: Automatic fine measurement using six images in two strips. Each column represents the matching results in one pyramid level starting at level 3 (120 μm, left column) and terminating at level 0 (15 μm, right column). Blue lines: extracted image edges; red crosses: accepted image edge vertices; yellow crosses: eliminated image line vertices; cyan: the adjusted primitive. Only roof edges were used for matching.

Summing up the results of our empirical test we can say that the parameters related to the planimetric positions and extents of the buildings can be derived with an accuracy of ± 2 cm to ± 5 cm. The parameters related to height can be determined with an accuracy of ± 3 cm to ± 10 cm. These figures apply to buildings with well-defined roof edges, the upper limit being relevant for the heights of points on small roof structures.

Figure 7.3 illustrates the matching process for primitive 1 in figure 7.2.

7.1.2 Influence of the control parameters

In this section we want to investigate the effects of the most important control parameters of the matching process on the results and the convergence behaviour. For each of these control parameters, a test series was performed using identical initial values. In each test series, the control parameters not currently being varied were kept at their default values ($W_{min} = 2.5 \cdot median(W)$, $l_{start} = 3$, $\sigma_s = \pm 2$ cm, $\varepsilon = 2$ pixels, $\varepsilon_u = 5$ pixels).

Influence of the multiplication factor j for the threshold for feature extraction: Table 7.4 shows the differences between parameters derived automatically and the results of manual measurement in dependence of the multiplication factor j for the threshold for feature extraction. All figures refer to the best resolution (15 μm). The number n_{it} of iterations for robust estimation and the number n_h of hypotheses are presented in table 7.5. As the threshold W_{min} for texture classification is chosen to be proportional to the median of texture strength W, the factor of proportionality j in equation 5.9 influences the number of features that is extracted. With respect to automatic fine measurement, the effects are as follows:

	ΔX_0 [m]					ΔY_0 [m]				
j	1.0	2.5	3.5	5.0		1.0	2.5	3.5	5.0	j
P1	-0.03	-0.03	-0.03	-0.03		-0.04	-0.04	-0.03	-0.03	P1
P2	0.04	0.03	0.04	0.06		0.05	0.02	0.16	0.62	P2
P3	-0.02	-0.02	-0.02	-0.01		-0.04	0.01	0.00	0.02	P3
P4	0.01	0.01	0.00	-0.01		0.05	0.05	0.05	0.06	P4

	Δa_{00}^f[m]					Δb_{00}^r[m]				
j	1.0	2.5	3.5	5.0		1.0	2.5	3.5	5.0	j
P1	-0.04	-0.04	-0.04	-0.04		-0.02	-0.02	-0.02	-0.02	P1
P2	-0.04	-0.04	-0.84	-0.80		-0.01	0.01	0.14	0.61	P2
P3	0.03	0.01	0.01	0.01		-0.78	0.00	0.00	-0.10	P3
P4	-0.05	-0.05	-0.04	-0.05		0.00	0.00	0.00	0.01	P4

	Δc_{00}^r[m]					Δc_{10}^r[%]				
j	1.0	2.5	3.5	5.0		1.0	2.5	3.5	5.0	j
P1	-0.07	-0.06	-0.06	-0.06		-2.0	-1.9	-2.0	-1.8	P1
P2	0.02	0.01	0.01	0.01		0.0	-0.7	-10.4	-11.1	P2
P3	0.12	0.00	0.00	0.00		9.1	-0.7	-0.7	0.9	P3
P4	0.00	0.00	0.00	-0.03		0.2	-0.2	-0.5	0.1	P4

Table 7.4: Influence of the multiplication factor j for the threshold for feature extraction $W_{min} = j \cdot median(W)$ on some of the parameters for primitives P1-P4 in figure 7.2. The values represent parameter differences between automatic and manual measurement.

	n_{it}					n_h				
j	1.0	2.5	3.5	5.0		1.0	2.5	3.5	5.0	j
P1	15	13	11	6		686	634	580	438	P1
P2	27	29	27	26		1100	910	618	588	P2
P3	30	19	21	16		674	694	624	544	P3
P4	29	26	25	32		696	632	578	526	P4

Table 7.5: Influence of the multiplication factor j for the threshold for feature extraction on two of the significant figures of adjustment for primitives P1-P4 in figure 7.2. n_{it}: number of iterations. n_h: number of hypotheses. All figures refer to a resolution of 15 μm. $\hat{\sigma_o}$ was between $\pm$1.4 and $\pm$1.7 in all cases.

- Small values for j result in a greater number of extracted features and, thus, in a greater number of hypotheses. As a consequence, the number of iterations required is increased (table 7.5). In some cases, this is not the case because the additional features have obviously not been extracted in the search regions or because the matching process has already diverged in an upper pyramid level so that the figures of pyramid level 0 are not representative.

- Large values for j result in a smaller number of extracted features and thus, reduce both the number of hypotheses and the number of iterations required, with the exceptions already mentioned above.

- The greater number of features extracted for $j = 1$ does not result in more accurate results. There are obviously too many false hypotheses which just increase the computational efforts which are required for robust estimation. In case of primitive 3, the matching process even fails ($\Delta Y_{00} = -78$ cm).

- On the other hand, if j is chosen too great, matching might fail because the number of hypotheses is too small and because important features are missed (e.g. primitive 3: $\Delta Y_{00} = 61$ cm). This is especially critical in the upper pyramid levels because once matching failed, the hierarchical procedure cannot recover in the low resolution images.

The default value $j = 2.5$ appears to be a good choice. Only in image areas characterized by low contrast it might be convenient to choose a smaller value, e.g. $j = 1.5$.

Influence of the a priori r.m.s. error σ_s of the surface observations: Table 7.6 shows the differences between parameters derived automatically and the results of manual measurement in dependence on the a priori r.m.s. error σ_s of the surface observations. All figures refer to the best resolution (15 μm). The number n_{it} of iterations for robust estimation and the r.m.s. error of the weight unit $\hat{\sigma_o}$ are presented in table 7.7. The a priori r.m.s. error σ_s describes the "rigidity" of the building models in the matching process. It reflects the accuracy of definition of the building edges in object space. With respect to automatic fine measurement, the effects are as follows:

- If σ_s is chosen to be small, the conditions imposed on the matching process by the building model have to be fulfilled rather strictly. This means that a greater number of hypotheses will be eliminated from adjustment, which is reflected by a considerably greater number of iterations required for robust estimation (cf. table 7.7, especially n_{it} for σ_s = $\pm$1 cm and σ_s = $\pm$2 cm).

- If σ_s is chosen to be large, the number of hypotheses eliminated in robust estimation is decreased, and so is the number of iterations. However, this means that in extreme cases ($|\sigma_s| \geq 10$ cm; cf. table 7.7) no false matches are detected at all, all hypotheses fit to the model, and the results are wrong.

- The best results are achieved for $\sigma_s = \pm 2$ cm (table 7.6). The results for σ_s = $\pm$1 cm and σ_s = $\pm$5 cm are still acceptable. This seems to be realistic if we consider the error tolerances in building construction.

- Note that the fact that no more hypotheses are eliminated is reflected in $\hat{\sigma_o}$: For $|\sigma_s| > 10$ cm, $\hat{\sigma_o}$ is considerably smaller than 1: There are no contradictions in adjustment because the influence of the surface observations is so small.

	ΔX_0 [m]							ΔY_0 [m]						
σ_s	1	2	5	10	50	100		1	2	5	10	50	100	σ_s
P1	-0.03	-0.03	-0.03	-0.03	-0.07	-		-0.03	-0.04	-0.05	-0.05	-0.50	-	P1
P2	0.03	0.03	0.03	0.05	-0.04	0.02		0.01	0.02	0.07	0.25	0.41	0.44	P2
P3	-0.02	-0.02	-0.02	-0.01	-0.08	-0.08		0.00	0.00	0.04	0.13	0.05	0.05	P3
P4	0.02	0.01	0.00	0.00	-0.82	-0.86		0.07	0.05	0.04	0.09	0.17	0.19	P4

	Δa_{00}^f[m]							Δb_{00}^r[m]						
σ_s	1	2	5	10	50	100		1	2	5	10	50	100	σ_s
P1	-0.03	-0.04	-0.04	-0.04	-0.53	-		-0.03	-0.02	-0.01	-0.01	0.03	-	P1
P2	-0.05	-0.04	-0.05	-0.19	-0.24	-0.16		0.01	0.01	0.01	0.16	0.32	0.33	P2
P3	-0.01	-0.01	-0.01	-0.01	0.06	0.10		-0.01	0.00	0.06	0.15	0.06	0.05	P3
P4	-0.06	-0.05	-0.03	-0.02	0.11	0.09		-0.04	0.00	0.04	0.06	-2.05	-2.04	P4

	Δc_{00}^r[m]							Δc_{10}^r[%]						
σ_s	1	2	5	10	50	100		1	2	5	10	50	100	σ_s
P1	-0.06	-0.06	-0.05	-0.04	-0.20	-		-2.3	-1.9	-1.4	-1.2	-13.7	-	P1
P2	0.01	0.01	0.01	0.03	-0.28	-0.56		-1.1	-0.7	0.1	5.8	-5.7	-10.2	P2
P3	0.00	0.00	-0.02	-0.27	-0.09	-0.15		0.8	0.7	0.0	6.3	1.6	2.7	P3
P4	0.01	0.00	0.01	-0.09	-1.87	-2.04		0.0	-0.2	-0.8	1.3	54.9	60.7	P4

Table 7.6: Influence of the a priori r.m.s. error σ_s of the surface observations on some of the parameters for primitives P1-P4 in figure 7.2. The values represent parameter differences between automatic and manual measurement. σ_s is given in [cm]. All figures refer to a resolution of 15 μm. For primitive 1, the trial for $\sigma_s = \pm 100$ cm is missing ("-")

	n_{it}							$\hat{\sigma_o}$						
σ_s	1	2	5	10	50	100		1	2	5	10	50	100	σ_s
P1	34	13	1	1	1	-		1.6	1.6	1.3	0.7	0.2	-	P1
P2	52	29	9	3	1	1		1.5	1.4	1.4	1.1	0.4	0.2	P2
P3	30	22	11	1	1	1		1.4	1.3	1.2	1.1	0.2	0.1	P3
P4	47	26	9	1	1	1		1.5	1.7	1.3	1.4	0.3	0.2	P4

Table 7.7: Influence of the a priori r.m.s. error σ_s of the surface observations on two of the significant figures of adjustment for primitives P1-P4 in figure 7.2. n_{it}: number of iterations. σ_s is given in [cm]. $\hat{\sigma_o}$: estimated r.m.s. error of the weight unit a posteriori. σ_o a priori was chosen to be 1. All figures refer to a resolution of 15 μm. For primitive 1, the trial for $\sigma_s = \pm 100$ cm is missing ("-").

Influence of the search space threshold ε_u on the upper pyramid level: The distance threshold ε_u in the upper pyramid level l_{start} has a significant influence on the convergence behaviour of automatic fine measurement: it directly delimits the overall search space. If matching succeeds on the upper pyramid level, the initial values for the subsequent levels will be more or less identical with only small variations. That is why, evaluating the influence of ε_u, we are mainly interested in two questions:

1. *Convergence ability:* For which values of ε_u can convergence be achieved?

2. *Behaviour of matching on pyramid level l_{start}:* How does the number of hypotheses and the number of iterations change in dependence of ε_u?

ε_u	2	3	5	7	10	15
P1	y	y	y	y	y	n
P2	y	y	y	y	y	y
P3	y	y	y	y	y	n
P4	y	y	y	y	y	n

Table 7.8: Convergence of automatic fine measurement in dependence of ε_u for l_{start} = 3 (120 μm). y: convergence could be achieved. n: no convergence.

The first question is answered by table 7.8. From that table we can see that automatic fine measurement converged for $\varepsilon_u \leq 10$ pixels (1.2 mm), for primitives 1, 3, and 4, whereas for primitive 2 it converged even for $\varepsilon_u = 15$ pixels (1.8 mm). The results on pyramid level 0 are equal to those presented in tables 7.1 and 7.2 in section 7.1.1 with respect to the trials in which convergence could be achieved. In this context, "*convergence*" was defined to be achieved if the difference of parameters was smaller than 7 cm in planimetric position and/or extent and smaller than 10 cm in height. With respect to the small triangular faces, somewhat larger thresholds were used because the parameters do not represent the actual error in height, as already discussed in section 7.1.1. Note that the length of the buildings corresponding to primitives 1, 3, and 4 is about 12-15 m in object space, which corresponds to 2.7-3.3 mm in image space. We can see that the procedure converges if the overall search area is in the range of about half the object extension. Thus, as the length of primitive 2 is about 25 m (corresponding to 5.5 mm in the images) the procedure still converges for ε_u = 15 pixels with respect to that building.

Table 7.9 shows the number of iterations n_{it} and the number of hypotheses n_h on pyramid level 3 (120 μm) in dependence of ε_u. Of course, the number of hypotheses and, thus, the number of iterations required for eliminating the false correspondences grows with increasing size of the search area. As $\hat{\sigma_0}$ was between 0.9 and 1.5 in all cases, it cannot be used as a good indicator for convergence.

Influence of the search space threshold ε: The threshold ε delimits search space in all pyramid levels except the upper one. If matching converges in the upper pyramid level l_{start}, the results of that level should be good enough so that ε can be kept small. In our test series for ε we observed that the final results were not influenced by the size of ε which we kept in the range between 2 and 4 pixels of the current pyramid level: all parameters varied by less than $\pm$2 cm (and corresponding figures for the tilts and angles). However, the number of hypotheses and, thus, the number of iterations required to eliminate false matches is increased considerably as ε grows (cf. table 7.10). For ε = 4 pixels, the number of iterations required is almost doubled in comparison to those required for ε = 2 pixels. We consider the default value ε = 2 pixels to be convenient in most cases.

	n_{it}							n_h						
ε_u	2	3	5	7	10	15		2	3	5	7	10	15	ε_u
P1	6	9	10	11	12	17		76	80	82	86	90	96	P1
P2	8	11	14	15	15	18		82	88	90	92	92	96	P2
P3	7	7	10	10	14	24		150	150	150	152	162	164	P3
P4	8	10	10	14	15	15		132	132	134	140	142	146	P4

Table 7.9: The number of iterations n_{it} and the number of hypotheses n_h on pyramid level 3 (120 μm) in dependence of ε_u for primitives P1 - P4. $\hat{\sigma_o}$ was between 0.9 and 1.5 in all cases.

	n_{it}				n_h			
ε	2	3	4		2	3	4	ε
P1	13	18	25		636	670	728	P1
P2	31	40	47		916	976	1034	P2
P3	19	29	35		694	790	832	P3
P4	26	36	45		632	598	682	P4

Table 7.10: The number of iterations n_{it} and the number of hypotheses n_h on pyramid level 0 (15 μm) in dependence of ε for primitives P1 - P4. All the geometrical parameters of the primitives varied by less than $\pm$2 cm (and corresponding figures for the tilts and angles).

Effects of the coarse-to-fine strategy: Finally, we want to see what the benefits of applying a coarse-to-fine strategy for automatic fine measurement are. It has been stated in section 6.4.3 that the overall radius of search space is $\varepsilon_u \cdot 2^{l_{start}}$ pixels on the lowest pyramid level, which evaluates to 40 pixels on the lowest pyramid level for the default values. We want to obtain an answer to the question whether automatic fine measurement converges and gives identical results if ε_u and l_{start} are varied in a way that the size of the overall search space is constant.

ε_u	5	10	20	40
l_{start}	3	2	1	0
Res.	120	60	30	15
P1	y	n	n	n
P2	y	y	y	n
P3	y	n	n	n
P4	y	y	n	n

Table 7.11: Convergence behaviour for several values of ε_u and l_{start} that correspond to an identical size of the overall search space of 40 pixels in the lowest pyramid level (15μm). ε_u is given in [pixels] on level l_{start}. Res.: resolution of the images at pyramid level l_{start} in [μm]. y: convergence could be achieved. n: no convergence.

	n_{it}					n_h				
ε_u	5	10	20	40		5	10	20	40	ε_u
l_{start}	3	2	1	0		3	2	1	0	l_{start}
Res.	120	60	30	15		120	60	30	15	Res.
P1	13	16	18	76		634	464	446	580	P1
P2	29	26	24	182		916	902	880	1788	P2
P3	22	27	28	56		696	522	440	600	P3
P4	26	26	29	69		632	630	476	682	P4

Table 7.12: The number of iterations n_{it} and the number of hypotheses n_h on pyramid level 0 for several values of ε_u and l_{start} that correspond to an identical size of the overall search space of 40 pixels in the lowest pyramid level (15μm). Res.: resolution of the images at pyramid level l_{start} in [μm]. $\hat{\sigma_o}$ was between 1.4 and 1.7 in all cases.

From table 7.11 we can see that automatic fine measurement did not converge in most cases even though the overall search space was identical. The results on pyramid level 0 are equal to those presented in tables 7.1 and 7.2 in section 7.1.1 with respect to the trials in which convergence could be achieved. Again, "*convergence*" was defined to be achieved if the difference of parameters was smaller than 7 cm in planimetric position and/or extent and smaller than 10 cm in height, and with respect to the small triangular faces, somewhat larger thresholds were used because the parameters of these faces do not represent the actual error in height, as already

discussed in section 7.1.1. It appears to be strange that automatic fine measurement fails to converge already for ε = 10 pixels on level l_{start} = 2 for primitives 1 and 3. Obviously, robust estimation is not able to separate the false hypotheses from the correct ones in these cases. In case convergence could be achieved, the number of iterations n_{it} and the number of hypotheses n_h on pyramid level 0 are approximately identical (table 7.12). Otherwise, the number of hypotheses is much smaller: the approximations are too far off the correct values so that a smaller number of possible candidates is within the search area of ε = 2 pixels. The exception from this observation is the trial starting at level 0 because in this trial, ε_u = 40 pixels is used to delimit search space in the images with full resolution. We see that in this case, the number of possible candidates for (false) hypotheses is quite large for some of the primitives.

From the results of our experiment documented in tables 7.11 and 7.12 we conclude that using the coarse-to-fine strategy in the matching process is very important for achieving correct results. These observations seem to be somewhat contradictory to the results of the test series varying ε_u, where we noticed that convergence could be achieved for greater search area thresholds ε_u in pyramid level 3. We think that the possibility to increase search space in the upper pyramid level is due to the effects of lowpass filtering being applied for the generation of the image pyramids: only the most salient features survive in pyramid level 3, and the overall number of possible candidates for (false) matches is rather small because of the smoothing effects of the filtering process.

7.1.3 Influence of the quality of the approximate values

In order to evaluate automatic fine measurement with respect to its sensitivity to the quality of the approximate values, the primitives 1 - 4 from figure 7.2 were reconstructed starting from four different approximate positions. The default values of the control parameters for automatic fine measurement were used in all cases. The differences between the approximate values for some of the primitive parameters and the parameters determined by interactive measurement are presented in table 7.13. The approximate values differ from the final ones by up to 0.4 m in planimetric position and extents (corresponds to 6 pixels in the images on pyramid level 0)

	ΔX_0 [m]					ΔY_0 [m]				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	-0.04	0.06	0.07	-0.09		-0.41	0.14	-0.15	-0.13	P1
P2	0.08	0.10	0.11	0.04		-0.04	-0.09	-0.03	-0.05	P2
P3	0.32	-0.22	0.15	0.03		-0.22	-0.02	0.09	0.02	P3
P4	-0.07	0.01	-0.03	-0.03		0.08	0.29	-0.01	0.14	P4

	Δa^f_{00}[m]					Δb^r_{00}[m]				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	0.24	-0.12	-0.12	-0.06		-0.27	0.00	0.05	0.05	P1
P2	-0.03	-0.08	0.08	0.04		0.03	-0.01	-0.03	0.02	P2
P3	-0.03	-0.12	-0.23	-0.23		0.12	-0.09	-0.09	0.04	P3
P4	-0.06	0.40	0.06	0.11		0.09	-0.46	-0.10	-0.09	P4

	Δc^r_{00}[m]					Δc^r_{10}[%]				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	1.48	1.67	1.58	1.31		-24.25	-24.18	-24.12	-24.21	P1
P2	-1.25	-1.32	-1.00	-0.91		19.00	19.18	18.42	15.47	P2
P3	-0.33	0.11	0.12	-0.39		-2.34	-2.86	-2.65	-1.87	P3
P4	0.56	-0.01	-0.25	0.19		2.64	0.53	1.13	0.97	P4

Table 7.13: Differences between four versions of approximate values of six of the primitive parameters and the values determined by interactive measurement. Approx.: version number of the set of approximate values.

and 1.7 m in height. In order to determine these approximate values, three points belonging to the eaves were measured in one image, and one of them was also measured in a second image. The procedure corresponds to the one described in section 6.3 (cf. figure 6.12).

Table 7.14 shows the number of hypotheses n_h and the number of iterations n_{it} required at the upper pyramid level (120 μm) for the four sets of approximate values. The numbers are more or less identical.

The differences of the matching results for the results of manual measurement for primitives 1 - 4 are shown in Table 7.15. From that table we conclude that if the quality of the approximate values is good enough for convergence, the results of the matching process will be identical independently from the variation of the approximate values.

	n_{it}					n_h				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	10	8	9	9		82	84	88	96	P1
P2	11	10	13	11		90	104	108	104	P2
P3	6	4	6	5		150	142	140	132	P3
P4	13	10	8	11		134	154	150	134	P4

Table 7.14: The number of iterations n_{it} and the number of hypotheses n_h on pyramid level 3 (120 μm) for four different sets of approximate values. Approx.: version number of the set of approximate values.

	ΔX_0 [m]					ΔY_0 [m]				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	-0.03	-0.03	-0.03	-0.03		-0.04	-0.04	-0.04	-0.04	P1
P2	0.03	0.03	0.03	0.03		0.02	0.02	0.01	0.01	P2
P3	-0.02	-0.02	-0.02	-0.02		0.00	0.00	0.00	0.00	P3
P4	0.00	0.01	0.01	0.00		0.06	0.05	0.05	0.05	P4

	Δa_{00}^f[m]					Δb_{00}^r[m]				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	0.04	0.04	0.04	0.04		0.02	0.02	0.02	0.02	P1
P2	0.04	0.04	0.04	0.04		-0.01	-0.01	0.00	0.00	P2
P3	0.01	0.01	0.01	0.01		0.00	0.00	0.01	0.01	P3
P4	-0.05	-0.05	-0.05	-0.04		0.01	0.00	0.01	0.01	P4

	Δc_{00}^r[m]					Δc_{10}^r[%]				
Approx.	1	2	3	4		1	2	3	4	Approx.
P1	-0.06	-0.06	-0.06	-0.06		1.90	1.92	1.90	1.90	P1
P2	0.01	0.01	0.01	0.01		0.73	0.75	0.73	0.75	P2
P3	0.00	0.00	0.00	0.00		-0.71	-0.72	-0.72	-0.72	P3
P4	0.00	0.00	0.00	0.01		-0.13	-0.23	-0.13	-0.67	P4

Table 7.15: Differences between the primitive parameters derived by automatic fine measurement starting from four different sets of approximate values and the values determined by interactive measurement. Approx.: version number of the set of approximate values.

7.1.4 Problem areas

There are several problems which might prevent automatic fine measurement from success:

- *Low contrast in the images:* If the contrast is low in the images, feature extraction might fail to detect image edges corresponding to relevant structures of the building. This is especially critical in the upper pyramid levels because it will cause the hierarchical procedure to fail.
- *Image noise:* Image noise might also influence feature extraction in a bad way. The images of long object edges might be broken into small separated image edges or even disappear because the signal available in the grey levels cannot be separated from the noise. In addition, image noise afflicts the positioning accuracy of the image features.
- *Lighting conditions, shadows:* Shadows can have a bad influence on the results of automatic fine measurement in two ways:
  1. Shadow areas appear very dark in the images, which results in low contrast and, thus, yields bad effects on feature extraction
  2. The shadow borders are often clearly defined, and their shapes are similar to the shapes of the actual building edges which are to be detected. In addition, the actual edges might obtain low contrast. That is why shadow borders are perfect candidates for false matches.
- *Occlusions:* Occlusions of roof edges are likely to happen in densely built-up areas, and they are the more likely the smaller the number of images involved in matching. That is why we try to use more than two images for matching. However, in some cases, occlusions cannot be avoided. In this case, a part of the building is invisible in all images, and the building parameters depending on the invisible parts might be determined falsely.
- *Bad definition of the building edges:* If the building edges are badly defined, the according image edges cannot be determined, which might also cause the matching procedure to fail.
- *Small building features:* Small building features might disappear in the image pyramid levels having a coarse resolution only. In this case, there can only be false hypotheses of correspondence, which will cause a failure of automatic fine measurement. For instance, the small triangular faces of primitives 3 and 4 in figure 7.2 might already be critical in smaller image scales.
- *Bad fit of the model to the actual building:* Of course, matching only can work if the underlying building model actually resembles the building to be reconstructed.

We want to demonstrate the effects of some of these error sources using several examples.

Example 1: Figure 7.4 shows two of the images used for the reconstruction of a hip-roof building. The results of feature extraction on pyramid level 2 are superimposed to the images in white colour. Due to the low contrast, several roof edges are not detected. Note that in the left image, both the upper and the leftmost roof edges are very close to the edges extracted on the floor. As a result, the floor edges are matched with the roof edges (even one of the tilted edges which might have given support to its neighbouring faces and thus might have had a correcting influence in adjustment is missing). There are too many false observations on the upper side of the building in this pyramid level. In the successive levels, the approximate values are too bad, and the building cannot be matched. This is a typical example where low contrast caused by lighting effects and shadows prevent some important features from being detected in the images. In order to reconstruct this building automatically, the threshold for feature matching had to be adapted to $W_{min} = 1 \cdot median(W)$, and l_{start} had to be set to pyramid level 1, a procedure which, however, requires rather good approximate values.

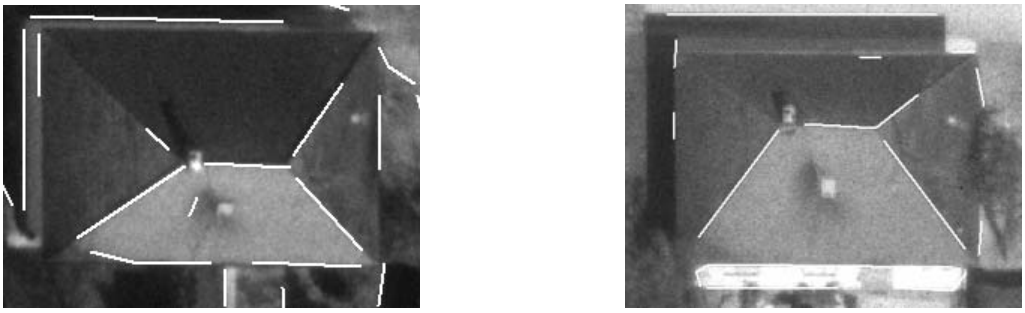


Figure 7.4: Two of the images resembling a hip roof building with the extracted image edges from pyramid level 2 (60 μm) superimposed to them in white colour (example 1). Too many important image edges could not be found due to low contrast and shadows, and the image edge corresponding to the floor (upper part in the left image) was erroneously matched with the roof edge.

Example 2: Another critical situation is illustrated in figure 7.5. The small garage between two larger buildings could not be reconstructed. The figure shows two of six images used for that purpose and the image edges from pyramid level 3. This is a typical example for occlusions causing the matching procedure to fail: the leftmost roof edge of the garage is only visible in one of six images (the right image in figure 7.5), and in this image, the corresponding image edge is not extracted at that pyramid level due to low contrast. As a all the other roof edges of the garage are extracted, the roof height is reconstructed correctly, but the width of the garage is not. The left roof edge will always be positioned a certain distance off the intersection with the wall of the hip-roof.

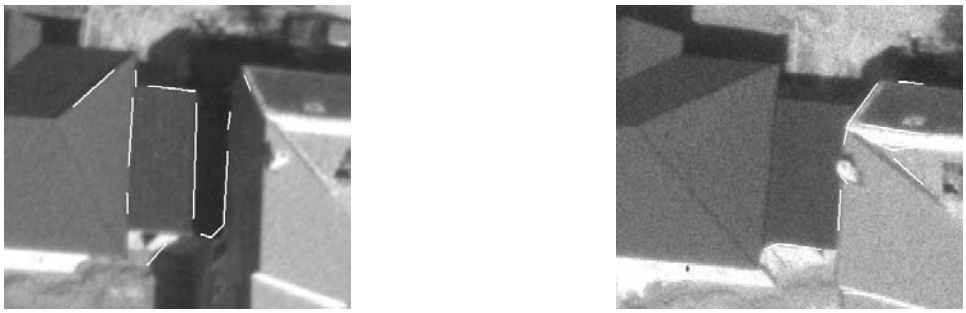


Figure 7.5: Two images for the reconstruction of a small garage between two larger buildings with the extracted image edges from pyramid level 3 (120 μm) superimposed to them in white colour (example 2). The garage could not be reconstructed correctly because one of its roof edges was occluded in all images but the right one in the figure, where it could not be extracted due to the low contrast.

Example 3: Figure 7.6 shows one image used for the reconstruction of a saddle back roof with the features from pyramid level 0 (120 μm) superimposed to it. In this case, the central ridge of the roof is covered by tin which appears as a bright band in the images. As a result, the corresponding image edges are no longer step edges, but rather image strips a few pixels wide. This is not considered in the image model we use for feature extraction. That is why the edges of the bright band are extracted and, consequently, determined to be candidates for correspondence to the central ridge of the building. The number of hypotheses from these false correspondences is so great that they cannot be overridden by other, correct, correspondences. The height of the central ridge and the roof tilt is not determined correctly in this case.

Example 4: Figure 7.7 shows an example for another error source. The primitive resembling a semi-hip roof was used to reconstruct the building depicted in the figure. However, in reality, the leftmost wall of the building is not orthogonal to its neighbours. As the orthogonality conditions are modelled in an implicit

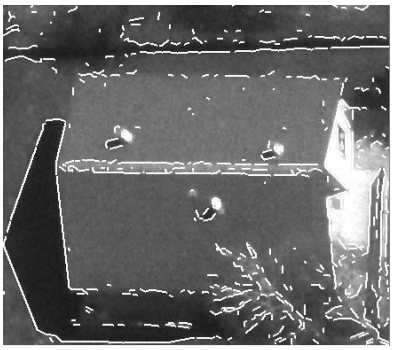


Figure 7.6: One of six images used for the reconstruction of a saddle back roof with the extracted image edges from pyramid level 0 (15 μm) superimposed to it (example 3). As the central ridge appears as a small bright band, its edges are extracted from the images instead of its centre, and, consequently, they are assigned to the central ridge of the primitive.

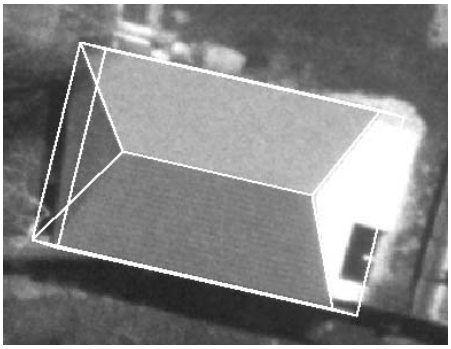


Figure 7.7: One of the images used for the reconstruction of a building that was modelled as a semi-hip roof (example 4). The final position of the automatic process is superimposed to the image in white colour. Matching failed because the model was inappropriate for the building: in reality, the leftmost wall is not orthogonal to its neighbouring faces.

way, the matching procedure is not flexible enough to overcome this situation. This is not a drawback of our method: In fact, the "rigidity" of the model is a prerequisite for separating false matches from correct ones. The problematic situation presented in figure 7.7 can be overcome in two ways:

- Select another primitive fitting to the actual shape of the building in a better way. The problem is considered to be the result of an interpretation error of the human operator.
- As the data base of known building shapes must not become too large, the above strategy cannot be applied in all cases. In this case, the system should be made more flexible in the way that the user could adapt the way the primitives are modelled at run-time by a tool which is easy to handle. For instance, with respect to the building depicted in figure 7.7, the leftmost wall as well as the triangular roof face could be declared to obtain an additional parameter modelling the deviation from the original model.

7.2 Applicability of the overall process

In order to evaluate the applicability of the overall process of semi-automatic building extraction as it is realized in our system, a part of the centre of the village of Stoitzendorf was completely reconstructed using the new technique. Figure 7.8 shows a part of one of the digital images used for reconstruction with the reconstructed buildings super-imposed to it as wire frames. The scene consists of altogether 18 distinct building blocks. If

it was clear from the aerial view that two neighbouring buildings had different postal addresses even though they touched each other, the two buildings were treated individually. If this distinction was not clear, they were supposed to be parts of one compound building. For instance, in the right upper corner of figure 7.8, two buildings in the sense just described were reconstructed: The hip roof with the garage attached to it and the group consisting of five primitives, three of them resembling tilted roofs, one a semi-hip roof and one a saddle back roof with one cut-off gable.

Figure 7.8: A part of one of the digital images used for reconstruction with all the reconstructed buildings super-imposed to it in white colour.

Prim.	saddle back roof	tilted roof	saddle back w. cut-off gable	semi-hip roof	pyramid (4 walls)	hip roof	flat roof	pyramid 6 sides	Total
Number	26	11	3	3	2	1	1	1	48
Autom.	9	2	2	0	0	0	0	0	13
Total number of buildings: 18									

Table 7.16: Statistics about the primitives used for the reconstruction of the buildings in figure 7.8. Prim.: Name of the primitive type. Number: The number of primitives of the respective type that were used in the test. Autom.: Number of primitives that could be matched automatically. Total: Total number of primitives and primitives that could be matched, respectively.

Table 7.16 gives the statistics about the primitives used for the reconstruction of 18 buildings. The buildings in our example consist of up to 7 primitives which were combined by a Boolean union operation. No other Boolean operators were required. A total number of 48 primitives were used, more than 50% of them resembling saddle back roofs. Of these 48 primitives, 13 or 27% could be measured automatically. Automatic fine measurement turned out to be rather difficult in many cases, especially with the buildings on the southern side of the road because some of them did not fit to the primitives exactly (cf. example 4 in section 7.1.4, especially figure 7.7), and because for most of them the lack of contrast between the northern roofs and the shadows caused feature extraction to fail. By tuning some of the matching parameters it would have been possible to measure a greater number of primitives automatically, but in this test, only the default parameters were used. For instance, automatic fine measurement succeeded for the only hip roof in the test area (right upper part in figure 7.8) if more features were used (i.e., if a smaller multiplication constant j for feature extraction than the default one were used) and if matching started at pyramid level 1 (60 μm). In addition, only those primitives were assumed to be matched correctly which exactly fit to the image data. In some cases, the fit could not be examined properly because it was even hard for a human operator to interpret the images. In the cases where the model did not fit well to the image data because, for instance, the building was not really orthogonal, the results of automatic fine measurement could have been accepted, another primitive having to be added to the building using a Boolean difference operator in order to "cut off" parts of the model being outside the actual building.

A 2.5D grid DTM of the test area was derived by stereoscopic measurement using an analytic plotter so that the floor heights of the buildings could be determined. The DTM was derived using the program *SCOP* with a grid width of 25 m. Both the results of building extraction and the DTM were stored in *SCOP.TDM* in the way described in section 3.2.2. From these data, VRML models of the test area could be derived. The VRML model of the terrain which was created by the program *SCOP.ATM* and the VRML model containing the buildings were combined manually. Two views of the resulting combined VRML model of our test area are presented in figure 7.9.

Even though our test area is a part of a small village, it is rather densely built-up. The fringes of the village are characterized by stand-alone buildings which can be reconstructed automatically with greater ease than the buildings in the centre. Thus, one can say that the experience made in the course of the test project is quite promising. Although the number of primitives that could be measured automatically is not too high, we know how the success rate of the process could be improved (tuning parameters, editing the mathematical formulation of parts of the primitives). A considerable amount of work was saved not only by using automatic fine measurement, but also by the interactive tools. Topology-based interactive measurement turned out to be very efficient.

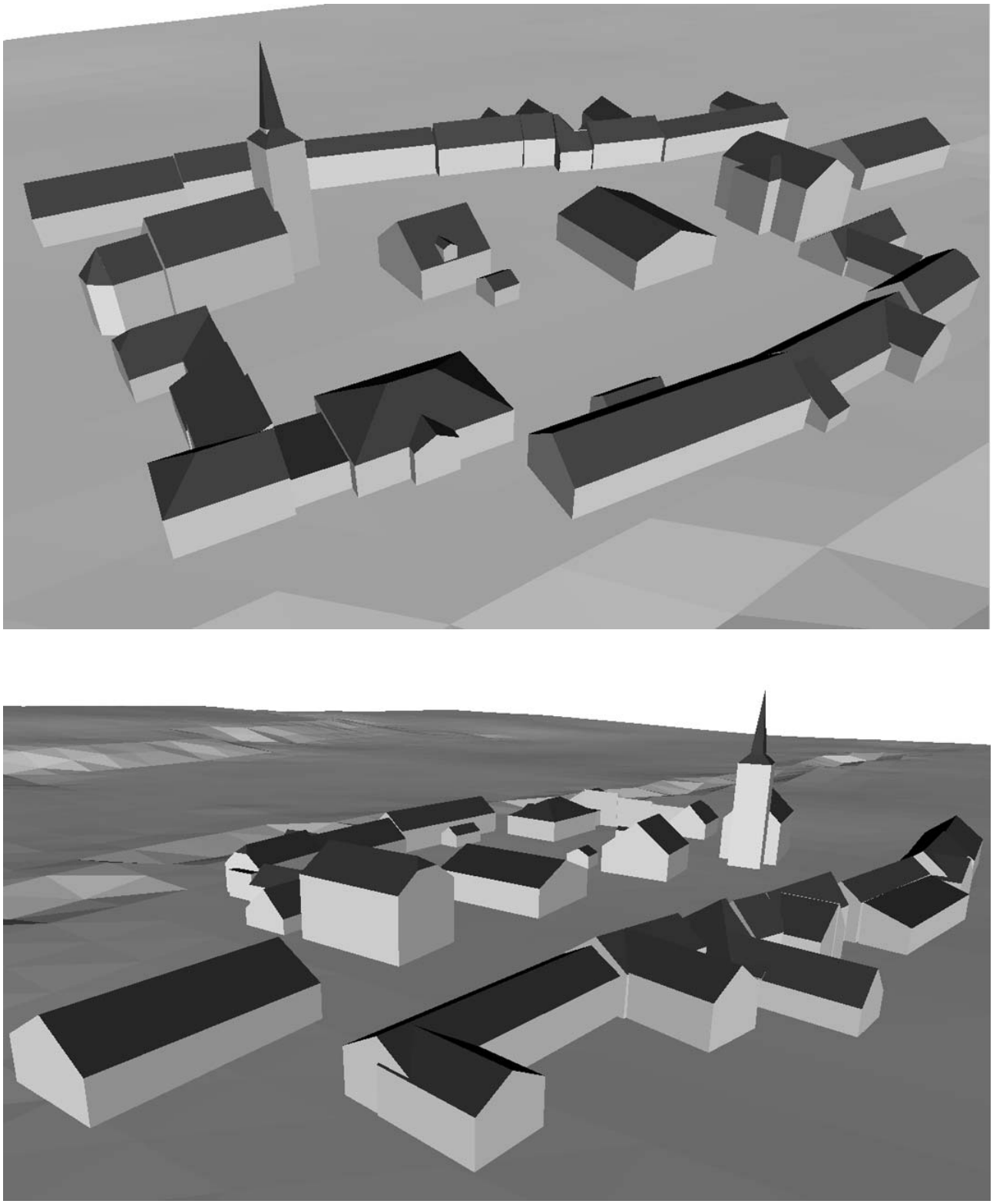


Figure 7.9: Two screen shots of a view on the VRML model of the test area.

Part IV

Conclusion

Chapter 8

Conclusion and future work

In this work, a new system for semi-automatic building extraction has been described. The system is embedded in the program *ORPHEUS* for digital photogrammetric plotting. *ORPHEUS* offers a graphics user interface for the adjustment system *ORIENT* as well as modules for visualization of and interactive measurement in digital images. The core of the system is a new approach for modelling building primitives based on *ORIENT*'s possibilities for the formulation of surface observations and the integration of hybrid adjustment which supports both the interactive tools and the automated modules of the system. The automated modules are based on a general framework for object surface reconstruction which also has been presented in this work.

The degree of automation which can be achieved in object reconstruction depends on the complexity of the task to be solved: in some cases such as DEM generation for topographic mapping, the degree of automation can be very high, whereas in building extraction, human interaction remains an important part of the work flow even though the amount of work to be done by the human operator can be reduced considerably.

Along with the system for the acquisition of 3D building models, a method for the management of these data in a TIS has been developed. This method should be applicable on a national scale, i.e., it should be possible to manage the buildings of a whole state such as Austria. The method is based on an adaptation of an existing technique for the management of digital terrain models as it is used in the program *SCOP.TDM*.

There are still ample possibilities for the improvement of the operability of the new system. The most important ones to be tackled in the future are:

1. **Interactive measurement:** In the prototype system, interactive measurement is restricted to the measurement of building vertices in the digital images. There are some additional tools which might increase the performance of the system:

   - In densely built-up urban areas, the vertices of the primitives are often not visible in the digital images because they are inside another primitive which has to be intersected with the current one. In order to be able to determine the parameters of such buildings by interactive measurement without having to do rather rough estimations of the (invisible) building vertices, it would be desirable to offer the possibility to measure points on the building edges in the digital images rather than the vertices. From the point of view of the determination of the building parameters, no additional work has to be done. The only problem concerned with this enhanced mode of operation which still lacks a solution is the determination of the approximate values for the rotation angle of the observation co-ordinate system of the primitive with respect to the object co-ordinate system.
   - Another important feature to be implemented in the future is a tool for snapping existing building vertices in order to declare them to be identical to vertices of the current primitive.
   - Face glueing as a fourth possibility for the combination of primitives (in addition to the three Boolean operators) is not yet included in the system.

- In the current implementation of the Boolean operators by the *VRaniML*^{TM} library, two points are considered to be identical if their co-ordinates differ by less than 10^{-5} m. Similar thresholds are used to determine whether two lines intersect in object space and whether a point is situated on a plane. These thresholds are not applicable for real-world projects because they will be hurt in all cases. It is desirable to use more realistic thresholds for the comparisons required for the Boolean operators.
- In the data base of common building shapes, symmetry assumptions about the primitive are modelled implicitly and thus have to be fulfilled strictly. In order to simplify the reconstruction of buildings which correspond to a primitive from the data base with the exception of a small detail (cf. the building in figure 7.7 in section 7.1.4), it would be desirable for the user to be able to edit the properties of the primitive so that, for instance, one wall could be declared not to be orthogonal to its neighbours although it is so in the data base.
- In addition to the possibilities already offered by the prototype system, a tool for a numeric input of the primitive parameters would be helpful.

2. **Automatic fine measurement:** The prototype module gives satisfactory results. The following improvements are desirable in order to make it operational:

   - In the current version, the computation time required for the reconstruction of a primitive is still too long. 50% of the computation time is consumed by feature extraction. It would be desirable to perform feature extraction in advance in a pre-processing step so that in the course of automatic fine measurement, the relevant features only have to be read from a file. In order to accomplish this, an efficient way of making the feature adjacency graphs of all digital images persistent on disk has to be developed.
   - In order to speed up the evaluation of correspondence hypotheses, the superfluous unknowns (i.e., the object co-ordinates of the points inserted in the matching phase) have to be eliminated from the normal equation system, e.g., by using new observation types in the way already discussed in section 6.4.2.
   - In the current version, our system is restricted to using perspective photographs for building reconstruction. However, this is a limitation of the current implementation status of the *ORIENT* data base interface. In the future, the interface to the transformation functions of other imaging sensors (especially for line scanners) has to be implemented. As soon as this interface is available, our method will work for other sensors than perspective cameras as well.
   - In the current version, the feature extraction module can handle grey level images only. As colour information is very important for an easy interpretation of the images by the human operator, feature extraction has to be expanded to be applicable to colour images.

3. **Data management in a TIS:** The object oriented interface for topographic objects described in section 3.2.2 has only been partly implemented up to now. The parts which are still missing have to be implemented in the future, and other application programs have to be adjusted to the new data structure.

Besides the improvement and a further evaluation of the semi-automatic system, the degree of automation for building extraction shall be increased by integrating the data from other sensors into the process by data fusion techniques. It is one of the greatest advantages of the method for automatic fine measurement described in this work that it is in no way restricted to the primitives of the data base of common building shapes, but it can be applied to any polyhedral building shape. We want to use 3D laser scanner data for building extraction and the creation of a coarse polyhedral building model. In a recent diploma thesis at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology, it has been shown that points on the terrain can be separated from other points by applying robust estimation of DEM parameters in a hierarchical manner [Briese, 2000]. In the future we want to investigate techniques for further separating points on buildings from points reflected by vegetation or other objects. After that, the shapes of the regions classified as “buildings”

have to be further analyzed in order to detect co-planar regions. These co-planar regions can be approximated by planes, and the planes have to be grouped in order to obtain consistent building models. In this context, the results of the present work can be used in two ways:

1. The principle of “surface observations” can also be used to determine the parameters of planar faces from the laser scanner data: The laser scanner points can be seen as “observed object points”, i.e., as control points, and for each laser scanner point, a surface equation for the height can be inserted into adjustment. Even though the grouping rules for generic building modelling still have to be found, our previous work turns out to be quite useful in the new context.

2. A building model created from an analysis of the laser scanner points can be used just like a “primitive” in automatic fine measurement for semi-automatic building extraction. That is why the automated module can be used as a plug-in to increase the accuracy of the results obtained from analyzing the laser scanner data.

A prototype system for semi-automatic building extraction based on the integration of object parameter estimation into the photogrammetric process has been implemented, and it has also been tested in the course of a project carried out in Stoitzendorf (Lower Austria). A data base of common building primitives is provided by the system as well as tools for an interactive determination of the primitive parameters and for automatic fine measurement. The interactive tools turned out to be quite efficient because few user interactions were required to position the building models in the images. Even in the cases where the automatic determination of the building parameters failed, the work flow of building extraction was sped up significantly by using the model knowledge provided by the data base of common building shapes and by applying Boolean operators for the combinations of primitives. The results of automatic fine measurement are very promising. It has been shown that the accuracy of the results of the automated tools can be better than the one that can be achieved by manual measurement. However, these results have to be confirmed in a more elaborate test which has to be performed in the future. In that future test, it will be necessary to determine the building parameters independently from the automated measuring process. We think that after considering the improvements proposed in this chapter, our method for semi-automatic building extraction will be applicable under quite difficult circumstances such as those encountered, for instance, in the City of Vienna: Even though the buildings of Stoitzendorf have been constructed in the course of quite a long time and thus are characterized by quite a variability of shapes, the applicability of the system in urban areas still remains to be evaluated, too.

Bibliography

[Ackermann, 1984] Ackermann, F. (1984). High Precision Digital Image Correlation. In *Proceedings of the 39^{th} Photogrammetric Week*, volume 9 of *Schriftenreihe der Universität Stuttgart*, Stuttgart.

[Ackermann and Hahn, 1991] Ackermann, F. and Hahn, M. (1991). Image Pyramids for Digital Photogrammetry. In Ebner, H. and Fritsch, D. and Heipke, C., editor, *Digital Photogrammetric Systems*, pages 43–59, Karlsruhe, Germany. Wichmann.

[Ameri, 2000] Ameri, B. (2000). Feature based model verification (fbmv): A new concept for hypothesis validation in building reconstruction. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 24–35, Amsterdam.

[Ameri and Fritsch, 1999] Ameri, B. and Fritsch, D. (1999). 3-d Reconstruction of Polyhedral-like Building Models. In Ebner, H., Eckstein, W., Heipke, C., and Mayer, H., editors, *Proceedings ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, volume XXXII/3-2W5 of *International Archives of Photogrammetry and Remote Sensing*, pages 15–20, Munich, Germany.

[Amhar, 1997] Amhar, F. (1997). *A Contribution to Geometrically Correct Digital Orthophoto under Consideration of General 3D Objects and its Realization in a Software Package*. PhD thesis, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Baillard et al., 1999] Baillard, C., Schmid, C., Zisserman, A., and Fitzgibbon, A. (1999). Automatic Line Matching and 3D Reconstruction of Buildings from Multiple Views. In Ebner, H., Eckstein, W., Heipke, C., and Mayer, H., editors, *Proceedings ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, volume XXXII/3-2W5 of *International Archives of Photogrammetry and Remote Sensing*, pages 69–80, Munich, Germany.

[Baltsavias, 1991] Baltsavias, E. (1991). *Multiphoto Geometrically Constrained Matching*. PhD thesis, Institute of Geodesy and Photogrammetry, ETH Zürich. Mitteilungen Nr. 49.

[Baltsavias and Mason, 1997] Baltsavias, E. and Mason, S. (1997). Automated Shack Reconstruction Using Integration of Cues in Object Space. In *Proceedings ISPRS Commission III / IV Workshop*, volume XXXII-3-4W2 of *International Archives of Photogrammetry and Remote Sensing*, pages 96–105, Stuttgart.

[Baumgart, 1975] Baumgart, B. (1975). A polyhedron representation for computer vision. In *National Computer Conference*, pages 589–596. AFIPS Conf. Proc.

[Baumgartner et al., 1999] Baumgartner, A., Steger, C., Mayer, H., Eckstein, W., and Ebner, H. (1999). Automatic Road Extraction in Rural Areas. In Ebner, H., Eckstein, W., Heipke, C., and Mayer, H., editors, *Proceedings ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, volume XXXII/3-2W5 of *International Archives of Photogrammetry and Remote Sensing*, pages 107–112, Munich, Germany.

[Bill and Fritsch, 1991] Bill, R. and Fritsch, D. (1991). *Grundlagen der Geo-Informationssysteme*. Herbert Wichmann Verlag, Karlsruhe, Germany.

[Brandstätter, 1991] Brandstätter, G. (1991). Notitzen zur voraussetzungslosen gegenseitigen Orientierung von Meßbildern. *Österreichische Zeitschrift für Vermessungswesen und Photogrammetrie*, 79/4:273–280.

[Brenner, 1999] Brenner, C. (1999). Erfassung von 3D-Stadtmodellen. In Fritsch, D., Brenner, C., Haala, N., and Walter, V., editors, *Tutorium Phowo '99: Algortithmen und ihre Automatisierung in der photogrammetrischen Datenauswertung*, Stuttgart, Germany. Institute for Photogrammetry at Stuttgart University.

[Brenner, 2000] Brenner, C. (2000). Towards fully automatic generation of city models. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 85–92, Amsterdam.

[Briese, 2000] Briese, C. (2000). Digitale Modelle aus Laser-Scanner-Daten in städtischen Gebieten. Diploma Thesis, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Brügelmann and Förstner, 1992] Brügelmann, R. and Förstner, W. (1992). Noise Estimation for Color Edge Extraction. In Förstner, W. and Ruwiedl, S., editors, *Robust Computer Vision*, pages 90–106, Karlsruhe, Germany. Wichmann.

[Brunn, 1998] Brunn, A. (1998). Techniques for Automatic Building Extraction. In *Third Course in Digital Photogrammetry*, Bonn, Germany. Institute for Photogrammetry at Bonn University and Landesvermessungsamt Nordrhein-Westfalen.

[Brunn and Weidner, 1997] Brunn, A. and Weidner, U. (1997). Extracting Buildings from Digital Surface Models. In Baltavias, E., Eckstein, W., Gülch, E., Hahn, M., Stallmann, D., Tempfli, K., and Welch, R., editors, *Proceedings ISPRS Commission III/IV Workshop on 3D Reconstruction and Modelling of Topographic Objects*, volume XXXII/3-4W2 of *International Archives of Photogrammetry and Remote Sensing*, pages 27–34, Stuttgart, Germany.

[Burge and Burger, 2000] Burge, M. and Burger, W. (2000). Structural Object Recognition. In Kropatsch, W. and Bischof, H., editors, *Digital image analysis: selected techniques and applications*, pages 249–262, New York. Springer.

[Canny, 1983] Canny, J. (1983). Finding Edges and Lines in Images. Technical Report 720, MIT Artificial Laboratory, Massachussetts.

[Canny, 1986] Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

[Drewniok and Rohr, 1996] Drewniok, C. and Rohr, K. (1996). Automatic Exterior Orientation of Aerial Images in Urban Environments. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 146–152, Vienna.

[Ebner and Reiß, 1978] Ebner, H. and Reiß, P. (1978). Height interpolation by the method of finite elements. *Nachrichten aus dem Karten- und Vermessungswesen*, pages 79–94.

[Englert, 1998] Englert, R. (1998). *Learning Model Knowledge for 3D Building Reconstruction*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, Institute of Computer Science III, Bonn, Germany.

[Englert and Gülch, 1996] Englert, R. and Gülch, E. (1996). One-Eye Stereo System for the Acquisition of Complex 3D-Building Structures. *GEO-INFORMATION-SYSTEMS, Journal for Spatial Information and Decision Making*, 9(4):16–21.

[Faugeras et al., 1992] Faugeras, O., Fua, P., Hotz, B., Ma, R., Robert, L., Thonnat, M., and Zhang, Z. (1992). Quantitative and Qualitative Comparison of some Area and Feature-Based Stereo Algorithms. In Förstner, W. and Ruwiedl, S., editors, *Robust Computer Vision*, pages 1–26, Karlsruhe, Germany. Wichmann.

[Forkert et al., 1995] Forkert, G. Kerschner, M., Prinz, R., and Rottensteiner, F. (1995). Reconstruction of Free-formed Spatial Curves from Digital Images. In *Proceedings of the ISPRS WG 5 Symposium*, volume XXX-5W1 of *International Archives of Photogrammetry and Remote Sensing*, pages 163–168, Zurich, Switzerland.

[Förstner, 1978] Förstner, W. (1978). *Die Suche nach groben Fehlern in photogrammetrischen Lageböcken*, volume C-240. Deutsche Geodätische Kommission, München.

[Förstner, 1986] Förstner, W. (1986). A Feature Based Correspondence Algorithm for Image Matching. In *Proceedings ISPRS Commission III Symposium*, volume XXVI-3/3 of *International Archives of Photogrammetry and Remote Sensing*, pages 150–166, Rovaniemi, Finland.

[Förstner, 1991] Förstner, W. (1991). *Statistische Verfahren für die automatische Bildanalyse und ihre Bewertung bei der Objekterkennung und -Vermessung*, volume C-370. Deutsche Geodätische Kommission, München.

[Förstner, 1998] Förstner, W. (1998). Robust Estimation Procedures in Computer Vision. In *Third Course in Digital Photogrammetry*, Bonn, Germany. Institute for Photogrammetry at Bonn University and Landesvermessungsamt Nordrhein-Westfalen.

[Förstner and Gülch, 1987] Förstner, W. and Gülch, E. (1987). A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features. In *ISPRS Intercommission Workshop*, pages 281–305, Interlaken.

[Fritsch and Ameri, 1998] Fritsch, D. and Ameri, B. (1998). Geometric Characteristics of Digital Surfaces: A Key towards 3-D Building Reconstruction. In Schenk, T. and Habib, A., editors, *Proceedings ISPRS Commission III Symposium*, volume XXXII-3/1 of *International Archives of Photogrammetry and Remote Sensing*, pages 119–126, Columbus, OH.

[Fuchs, 1998] Fuchs, C. (1998). *Extraktion polymorpher Bildstrukturen und ihre topologische und geometrische Gruppierung*. PhD thesis, Institute of Photogrammetry, University of Bonn. Deutsche Geodätische Kommission Volume 502.

[Fuchs and Heuel, 1998] Fuchs, C. and Heuel, S. (1998). Feature Extraction. In *Third Course in Digital Photogrammetry*, Bonn, Germany. Institute for Photogrammetry at Bonn University and Landesvermessungsamt Nordrhein-Westfalen.

[Gonzalez and Wintz, 1977] Gonzalez, R. and Wintz, P. (1977). *Digital Image Processing*. Addison-Wesley, Reading, Massachusetts.

[Gotthardt, 1968] Gotthardt, E. (1968). *Einführung in die Ausgleichungsrechnung*. Herbert Wichmann Verlag Karlsruhe, Germany.

[Great Hill Corporation, 2000] Great Hill Corporation (2000). *http://www.greathill.com*.

[Gruen and Wang, 1998] Gruen, A. and Wang, X. (1998). CC-Modeler: A Topology Generator for 3-D City Models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(5):286–295.

[Gsandtner and Kager, 1988] Gsandtner, M. and Kager, H. (1988). An Out-of-Core Solution of Normal Equations Providing also Accuracy and Reliability Data. In *Proceedings of the XVI^{th} ISPRS Congress*, volume XXVII of *International Archives of Photogrammetry and Remote Sensing*, pages 52–59, Kyoto.

[Gülch, 1994] Gülch, E. (1994). *Erzeugung digitaler Geländemodelle durch automatische Bildzuordnung*. PhD thesis, Institute of Photogrammetry, University of Stuttgart. Deutsche Geodätische Kommission Volume 418.

[Gülch, 2000] Gülch, E. (2000). Digital systems for automated cartographic feature extraction. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B2 of *International Archives of Photogrammetry and Remote Sensing*, pages 241–256, Amsterdam.

[Haala et al., 1997] Haala, N., Brenner, C., and Anders, K.-H. (1997). Generation of 3D City Models from Digital Surface Models and 2D GIS. In Baltavias, E., Eckstein, W., Gülch, E., Hahn, M., Stallmann, D., Tempfli, K., and Welch, R., editors, *Proceedings ISPRS Commission III/IV Workshop on 3D Reconstruction and Modelling of Topographic Objects*, volume XXXII/3-4W2 of *International Archives of Photogrammetry and Remote Sensing*, pages 68–75, Stuttgart, Germany.

[Haala et al., 1998] Haala, N., Brenner, C., and Anders, K.-H. (1998). Urban GIS from Laser Altimeter and 2D Map Data. In Schenk, T. and Habib, A., editors, *Proceedings ISPRS Commission III Symposium*, volume XXXII-3/1 of *International Archives of Photogrammetry and Remote Sensing*, pages 339–346, Columbus, OH.

[Halmer et al., 1996] Halmer, A., Heitzinger, D., and Kager, H. (1996). 3D-Surface Modelling with Basic Topologic Elements. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B4 of *International Archives of Photogrammetry and Remote Sensing*, pages 407–412, Vienna.

[Heipke, 1990] Heipke, C. (1990). *Integration von Bildzuordnung, Punktbestimmung, Oberflächenrekonstruktion und Orthoprojektion innerhalb der digitalen Photogrammetrie*. PhD thesis, Institute of Photogrammetry, University of Technology of Munich. Deutsche Geodätische Kommission Volume 366.

[Heipke, 1997] Heipke, C. (1997). Automation of Interior, Relative, and Absolute Orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 52(1):1–20.

[Heitzinger, 1996] Heitzinger, D. (1996). 3D - Oberflächenmodellierung mit topologischen Grundelementen. Diploma Thesis, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Henricsson et al., 1996] Henricsson, O., Bignone, F., Willuhn, W., Ade, F., Kübler, O., Baltsavias, E., Mason, S., and Grün, A. (1996). Project AMOBE: Strategies, Current Status and Future Work. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 321–330, Vienna.

[Hinz et al., 2000] Hinz, A., Dörstel, C., and Heier, H. (2000). Digital Modular Camera: System concept and data processing workflow. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B2 of *International Archives of Photogrammetry and Remote Sensing*, pages 164–171, Amsterdam.

[Hochstöger, 1996] Hochstöger, F. (1996). Software for Managing Country-Wide Digital Elevation Data. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B2 of *International Archives of Photogrammetry and Remote Sensing*, Vienna.

[Kager, 1980] Kager, H. (1980). Das interaktive Programmsystem ORIENT im Einsatz. In *Proceedings of the XIV^{th} ISPRS Congress*, volume XXIII of *International Archives of Photogrammetry and Remote Sensing*, pages 390–401, Hamburg.

[Kager, 1989] Kager, H. (1989). ORIENT: A Universal Photogrammetric Adjustment System. In Grün, A. and Kahmen, H., editors, *Optical 3-D Measurement*, pages 447–455, Karlsruhe, Germany. Herbert Wichmann Verlag.

[Kager, 1995] Kager, H. (1995). *The ORIENT Manual*. Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Kager, 2000] Kager, H. (2000). Adjustment of algebraic surfaces by least squared distances. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 472–479, Amsterdam.

[Kager et al., 2000] Kager, H., Kerschner, M., Stadler, P., and Rottensteiner, F. (2000). *ORPHEUS User Manual*. Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Kerschner, 1995] Kerschner, M. (1995). Kantenextraktion aus digitalen Bildern und Verfolgung glatter Linien. Diploma Thesis, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Klein and Förstner, 1984] Klein, H. and Förstner, W. (1984). Realization of automatic error detection in the block adjustment program PAT-M43 using robust estimators. In *Proceedings of the 15th ISPRS Congress*, volume XXV-3a of *International Archives of Photogrammetry and Remote Sensing*, pages 234–245, Rio, Brazil.

[Koehl, 1997] Koehl, M. (1997). Topologic Models for Geometric Reconstruction. In Baltavias, E., Eckstein, W., Gülch, E., Hahn, M., Stallmann, D., Tempfli, K., and Welch, R., editors, *Proceedings ISPRS Commission III/IV Workshop on 3D Reconstruction and Modelling of Topographic Objects*, volume XXXII/3-4W2 of *International Archives of Photogrammetry and Remote Sensing*, pages 189–195, Stuttgart, Germany.

[Koehl and Grussenmeyer, 1998] Koehl, M. and Grussenmeyer, P. (1998). 3D data acquisition and modelling in a Topographic Information System. volume XXXII-4 of *International Archives of Photogrammetry and Remote Sensing*, pages 314–319, Stuttgart.

[Köstli and Sigle, 1986] Köstli, A. and Sigle, M. (1986). Die SCOP Datenstruktur zur Verschneidung und Korrektur von Geländemodellen. In *Proceedings ISPRS Commission III Symposium*, volume XXVI-3 of *International Archives of Photogrammetry and Remote Sensing*, Rovaniemi, Finland.

[Krattenthaler et al., 1993] Krattenthaler, W., Mayer, K., and Duwe, H.-P. (1993). 3D-Surface Measurement with Coded Light Approach. In Pölzleitner, W. and Wenger, E., editors, *Image Analysis and Synthesis, Proceedings of the 17th OeAGM Workshop*, volume 68 of *Schriftenreihe der OCG*, pages 103–114. R. Oldenbourg Verlag Wien - München.

[Kraus, 1993] Kraus, K. (1993). *Photogrammetry Volume 1. Fundamentals and Standard Processes*. Dümmler Verlag, Bonn, Germany, fourth edition. With contributions by Peter Waldhäusl.

[Kraus, 1997] Kraus, K. (1997). *Photogrammetry Volume 2. Advanced Methods and Applications*. Dümmler Verlag, Bonn, Germany, fourth edition. With contributions by J. Jansa and H. Kager.

[Kraus, 2000] Kraus, K. (2000). *Photogrammetrie Band 3. Topographische Informationssysteme*. Dümmler Verlag, Köln, Germany, first edition.

[Kraus and Pfeifer, 1998] Kraus, K. and Pfeifer, N. (1998). Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53:193–203.

[Kraus and Ries, 1999] Kraus, K. and Ries, C. (1999). Erfassung und Visualisierung von 3D-Welten. In Heipke, C. and Mayer, H., editors, *Festschrift für Prof. Dr.-Ing. Heinrich Ebner zum 60. Geburtstag*, pages 161–169, Munich, Germany. TU München.

[Kropatsch, 1991] Kropatsch, W. (1991). Image Pyramids and Curves. An Overview. Technical report, Department for Pattern Recongition and Image Processing of the Institute of Automation, University of Technology, Vienna, Vienna, Austria.

[Kropatsch et al., 2000a] Kropatsch, W., Bischof, H., and Englert, R. (2000a). Hierarchies. In Kropatsch, W. and Bischof, H., editors, *Digital image analysis: selected techniques and applications*, pages 211–230, New York. Springer.

[Kropatsch et al., 2000b] Kropatsch, W., Burge, M., and Glantz, R. (2000b). Graphs in Image Analysis. In Kropatsch, W. and Bischof, H., editors, *Digital image analysis: selected techniques and applications*, pages 191–210, New York. Springer.

[Krzystek, 1991] Krzystek, P. (1991). Fully Automatic Measurement of DEM with Match-T. In *Proceedings of the 43rd Photogrammetric Week*, volume 15 of *Schriftenreihe der Universität Stuttgart*, pages 203–213, Stuttgart.

[Krzystek, 1995] Krzystek, P. (1995). Generation of Digital Elevation Models. In *Second Course in Digital Photogrammetry*, Bonn, Germany. Institute for Photogrammetry at Bonn University and Landesvermessungsamt Nordrhein-Westfalen.

[Krzystek and Wild, 1992] Krzystek, P. and Wild, D. (1992). Experimental accuracy analysis of automatically measured digital Terrain models. In Förstner, W. and Ruwiedl, S., editors, *Robust Computer Vision*, pages 372–389, Karlsruhe, Germany. Wichmann.

[Lang, 1999] Lang, F. (1999). *Geometrische und semantische Rekonstruktion von Gebäuden durch Ableitung von Gebäudeecken*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, Institute of Computer Science III, Bonn, Germany.

[Lang and Förstner, 1998] Lang, F. and Förstner, W. (1998). Matching techniques. In *Third Course in Digital Photogrammetry*, Bonn, Germany. Institute for Photogrammetry at Bonn University and Landesvermessungsamt Nordrhein-Westfalen.

[Leonardis and Bischof, 2000] Leonardis, A. and Bischof, H. (2000). Robust Methods. In Kropatsch, W. and Bischof, H., editors, *Digital image analysis: selected techniques and applications*, pages 231–248, New York. Springer.

[Li et al., 1991] Li, J.-C., Schenk, T., and Toth, C. (1991). Towards an Autonomous System for Orienting Digital Stereopairs. *Photogrammetric Engineering and Remote Sensing*, 57(8):1057–1064.

[Loitsch and Molnar, 1991] Loitsch, J. and Molnar, L. (1991). A Relational Database Management System with Topological Elements and Topological Operators. In *Proceedings Spatial Data 2000*, pages 250–259. Department of Photogrammetry and Surveying, University College London.

[Mäntylä, 1988] Mäntylä, M. (1988). *An Introduction to Solid Modeling. Principles of Computer Science*. Computer Science Press, Maryland, U.S.A.

[Meyer, 1990] Meyer, B. (1990). *Objektorientierte Softwareentwicklung*. Hansa-Verlag.

[Mikhail and Ackermann, 1976] Mikhail, E. M. and Ackermann, F. (1976). *Observations and Least Squares*. Dun-Donnelley, New York, U.S.A.

[Mischke and Rottensteiner, 1997] Mischke, A. and Rottensteiner, F. (1997). Feature Extraction in an On-line Engineering Surveying System. In W., B. and M., B., editors, *Pattern Recognition 1997, Proceedings of the 21th OeAGM Workshop*, volume 103 of *Schriftenreihe der OCG*, pages 143–149. R. Oldenbourg Verlag Wien - München.

[Molnar et al., 1996] Molnar, L., Wintner, J., and Wöhrer, B. (1996). DTM System SCOP in a New Technological Generation. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B4 of *International Archives of Photogrammetry and Remote Sensing*, pages 569–574, Vienna.

[Müller, 1998] Müller, H. (1998). Experiences with Semiautomatic Building Extraction. In *Third Course in Digital Photogrammetry*, Bonn, Germany. Institute for Photogrammetry at Bonn University and Landesvermessungsamt Nordrhein-Westfalen.

[Nalwa and Binford, 1986] Nalwa, V. and Binford, T. O. (1986). On Detecting Edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–714.

[Paško and Gruber, 1996] Paško, M. and Gruber, M. (1996). Fusion of 2D GIS Data and Aerial Images for 3D Building REconstruction. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 257–260, Vienna.

[Pfeifer et al., 1999a] Pfeifer, N., Köstli, A., and Kraus, K. (1999a). Restitution of airborne laser scanner data in wooded area. *GIS Geo–Information–Systems*, 12:18–21.

[Pfeifer and Pottmann, 1996] Pfeifer, N. and Pottmann, H. (1996). Surface models on the basis of a triangular mesh – surface reconstruction. In *International Archives of Photogrammetry and Remote Sensing, Vol. XXXI, IWG III/IV*, pages 638–643, Vienna, Austria.

[Pfeifer et al., 1999b] Pfeifer, N., Reiter, T., Briese, C., and Rieger, W. (1999b). Interpolation of high quality ground models from laser scanner data in forested areas. In *International Archives of Photogrammetry and Remote Sensing, Vol. XXXII*, LaJolla, CA.

[PhotoModeler, 2000] PhotoModeler (2000). *http://www.photomodeler.com*.

[Preparata and Shamos, 1990] Preparata, P. and Shamos, M. (1990). *Computational Geometry*. Springer Verlag, New York.

[Prinz, 1995] Prinz, R. (1995). Aerotriangulation mit digitalen Bildern. Diploma Thesis, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Redlich, 1996] Redlich, J.-P. (1996). *CORBA 2.0: Praktische Einführung für C++ und Java*. Addison-Wesley, Bonn, Reading, Mass., et al.

[Rieger et al., 1999] Rieger, W., Kerschner, M., Reiter, T., and Rottensteiner, F. (1999). Roads and buildings from laser scanner data within a forest enterprise. *International Archives of Photogrammetry and Remote Sensing*, Vol. XXXII.

[Rottensteiner, 1993] Rottensteiner, F. (1993). Area-based Matching of Fiducial Marks in Scanned Images. In Pölzleitner, W. and Wenger, E., editors, *Image Analysis and Synthesis, Proceedings of the 17^{th} OeAGM Workshop*, volume 68 of *Schriftenreihe der OCG*, pages 163–172. R. Oldenbourg Verlag Wien - München.

[Rottensteiner, 1996] Rottensteiner, F. (1996). Three Dimensional Object Reconstruction by Object Space Matching. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 692–696, Vienna.

[Rottensteiner, 1998] Rottensteiner, F. (1998). Object Reconstruction in a Bundle Block Environment. In Schenk, T. and Habib, A., editors, *Proceedings ISPRS Commission III Symposium*, volume XXXII-3/1 of *International Archives of Photogrammetry and Remote Sensing*, pages 177–183, Columbus, OH.

[Rottensteiner, 2000] Rottensteiner, F. (2000). Semi-automatic building reconstruction integrated in strict bundle block adjustment. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B2 of *International Archives of Photogrammetry and Remote Sensing*, pages 461–468, Amsterdam.

[Rottensteiner and Prinz, 1996] Rottensteiner, F. and Prinz, R. (1996). Aerotriangulation mit digitalen Bildern: Der Testblock FORSSA der OEEPE. *Österreichische Zeitung für Vermessung und Geoinformation*, 2/96:189–195.

[Samet, 1989] Samet, H. (1989). *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, New York, first edition.

[Schickler, 1992] Schickler, W. (1992). Feature Matching for Outer Orientation of Single Images Using 3-D Wireframe Controlpoints. *International Archives of Photogrammetry and Remote Sensing*, XXIX-B3:591–598.

[Schickler and Poth, 1996] Schickler, W. and Poth, Z. (1996). The Automatic Interior Orientation and its Daily Use. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 746–751, Vienna.

[SCOP, 1994] SCOP (1994). SCOP – Produktinformation. Institut für Photogrammetrie und Fernerkundung, Technische Universität Wien.

[Stadler, 1997] Stadler, P. (1997). Klassenorientierte Formulierung des mathematischen Modells von ORIENT. Diploma Thesis, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology.

[Steger, 2000] Steger, C. (2000). Subpixel-precise extraction of lines and edges. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 141–156, Amsterdam.

[Streilein, 1999] Streilein, A. (1999). *Digitale Photogrammetrie und CAAD*. PhD thesis, Institute of Geodesy and Photogrammetry, ETH Zürich. Mitteilungen Nr. 68.

[Suveg and Vosselman, 2000] Suveg, I. and Vosselman, G. (2000). 3d reconstruction of building models. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 538–545, Amsterdam.

[Tang et al., 1996] Tang, L., Poth, Z., Ohlhof, T., Heipke, C., and Batscheider, J. (1996). Automatic Relative Orientation - Realization and Operational Tests. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 843–848, Vienna.

[Tempelmann et al., 2000] Tempelmann, U., Börner, A., Chaplin, B., Hinsken, L., Mykhalevych, B., Miller, S., Recke, U., Reulke, R., and Uebbing, R. (2000). Photogrammetric software for the LH Systems ADS40 airborne digital sensor. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B2 of *International Archives of Photogrammetry and Remote Sensing*, pages 552–559, Amsterdam.

[Tsingas, 1992] Tsingas, V. (1992). *Automatisierung der Punktübertragung in der Aerotriangulation durch mehrfache digitale Bildzuordnung*. PhD thesis, Institute of Photogrammetry, University of Stuttgart. Deutsche Geodätische Kommission Volume 392.

[van den Heuvel, 2000] van den Heuvel, F. A. (2000). Trends in cad-based photogrammetric measurement. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B5 of *International Archives of Photogrammetry and Remote Sensing*, pages 852–863, Amsterdam.

[Veldhuis, 1998] Veldhuis, H. (1998). Performance Analysis of two Fitting Algorithms for the Measurement of Parametrised Objects. In Schenk, T. and Habib, A., editors, *Proceedings ISPRS Commission III Symposium*, volume XXXII-3/1 of *International Archives of Photogrammetry and Remote Sensing*, pages 400–408, Columbus, OH.

[Vosselman, 1995] Vosselman, G. (1995). Applications of Tree Search Methods in Digital Photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(4):29–37.

[Wang and Gruen, 1998] Wang, X. and Gruen, A. (1998). A 3-D City Model Data Structure and its Implementation in a Relational Database. In *Proceedings of the International Conference of Spatial Information Science and Technology*, pages 429–436, Wuhan, China.

[Weidner, 1997] Weidner, U. (1997). *Gebäudeerfassung aus digitalen Oberflächenmodellen*. PhD thesis, Institute of Photogrammetry, University of Bonn. Deutsche Geodätische Kommission Volume 474.

[Wiedemann and Hinz, 1999] Wiedemann, C. and Hinz, S. (1999). Automatic Extraction and Evaluation of Road Networks from Satellite Imagery. In Ebner, H., Eckstein, W., Heipke, C., and Mayer, H., editors, *Proceedings ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, volume XXXII/3-2W5 of *International Archives of Photogrammetry and Remote Sensing*, pages 95–100, Munich, Germany.

[Wild and Krzystek, 1996] Wild, D. and Krzystek, P. (1996). Automated breakline detection using an edge preserving filter. In *Proceedings of the $XVIII^{th}$ ISPRS Congress*, volume XXXI-B3 of *International Archives of Photogrammetry and Remote Sensing*, pages 946–952, Vienna.

[Wild, 1983] Wild, E. (1983). *Die Prädiktion mit Gewichtsfunktionen und deren Anwendung zur Beschreibung von Geländeflächen*. PhD thesis, Institute of Photogrammetry, University of Stuttgart. Deutsche Geodätische Kommission Volume 217.

[Wrobel, B., 1987] Wrobel, B. (1987). Digitale Bildzuordnung durch Facetten mit Hilfe von Objektraummodellen. *Bildmessung und Luftbildwesen*, 3/87:93–104.

[Yang et al., 2000] Yang, B., Li, Q., and Li, D. (2000). Building model creating and storing in 3d urban gis. In *Proceedings of the XIX^{th} ISPRS Congress*, volume XXXIII-B4 of *International Archives of Photogrammetry and Remote Sensing*, pages 1192–1198, Amsterdam.

[Yuille and Poggio, 1986] Yuille, A. L. and Poggio, T. A. (1986). Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):15–25.

Curriculum vitae of DI. Franz Rottensteiner

June 22, 1967	born in Puchberg / Lower Austria. Parents: Franz Rottensteiner, electrician, and Maria Rottensteiner, tailor
1973-1977	Primary school in Puchberg
1977-1985	Grammar school (“Bundesrealgymnasium”) in Wiener Neustadt
June 4, 1985	Final examination with excellence
1985-1993	Studies in Geodesy at Vienna University of Technology
1990-1992	Studies in Spanish and History at Vienna University
June 7, 1993	Graduation in Geodesy with excellence. Title of diploma thesis: *Flächenbasierte Korrelation von Rahmenmarken in abgetasteten Bildern*
since August 1993	Research assistant at the Institute of Photogrammetry and Remote Sensing at Vienna University of Technology
1994-2000	Financed by the Austrian Research Program S7004-MAT on Pattern Recognition and Digital Image Processing
1996	Alternative service as an ambulance driver in Puchberg / Lower Austria
since July 29, 2000	Married to Andrea Schlacher, occupational therapist, from Judenburg (Styria).

GEOWISSENSCHAFTLICHE MITTEILUNGEN

Bisher erschienen:

Heft 1 Kolloquium der Assistenten der Studienrichtung Vermessungswesen. 1970-1973, Dezember 1976.

Heft 2 EGGER-PERDICH-PLACH-WAGENSOMMERER, Taschenrechner HP 45 und HP 65, Programme und Anwendungen im Vermessungswesen. 1. Auflage, März 1974, Special Edition in English Juli 1974, 2. verbesserte Auflage, November 1974.

Heft 3 Kolloquium der Assistenten der Studienrichtung Vermessungswesen. 1973-1974, September 1974.

Heft 4 EGGER-PALFINGER-PERDICH-PLACH-WAGENSOMMERER, Tektronix-Tischrechner TEK 31, Programmbibliothek für den Einsatz im Vermessungswesen, November 1974.

Heft 5 K. LEDERSTEGER, Die horizontale Isostasie und das isostatische Geoid, Februar 1975.

Heft 6 F. REINHART, Katalog von FK4 Horrebow-Paaren für Breiten von +30 bis +60, Oktober 1975.

Heft 7 Arbeiten aus dem Institut für Höhere Geodäsie, Wien, Dezember 1975.

Heft 8 Veröffentlichungen des Institus für Photogrammetrie zum XIII. Internationalen Kongreß für Photogrammetrie in Helsinki 1976, Wien, Juli 1976.

Heft 9 W. PILLEWIZER, Felsdarstellung aus Orthophotos, Wien, Juni 1976.

Heft 10 PERDICH-PLACH-WAGENSOMMERER, Der Einsatz des programmierbaren Taschenrechners Texas Instruments SR-52 mit Drucker PC100 in der ingenieurgeodätischen Rechetechnik, Wien, Mai 1976.

Heft 11 Kolloquium der Assistenten der Studienrichtung Vermessungswesen. 1974-1976, November 1976.

Heft 12 Kartographische Vorträge der Geodätischen Informationstage 1976, Wien, Mai 1976.

Heft 13 Veröffentlichungen des Institus für Photogrammetrie anläßlich des 80. Geburtstages von Prof. Dr. h.c. K. Neumaier, Wien, Januar 1978.

Heft 14 L. MOLNAR, Self Checking Analytical Relative Orientation and Strip Formation, Wien, Dezember 1978.

Heft 15 Veröffentlichungen des Institus für Landesvermessung anläßlich des 80. Geburtstages von Prof. Dr. Alois Bavir, Wien, Januar 1979.

Heft 16 Kolloquium der Assistenten der Studienrichtung Vermessungswesen. 1976-1978, November 1979.

Heft 17 E. VOZIKIS, Die photographische Differentialumbildung gekrümmter Flächen mit Beispielen aus der Architekturbildmessung, Wien, Dezember 1979.

Heft 18 Veröffentlichungen des Institus für allgemeine Geodäsie anläßlich des 75. Geburtstages von Prof. Dipl.-Ing. Dr. F. Hauer, Die Höhe des Großglockners, Wien, 1981.

Heft 19 H. KAGER, Bündeltriangulation mit indirekt beobachteten Kreiszentren, Wien, April 1981.

Heft 20 Kartographische Vorträge der Geodätischen Informationstage 1980, Wien, Mai 1982.

Heft 21 Veröffentlichungen des Institus für Kartographie anläßlich des 70. Geburtstages von Prof. Dr. Wolfgang Pillewizer: Glaziologie und Kartographie, Wien, Dezember 1982.

Heft 22 K. TEMPFLI, Genauigkeitsschätzung digitaler Höhenmodelle mittels Spektralanalyse, Wien, Mai 1982.

Heft 23 E. CSAPLOVICS, Interpretation von Farbinfrarotbildern, Wien, November 1982.

Heft 24 J. JANSA, Rektifizierung von Multispektral-Scanneraufnahmen - Entwicklung und Erprobung eines EDV-Programms, Wien, Mai 1983.

Heft 25 Zusammenfassung der Diplomarbeiten, Dissertationen und Habilitationen an den geodätischen Instituten der TU Wien, Wien, November 1984.

Heft 26 T. WUNDERLICH, Die voraussetzungsfreie Bestimmung von Refraktionswinkeln, Wien, August 1985.

Heft 27 G. GERSTBACH (Hrsg.), Geowissenschaftliche/geotechnische Daten in Landinformationssystemen - Bedarf und Möglichkeiten in Österreich, Juni 1986.

Heft 28 K. NOVAK, Orientierung von Amateuraufnahmen ohne Paßpunkte, Wien, August 1986.

Heft 29 Veröffentlichungen des Institus für Landesvermessung und Ingenieurgeodäsie, Abt. Ingenieurgeodäsie, anläßlich des 80. Geburtstages von Prof. Dipl.-Ing. Dr. F. Hauer, Wien, Oktober 1986.

Heft 30 K.-H. ROCH, Über die Bedeutung dynamisch ermittelter Parameter für die Bestimmung von Gesteins- und Gebirgseigenschaften, Wien, Februar 1987.

Heft 31 G. HE, Bildverbesserung mittels digitaler Filterung, Wien, April 1989.

Heft 32 F. SCHLÖGELHOFER, Qualitäts- und Wirtschaftlichkeitsmodelle für die Ingenieurphotogrammetrie, Wien, April 1989.

Heft 33 G. GERSTBACH (Hrsg.), Geowissenschaftliche/geotechnische Daten in Landinformationssystemen - Datenbestände und Datenaustausch in Österreich, Wien, Juni 1989.

Heft 34 F. HOCHSTÖGER, Ein Beitrag zur Anwendung und Visualisierung digitaler Geländemodelle, Wien, Dezember 1989.

Heft 35 R. WEBER, Lokale Schwerefeldmodellierung unter Berücksichtigung spektraler Methoden zur Geländereduktion, Wien, April 1990.

Heft 36 o. Prof. Dr. Hans Schmid zum 70. Geburtstag. Veröffentlichung der Abteilung für Landesvermessung, Wien, Oktober 1990.

Heft 37 G. GERSTBACH, H. P. HÖLLRIEGL und R. WEBER, Geowissenschaftliche Informationsbörse - Eine Nachlese zur GeoLIS II, Wien, Oktober 1990.

Heft 38 R. ECKER, Rastergraphische Visualisierungen mittels digitaler Geländemodelle, Wien, August 1991.

Heft 39 Kartographische Forschungen und anwendungsorientierte Entwicklungen, herausgegeben von W. Stams und F. Kelnhofer zum 80. Geburtstag von Prof. Dr. W. Pillewizer, Wien, Juli 1991.

Heft 39a W. RIEGER, Hydrologische Anwendungen des digitalen Geländemodelles, Wien, Juli 1992.

Heft 40 K. STEINNOCHER, Methodische Erweiterungen der Landnutzungsklassifikation und Implementierung auf einem Transputernetzwerk, Wien, Juli 1994.

Heft 41 G. FORKERT, Die Lösung photogrammetrischer Orientierungs- und Rekonstruktionsaufgaben mittels allgemeiner kurvenförmiger Elemente, Wien, Juli 1994.

Heft 42 M. SCHÖNER, W. SCHÖNER, Photogrammetrische und glaziologische Untersuchungen am Gsbre (Ergebnisse der Spitzbergenexpedition 1991), Wien, Mai 1996.

Heft 43 M. ROIC, Erfassung von nicht signalisierten 3D-Strukturen mit Videotheodoliten, Wien, April 1996.

Heft 44 G. RETSCHER, 3D-Gleiserfassung mit einem Multisensorsystem und linearen Filterverfahren, Wien, April 1996.

Heft 45 W. DAXINGER, Astrogravimetrische Geoidbestimmung für Ingenieurprojekte, Wien, Juli 1996.

Heft 46 M. PLONER, CCD-Astrometrie von Objekten des geostationären Ringes, Wien, November 1996.

Heft 47 Zum Gedenken an Karl Killian “Ingenieur” und “Geodät” 1903-1991, Veröffentlichung der Fachgruppe Geowissenschaften, Wien, Februar 1997.

Heft 48 A. SINDHUBER, Ergänzung und Fortführung eines digitalen Landschaftsmodelles mit multispektralen und hochauflösenden Fernerkundungsaufnahmen, Wien, Mai 1998.

Heft 49 W. WAGNER, Soil Moisture Retrieval from ERS Scatterometer Data, Wien, Dezember 1998.

Heft 50 R. WEBER, E. FRAGNER (Editoren), Prof. Bretterbauer, Festschrift zum 70. Geburtstag, Wien, Juli 1999.

Heft 51 Ch. ÖHRENEDER, A Similarity Measure for Global Image Matching Based on the Forward Modeling Principle, Wien, April 1999.

Heft 52 M. LECHTHALER, G. GARTNER (Hrsg.), Per Aspera ad Astra, Festschrift für Fritz Kelnhofer zum 60. Geburtstag, Wien, Jänner 2000.

Heft 53 F. KELNHOFER, M. LECHTHALER (Hrsg.), Interaktive Karten (Atlanten) und Multimedia-Applikationen, Wien, März 2000.

Heft 54 A. MISCHKE, Entwicklung eines Videotheodolit-Meßsystems zur automatischen Richtungsmessung von nicht signalisierten Objektpunkten, Wien, Dezember 2000.

Heft 55 K. KRAUS (Hrsg.), Festkolloquium anlässlich der Emeritierung von Prof. Dr. Peter Waldhäusl, in Vorbereitung.