DGK Deutsche Geodätische Kommission
bei der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 661

Dimitri Bulatov

Textured 3D reconstruction of urban terrain from UAV-borne video sequences

München 2011

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5073-0

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 291, Hannover 2011

DGK Deutsche Geodätische Kommission
bei der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 661

Textured 3D reconstruction of urban terrain from UAV-borne video sequences

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Ing. Dimitri Bulatov

München 2011

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission bei der C. H. Beck'schen Verlagsbuchhandlung München

ISSN 0065-5325 ISBN 978-3-7696-5073-0

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 291, Hannover 2011

Adresse der Deutschen Geodätischen Kommission:

Deutsche Geodätische Kommission

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100

e-mail hornik@dgfi.badw.de ● http://www.dgk.badw.de

Prüfungskommission

Vorsitzender:	Prof. Dr.-Ing. Steffen Schön, Institut für Erdmessung, Leibniz Universität Hannover
Referent:	Prof. Dr.-Ing. habil. Christian Heipke, Institut für Photogrammetrie und GeoInformation, Leibniz Universität Hannover
Korreferenten:	Prof. Dr.-Ing. habil. Stefan Hinz, Institut für Photogrammetrie und Fernerkundung, Karlsruher Institut für Technologie (KIT)
	Prof. Dr.-Ing. habil. Hansjörg Kutterer, Geodätisches Institut Hannover, Leibniz Universität Hannover

Tag der Einreichung der Arbeit: 05.01.2011

Tag der mündlichen Prüfung: 29.04.2011

ISSN 0065-5325 ISBN 978-3-7696-5073-0

Contents

Kurzfassung

Das Ziel dieser Arbeit ist die Entwicklung und Implementierung generischer, vom Modellwissen weitgehend unabhängiger Lösungsstrategien zur texturierten 3D Rekonstruktion urbaner Gebiete aus Videosequenzen. Solche Videosequenzen können sowohl mit einer Tageslicht- als auch Infrarotkamera aufgenommen werden; in unseren Anwendungen handelt es sich überwiegend um luftgetragene Aufnahmen. Die zahlreichen zivilen aber auch militärischen Anwendungsfelder der 3D Erschließung der Szene mit minimalem Aufwand verlangen von den zu entwickelnden Verfahren besondere Robustheit gegenüber Videosequenzen suboptimaler Qualität und kritischen Sensorbewegungen. Auch spielen ein einschätzbarer, parallelisierbarer Rechenaufwand und die Eignung der Verfahren, mit einem theoretisch unendlichen Datenstrom annähernd schritthaltend fertig zu werden, eine wichtige Rolle.

In dieser Arbeit wird vorausgesetzt, dass eine Euklidische Rekonstruktion durch Kameramatrizen (Orientierungen) sowie eine dünne Punktwolke vorliegt. Die entwickelten Methoden sind also in den Forschungsgebieten *Rekonstruktion dichter 3D Punktwolken aus Mehrkamerasystemen* sowie *Kompression dieser Punktwolken in Dreiecksvermaschungen* angesiedelt.

Um eine dichte Punktwolke aus einem Bildverbund zu erhalten, müssen Korrespondenzen einer dichten Menge der Pixel eines sogenannten Referenzframes in anderen Bildern wiedergefunden werden. Formeln zur schnellen Berechnung der vom Referenzframe in andere Bilder projizierten Punkte sind unentbehrlich; die schnellste Möglichkeit ist durch die Disparitätensuche in epipolar rektifizierten Bildern gegeben. Danach werden die Kostenfunktionen (auch Datenkosten genannt) zur effektiven Suche der Punktkorrespondenzen aggregiert. Da diese Datenkostenterme allein auch bei Mehrkamerasystemen nicht ausreichen, um die Tiefenwerte in schwach texturierten Bereichen sowie Bereichen von Verdeckungen und sich wiederholender Muster zu rekonstruieren, muss ein zusätzlicher Glattheitsterm eingeführt werden, der sich auf die Annahme stützt, dass die Tiefen eines überwiegenden Anteils der Pixel ungefähr gleich sind wie die Tiefen ihrer Nachbarn. Da das Finden eines exakten Minimums einer Gesamtkostenfunktion, die aus einem Datenterm, einem 2D Glattheitsterm und einem zusätzlichen, zwecks Ausgleichung von (insbesondere bei Schrägsichtaufnahmen typischen) Diskretisierungsartefakten eingeführten Dreiecksterm besteht, in der Praxis unmöglich ist, werden Approximationsverfahren angewandt. Die *Verallgemeinerung des semiglobalen Algorithmus* auf Multi-view Systemen und die *Benutzung sowie Evaluierung der Dreiecksvermaschungen aus den bereits detektierten Punkten* stellen den wissenschaftlichen *Hauptbeitrag zum bildbasierten Teil* der Funktionsbibliothek dar.

Unter der Annahme, dass sich die Gebäudeoberflächen anhand von Dreiecksvermaschungen zu texturierten Flächensegmenten aggregieren lassen, wurden im Rahmen dieser Dissertation zahlreiche Verfahren zur Rekonstruktion der Oberflächen aus Punktwolken untersucht, weiterentwickelt und bewertet. Am robustesten gegenüber sehr variabler Punktdichte, Rauschen und Ausreißern (weit von der Oberfläche entfernt liegende Punkte, die beispielsweise durch Spiegelungen, Verdeckungen und kleine bewegte Objekte entstehen) hat sich

die auf L_1-*Splines basierender Algorithmus* gezeigt, der den *Hauptbeitrag des punktbasierten Teils* der Arbeit darstellt. Hier kann sowohl die Rekonstruktion einer skalaren Funktion als auch der Übergang zu einer automatisch parametrisierten 3D Oberfläche stattfinden. Im letzten Schritt solcher globalen Verfahren wird zu jedem Dreieck der Vermaschung ein Referenzframe gewählt, in dem das Dreieck vollständig sichtbar ist (Texturierung).

Zur Visualisierung der Ergebnisse wurden zahlreiche Datensätze getestet, die zum Teil anspruchsvolle historische Gebäude darstellen, zum anderen Teil aber zerstörte Gebiete, deren genaue Rekonstruktion mit Hilfe modellbasierter Verfahren kaum möglich ist. Zur quantitativen Bewertung der Verfahren wurde für einen synthetischen und einen realen, mit einer sehr dichten Laserpunktwolke als Ground Truth gegebenen Datensatz die *Hausdorff-Distanz* als Maß für Vollständigkeit und Korrektheit einbezogen.

Im letzten Teil der Arbeit wird zusammenfassend auf die Stärken und Schwächen der vorgestellten Verfahren eingegangen und mögliche Ansätze zur Behebung dieser Schwächen werden erläutert.

Zusammenfassend wird aus der Arbeit ersichtlich, dass sich das vorgestellte Konzept zur qualitativ ansprechenden Rekonstruktion von Gebäuden und urbanem Gelände aus Luftvideos hervorragend eignet.

Summary

The goal of this thesis is development and implementation of a generic procedure for textured 3D reconstruction of urban terrain from video sequences. These video sequences can be recorded by daylight or infrared cameras; in our applications these cameras are mostly mounted onboard airborne sensor platforms. There are numerous civil and military applications of 3D reconstruction from videos obtained from cheap, miniaturized cameras without any other information, but the reconstruction algorithms must be robust enough to process video sequences of limited quality and cope with critical motions and scenes. The parallelizable computation costs, which can be estimated, as well as adequacy of reconstruction procedures to keep step with a theoretically endless data stream play an important role in our considerations.

We assume in this work that an Euclidean Reconstruction is given by a set of extrinsic and intrinsic camera parameters (orientations) corresponding to frames of the given video sequence as well as several 3D points. Two main directions of research will be obtaining *dense 3D point clouds from multi-view systems* and *compressing these point clouds into triangular meshes.*

To extract a dense point cloud from an image sequence, one must be able to perform matching of a dense set of pixels within the so-called reference image of this sequence. We derive fast equations for point projection in other images and obtain initial information by comparing intensities of projected points (data terms). The fastest way to project points is given by considering disparity values from epipolarly rectified image pairs. Alternatively, depth values can be used. In the next step of the matching process, data cost aggregation is carried out over all images. Unfortunately, even for multi-view systems, the data term alone is not sufficient for assigning correct depth values in areas of homogeneous color distribution, repetitive patterns of texture, and near occlusions, so a smoothness term, which encourages neighboring pixels to have similar depth values, must be introduced. Computationally efficient methods must be applied for total energy minimization of a functional consisting of the data term, the 2D smoothness term and an additional triangulation-based smoothness term whose main task consists of reducing discretization artifacts typical for slanted surfaces by biasing depth values towards the triangular mesh from already available points. The *generalization of a semi-global algorithm for energy minimization to the multi-camera systems* as well as *application and evaluation of triangular meshes from already detected points* represent the *principal innovations of the image-based* part of this thesis.

A reasonable assumption that the surface of buildings can be aggregated to polygonal meshes motivated us to investigate, modify and evaluate numerous algorithms for shape reconstruction from point clouds. The best results with respect to varying point density, data noise and a considerable number of outliers (points far away from the surface resulting, for instance, from reflections, occlusions or small moving objects) were obtained with the L_1*-spline-based procedure* for geometric reconstruction which is *the principal contribution of the shape reconstruction portion* of our reconstruction pipeline. This can include either a

reconstruction of a scalar function representing a 2.5D surface or a real 3D surface in an automatically generated parameter domain. The last step of all these methods consists of assigning to every polygon (triangle) in the resulting mesh a reference camera which completely observes it (texturing). Reconstruction results from numerous data sets representing complex historical buildings as well as destroyed structures, which can hardly be modeled with non-generic approaches, demonstrate the effectiveness of our algorithms. As a measure of completeness and correctness for quantitative evaluation of algorithms on a synthetic data set and a simple real data set with a dense laser point cloud as ground truth, the *Hausdorff distance* was used.

The last part of the dissertation summarizes the advantages and disadvantages of the algorithms and introduces concepts for future work for coping for remaining problems.

It becomes clear that the reconstruction procedure presented in this work can be used for obtaining excellent textured 3D models for buildings and surrounding terrain from aerial and UAV-videos.

Chapter 1

Introduction

1.1 Motivation, sensors and requirements

Because of their ability to cover large parts of the scenery, aerial images have always been an extraordinarily attractive tool to gain information. In the past decade, it has become attractive to utilize unmanned aerial vehicles (UAVs) because of their low cost and easy use. The application areas for videos captured by UAV can vary from civil engineering and urban planning to surveillance, automatic navigation, and defense research. Although in the course of this work, external references for sensor platforms are not required, the technical equipment of the miniaturized aerial vehicles has experienced rapid progress in the most recent couple of years: historically, UAVs were simple remotely piloted drones, but autonomous control and capability to carry out pre-programmed flight plans is increasingly being employed in UAVs. Figure 1.1 shows several unmanned sensor platforms used for data acquisition in our work.

From the mathematical point of view, the applications of these videos can be divided into essentially two main categories. On the one hand, the spatial depth is negligible for many applications, such as video stabilization, image-mosaicking, image-based 2D geo-referencing, detection of moving objects and annotation of space-oriented information into the video sequence, see [121]. Real-time algorithms play an indispensable role here because potential threats and targets must be detected in time to take action. For these applications, the (bijective) mapping from view to view can be described by a transformation of the plane, or the so called 2D homography, which is given by a regular 3×3 matrix, and the 3D character of scenes only interferes in the results of the performance wherefore efforts must be taken to exclude its negative effects from consideration (see Fig. 1.2).

On the other hand, algorithms for 3D reconstruction require flights at relatively small altitudes and with slowly flying platforms. Although there are also quasi-3D methods, such as image morphing described in [32], where, given an optical flow function between two or more images, intermediate images can be rendered without explicit computation of the 3D structure of the scenery, an accurate 3D reconstruction from a general configuration of cameras can be achieved only by obtaining structure and motion followed by dense reconstruction.

However, because of the need to open up the *third dimension* out of *two-dimensional* images, the algorithms for 3D reconstruction are time-consuming, and, since our area of applications always lies in the margin zone between 2D and 3D, they are less numerically stable. The lightweight equipment that such aerial vehicles may carry and the local instability that characterizes the paths of these small vehicles result in considerable uncertainty in reconstruction and texturing of terrain. When external references such as GPS are not

available, the uncertainty is larger still, because the drift errors in camera position and orientation negatively influence the results. In addition, the quality of data acquired by small, instable, unmanned sensors is usually much worse than that of typical high-resolution aerial images because of interlacing effects, lens distortions, motion blur and a rather low spatial resolution.

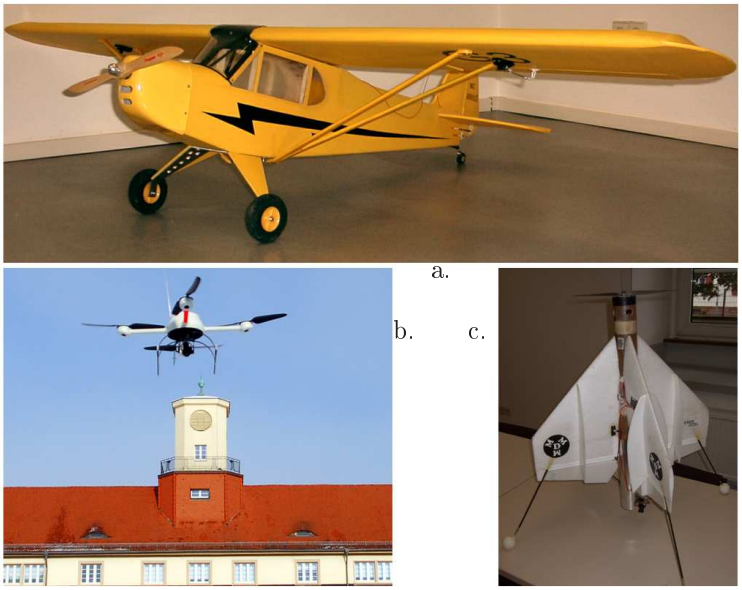

Figure 1.1: a. Piper cup plane is able to carry onboard a unit consisting of a daylight camera and an infrared camera. Since it can achieve a height of up to 100 meters and a velocity of up to 15m/s, it is suitable mainly for 2D applications. b. The md4-quadrocopter is able to store the video data onboard and perform automatic flights. Therefore the data can be evaluated after the mission is completed. c. The m3d-UAV can be operated in hovering and cruising modes.

The majority of the current state-of-the-art object reconstruction methods first retrieves the camera trajectory and the object contours (given by sparse point clouds) and then generates a dense reconstruction with texturing. Although there are several possibilities for visualization, for example, voxels, level-sets, depth maps and polygonal meshes (see Fig. 1.3), we decided to represent our objects by triangular meshes since they provide a more comfortable way for many relevant applications, such as visibility calculation. This is important for automatic navigation while textured models are important for visual impression as well as mission planning to ease user's orientation in the unknown terrain. The other three possibilities will either be mentioned in Chapter 3 (related work) or or will serve as intermediate results in the course of this work. In urban areas, an additional challenge is created by the need for replacement of traditional 2.5D "terrain skins" (representations of height as a univalent function of latitude and longitude) by a fully 3D terrain representation with multivalent height (vertical walls, balconies, overhanging roofs etc.). In many applications, model generation must be performed in a reasonable time, which justifies us to prefer – sometimes – one algorithm because it is faster than another algorithm, even though its performance is slightly worse. Moreover, we will classify our algorithms into *local*, or *close-to-real-time* ones, i. e. those that can process the video sequence either frame by frame or using "short" sub-sequences, and *global* ones that can be applied only after the whole sequence has been captured and processed by local algorithms. Application of global methods for shape-reconstruction on 3D point sets obtained from local methods makes up the most important scientific contribution of our work.

Figure 1.2: Examples of 2D applications: Top left: In almost-planar scenes, detection of moving objects can reliably be performed by means of homographies. In urban scenes, the 3D character of the terrain causes parallaxes which are the main reason for false alarms (e. g. the church tower top right). These false alarms can be successfully eliminated if the video stream is geo-referenced onto the orthophoto (bottom, see also [121]). In this case, it is also possible to estimate the velocities and heading directions of moving objects.

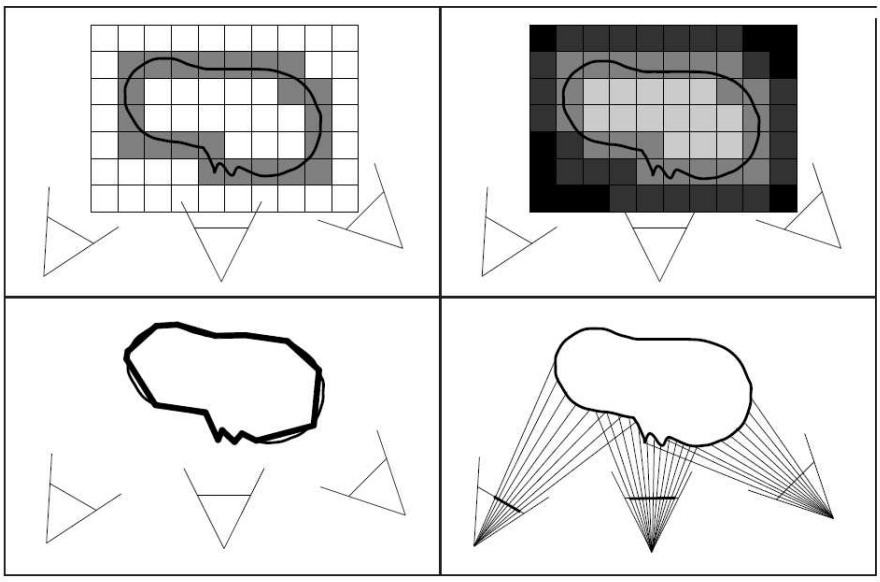

Figure 1.3: Four possibilities for scene (black curve) representation: Voxel grid (top left), level-sets (top right), a triangular mesh, which is the desired output of our work (bottom left) and a depth-map representation (bottom right) (Fig. courtesy of C. Strecha).

1.2 Reconstruction pipeline and organization of this work

As described in the previous section, our goal is to obtain a textured surface from a video sequence. We describe in the two following subsections the outline of the reconstruction procedure and the organization of this work.

1.2.1 Reconstruction pipeline

One popular framework for 3D reconstruction from video sequences in a reasonable time, possibly proportional to the speed of video rendering, consists of three main steps 1) obtaining camera poses and 3D points by means of detecting and tracking characteristic points, 2) creating dense 3D point clouds from several (reference) images, 3) geometric model generation and texturing (see Alg. 1.1)

The first step will not be in the focus of this thesis. For the main references about methods needed to obtain the camera trajectory and a sparse point cloud from (calibrated or uncalibrated) image sequences, we refer to [9, 22, 105]. The second step includes *image-based methods* and will be performed incrementally for several reference frames. Together with Step 3.1 of local tessellations, it has a concept of a *real-time oriented model generation*. The main function of Step 2.1 – *sparse tracking and triangulation* – consists of regularizing the density of points (a process also called *enriching*) since the original point cloud has extremely low density in untextured regions. A coarse visibility information can be generated by a triangular mesh from point sets. To improve and further enhance this visibility information, Step 2.2 is applied. The task of this *dense reconstruction* module is to to provide exact (apart from discretization errors) depth values for every pixel in every (reference) image. *Local tessellations* are needed if there is no time to apply a global method for post-processing. In this case, the reconstruction terminates after Step 3.1. Otherwise, the whole available information – point sets, camera matrices and visibility information – is used in *global approaches*, which make up Step 3.2 of our pipeline. This step consists of retrieving

Input: video sequence	
Step 1: Relative orientation	*% see [9, 22, 105]*
Step 2 Image-based reconstruction	
Step 2.1: Sparse tracking and triangulation	*% see Sec. 4.4*
Step 2.2: Dense reconstruction	*% see Sec. 4.5*
Step 3: Shape reconstruction	
Step 3.1: Local tessellations	*% see Sec. 5.1*
Step 3.2: Global surf. extraction and texturing	*% Global approach, see e. g. Sec. 5.2*
Output: triangular mesh	

Algorithm 1.1: Three main steps of the reconstruction pipeline.

triangulated surfaces, (optional) mesh manipulation and texturing triangles that make up the mesh.

1.2.2 Organization of this work

As indicated in Alg. 1.1, we cover the image-based methods and those for shape reconstruction in Chapters 4 and 5, respectively. These steps require quite different technologies. While during enriching, information from video frames, and, consequently image-processing methods will be used, the stage of post-processing presupposes application of shape reconstruction methods and color or intensity information will not be considered before texturing. The related work, preceding these sections will be grouped into an image-based Sec. 3.1 and a point-based Sec. 3.2, followed by a short Sec. 3.3, which describes several already existing reconstruction procedures. For reasons of completeness, Chapter 2 will show the most important concepts for point matching and shape reconstruction. The evaluation of the reconstruction algorithms will be demonstrated for several data sets in Chapter 6. Finally, conclusions and directions of future research are given in Chapter 7.

1.3 Main contributions

Several new ideas will be developed in this work.

1. Most state-of-the-art approaches do not consider points already reconstructed during Step 1 of the reconstruction pipeline in the course of computation of depth maps. However, these points can propagate the depth information to neighboring pixels; as a consequence, local triangular networks, also called *tessellations*, are used in this work. The starting point is usually the *Delaunay triangulation* of points in the images. These triangles do not always coincide, not even approximately, with the object surface. Therefore, we introduce novel ideas to evaluate the triangles as consistent and inconsistent with the surface, to try to correct the depth values of the inconsistent triangles using color information and to support the pixel costs to be low at the disparity values given by triangles consistent with the surface. A triangulation-based smoothness term will be the topic of Sec. 4.5 while the necessary theoretic background is provided in Sec. 4.1 and Sec. 4.3.

2. Applying non-local algorithms for multi-view configurations and not for stereo image pairs has become attractive only in the recent years. A relatively fast and easily-

implementable approach of semi-global optimization was first introduced by Hirschmüller in [67] for rectified image pairs. Few generalizations of this approach exist, like for example for the case of three cameras in a special trinocular configuration [62]. The principal innovation of our work, described in Sec. 4.5.3, is to apply this algorithm for an arbitrary number of not necessarily rectified images after a local approach, supported by triangular meshes, assigns a cost value to every pixel and every depth label. An important contribution concerns the automatic choice of smoothness parameters (Sec. 4.5.4).

3. Point clouds reconstructed by passive sensors with small, uncalibrated cameras often have rather dramatic negative properties of varying density, Gaussian noise and outliers (points far away from the surface, which can result, for example, from shadows, reflections and moving objects). A broad, detailed analysis of the performance of methods for shape reconstruction applied on these point clouds has, to our knowledge, not yet been carried out. It will thus be important to investigate how the state-of-the-art methods for shape reconstruction – being applied on the original and enriched point cloud – can cope with the negative properties mentioned above. Section 3.2, dedicated to already existing methods of surface reconstruction, is therefore covered with a higher level of detail. We will see that the L_1-splines-based procedure of Sec. 5.2, which represents the most important contribution of this work, provides the most accurate reconstruction. The high computing time of this procedure can be explained in part by some technical limitations of the current implementation and in part because computation of an L_1-spline requires solving a linear program. In Chapter 7, we will discuss how the computing time can be reduced.

Beside these three main contributions, we also care about

1. Fast and point projection equations that allow simultaneous processing of large point sets. A compact *closed-form representation* of depth and disparity values as well as 3D points is given in Sec. 4.1.

2. *Sparse tracking* with the search space for correspondences reduced to a line segment because we are given camera matrices and disparity ranges from the already available points. These points also provide initial values for two iterative algorithms, namely *epipolar* and *simultaneous* tracking, described in Sec. 4.4.2. The cost function and minimization procedure are then similar to the already existing methods of [94].

3. Binocular stereo reconstruction, since there is a large amount of software with different conceptional advantages available in the Internet. Since we must exploit the redundant information from many images, the algorithm of *median-depth maps* was developed and is described in Sec. 4.5.2.

4. Reducing and homogenizing the number of triangle vertices in the images by applying *restricted top-down quadtree triangulations* results in surfaces without cracks. This topic, described in Sec. 5.1.1, is an essential step to prepare the *shape reconstruction* on triangular grids, which have certain advantages compared with tensor-product surfaces considered in Sec. 3.2.4 and 5.2.1.

5. *Incremental reconstruction*, which ideally must be close to real time and which can be carried out without computationally challenging iterative or non-local methods. The evaluation of triangles is performed by a local method (LIFT, see [22] and Sec. 5.1.2) and can be incrementally updated.

1.4 Some notation

Besides elementary knowledge of linear algebra and numerical analysis, the reader of this dissertation is presumed to have basic knowledge of computer vision. For detailed clarification of terms *homography, fundamental matrix,* etc. we refer to the book due to Hartley and Zisserman, [61]. The most important parameters which can be found in more than one chapter of this work are included in the list below:

$\mathbf{x}, \mathbf{y}, \mathbf{p}, \mathbf{X}$	points
$\mathcal{X}$	point list
π	plane in space, given by a 1×4 vector
$\mathcal{I}$	images
P	camera matrices
d/j	depth value / discrete depth or disparity label
$\mathcal{D}$	depth or disparity map
T	triangle (a triple of integer numbers)
$\mathcal{U}, \mathcal{V}, \mathcal{W}$	local barycentric coordinates of the triangle
$\mathcal{T}$	triangular mesh
$\mathcal{F}$	surface
$\mathbf{n}$	normal vector
$\cdot_x, \cdot_y, \cdot_u \cdot_v$ etc.	partial derivatives $\partial \cdot / \partial x$ etc.
$c, E, \mathcal{J}$	cost, energy function, Jacobian (matrix)
$\|\cdot\|_p$	L_p vector norm $\|\mathbf{x}\|_p = (\sum_i \|x\|_i^p)^{1/p}$, $p = 2$ if nothing else is stated
dst	Euclidean distance function, $\mathrm{dst}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$
c^+/c^-	$\max(c, 0)/\max(-c, 0)$
$(\cdot)^T/(\cdot)^{-1}$	matrix transpose / inverse
$\mathbf{0}_w, I_w$	zero-vector of length w, $w \times w$ identity matrix
U	$U(a) = 1$ if a is true and 0 otherwise

Remarks: Frames of a video sequence taken at time k will be denoted by $\mathcal{I}_k$ and corresponding camera matrices by P_k.

Generally, we denote 2D and 3D points and vectors by bold variables $(\mathbf{x}, \mathbf{y}, \mathbf{X})$. Letters in lower case (e. g. $\mathbf{x}, \mathbf{y}$) will usually denote points/pixels in images; upper case is reserved – especially if ambiguous representations are possible – for 3D points. Also $\tilde{\cdot}$ stands for homogeneous coordinates and $\simeq$ denotes equality up-to-scale.

We will denote incidence relations with "$\in$". For example, $\mathbf{x} \in \mathcal{I}$ means that $\mathbf{x}$ lies within the rectangular domain of image $\mathcal{I}$ and $\mathbf{x} \in T$ means that $\mathbf{x}$ lies in the triangle. The constraint on barycentric coordinates of $\mathbf{x}$ is in this latter case $\mathcal{U} + \mathcal{V} + \mathcal{W} = 1$ and $\mathcal{U}, \mathcal{V}, \mathcal{W} > 0$. The inequalities in terms of x, y (coordinates of $\mathbf{x}$) from the height and width of $\mathcal{I}$ in the first, coordinates of vertices of T in the second case can be easily established.

Chapter 2

Theoretical background

This chapter summarizes the most important basics and tools for computer vision and shape reconstruction. Image pair rectification to epipolar geometry is an important tool to accelerate computations and also to make window-based matching algorithms invariant against rotation. Therefore we will consider this topic separately in Sec. 2.1. Then, two main ideas of matching – the photo-consistency terms (Sec. 2.2) and the smoothness assumptions (Sec. 2.3) – are presented. Finally, a short introduction to approximation of surfaces from triangular irregular networks (TINs) is given in Sec. 2.4.

2.1 Image rectification

Image rectification is an elegant way to perform a search for correspondences in one constant direction and thus computationally optimize matching algorithms. We will now briefly review implementation details, advantages, and disadvantages of binocular (Sec. 2.1.1) and trinocular rectification algorithms (Sec. 2.1.2).

2.1.1 Image pair rectification

Given a fundamental matrix F, searching for correspondences can take place along epipolar lines in the binocular case. For reasons of speed and in order to compensate for rotational deviations in the orientation of windows around corresponding pixels, rectification transformations are applied on images. All epipolar lines in the rectified images are parallel, for example, to the x-axis. The computation of the fundamental matrix for two cameras[1] P_1, P_2 is carried out according to:

$$F = (P_2 \cdot \check{\mathbf{C}}_1) \times (P_2 \cdot P_1^{\dagger}), \tag{2.1}$$

(see Eq. 9.1 in [61]) where $\mathbf{C}_1$ is the location of the first camera given by the one-dimensional null-space of the 3×4-matrix P_1 and $P_1^{\dagger}$ is pseudo-inverse of P_1. If the epipole is inside the image domain, one possibility for rectification is to extract epipolar lines directly and to orient them by means of polar coordinates (r, ϕ), where r is the distance to the epipole and ϕ is the inclination angle of an epipolar line (see [110]). Otherwise, one can find two homographies H_1^R and H_2^R that transform the epipole to the point at infinity $[1\ 0\ 0]^T$ and thus make epipolar lines lie horizontally in the images. There are nine degrees of freedom[2] which can regulate H_1^R and H_2^R in the way such that images look like original images after

[1] Throughout this work, *camera* will be an abbreviation for *camera matrix*. We use monocular image sequences in our data sets, so there will be no possibility for misinterpretations.

[2] The fundamental matrix has 7 degrees of freedom and each of two homographies has 8. Since the fundamental matrix must be fixed, we have $2 \cdot 8 - 7 = 9$ degrees of freedom.

transformation; in other words, projective and affine components of H_1^R, H_2^R are minimized. Such a pair of homographies can be obtained by some simple method (e. g. [110], p. 66) and then optimized using some meaningful criterion [96]. In this work, explicit minimization of projective and affine components of the transformed images was chosen and is carried out by the method of Loop and Zhang [90], which extracts first one parameter λ responsible for the projective transformation of images by means of a standard optimization problem. The cost function for this optimization uses the fact that a projective transformation minimizing image loss should be as close as possible to an affine one. After λ is extracted, the choice of other parameters is rather trivial.

We show the results of rectification by this method in Fig. 2.1 and also Fig. 4.7 (see p. 55)[3] and conclude that projective image distortion of the rectified images is rather small since image transformations are very similar to rotations.

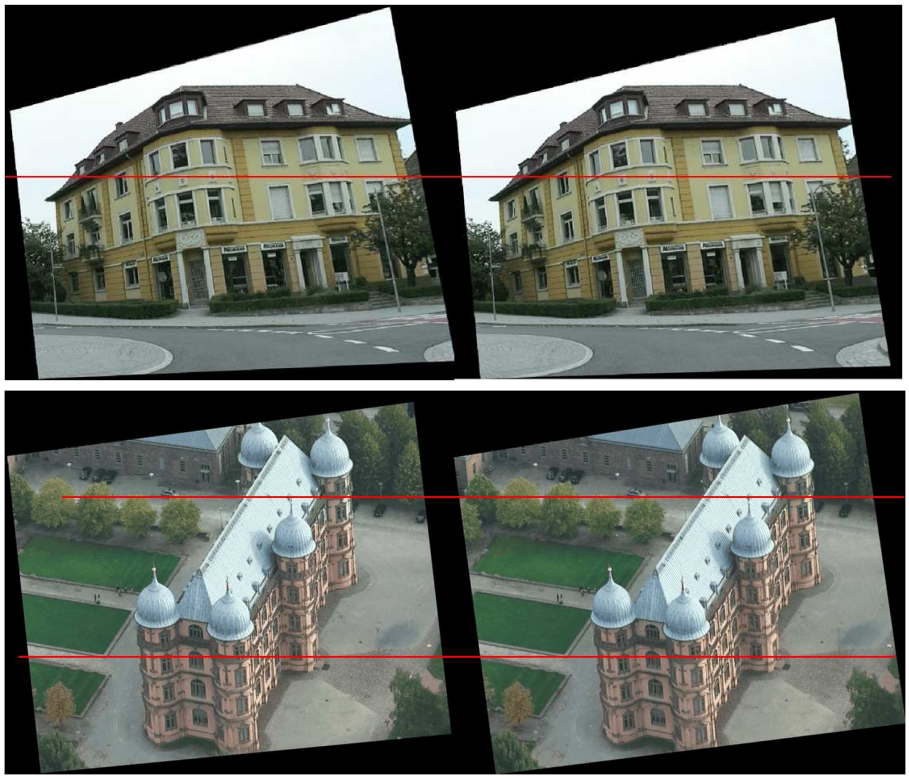

Figure 2.1: Top: Two frames from the sequence *House* rectified to epipolar geometry. Bottom: Two frames from the sequence *Gottesaue* rectified to epipolar geometry. Several horizontal epipolar lines are depicted in red. The parameters of rectifying homographies are chosen by means of [90] and as a result, the projective distortion of images is almost negligible.

2.1.2 Trinocular rectification

Since our sequences are not restricted to pairs of images, it is important to mention the existing ways to rectify also triplets of images. Given images $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$, there is a possibility to rectify the images in a way that $\mathcal{I}_1^R, \mathcal{I}_2^R$ are aligned horizontally, $\mathcal{I}_2^R, \mathcal{I}_3^R$ vertically and $\mathcal{I}_1^R, \mathcal{I}_3^R$ diagonally (i. e. for $(x_1, y_1) \in \mathcal{I}_1^R, (x_3, y_3) \in \mathcal{I}_3^R$, the relation $y_3 - y_1 = \lambda(x_3 - x_1)$ where

[3] a detailed description of data sets is given in Sec. 6.1.

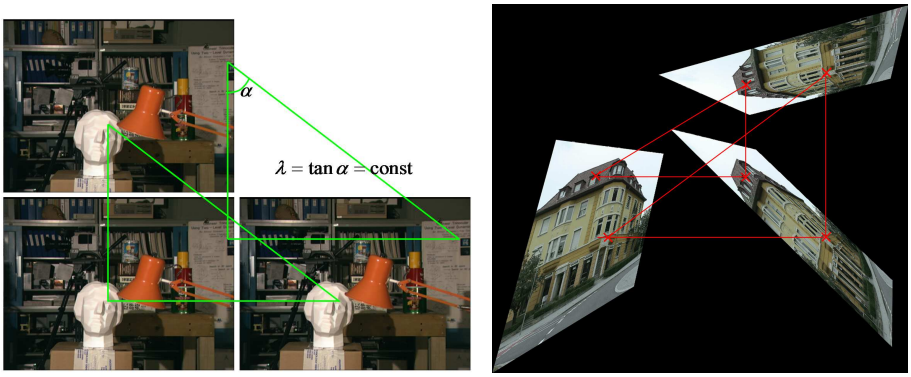

Figure 2.2: Left: Three images from the well-known benchmark data set *Tsukuba* [115] in a trinocular configuration; Right: for a general video stream taken from approximately the same altitude, trinocular rectification of images without significant distortion is hardly possible.

λ a scalar preferably ± 1 holds). The advantage of this kind of rectification is its robustness and elegance, since it can be performed linearly [137]. But it has one big disadvantage: It can be performed only for several special cases, for example, for the camera configuration of the kind of Fig. 2.2 right, mounted on a robot in [62]. For the general case, it is already difficult to fulfill two first conditions: Given that the epipole $\mathbf{e}_{12}$ is transformed to $[1\ \ 0\ \ 0]^T$, and, at the same time, $\mathbf{e}_{23}$ is transformed to $[0\ \ 1\ \ 0]^T$, then new line at infinity results from the straight line connecting prototypes of $\mathbf{e}_{21}$ and $\mathbf{e}_{23}$. But if this line intersects the image domain of $\mathcal{I}_2$ or just passes nearby, then there is no possibility of rectification without significant distortion (see Fig. 2.2, right). The problem of a straight line intersecting an image domain arises more often (at least, in our applications, where the images were taken from approximately the same height) than a single point lying inside it. For this reason, we will create sequences of rectified image pairs, as described in Sec. 4.1, instead of performing multi-image rectification for depth estimation.

2.2 Image-based methods – data cost functions

The basic task of 3D reconstruction is to obtain the spatial coordinates and color/intensity values of a point given its color/intensity values of pixels in the images. If we use the reference image $\mathcal{I}_0$ to color the 3D points, then, for another image $\mathcal{I}_k$ we are interested in a geometric transformation G_k and a radiometric transformation R_k such that

$$\mathcal{I}_0(\mathbf{x}) = R_k\left(I_k\left(G_k(\mathbf{x})\right)\right) + r(\mathbf{x}, k), \tag{2.2}$$

where the *residual term* $r(\mathbf{x}, k)$ is zero in the ideal case and can be supposed to be small for practical situations. The geometric transformation G_k depends on the camera model. For example, if the depth of the scene is negligible (see [121]), an (image-to-image) homography $\check{\mathbf{x}}_k = H_k\check{\mathbf{x}}$ can be used. For a classical pinhole camera, which stands in focus of our applications, the relations can be expressed in terms of depth for multi-view configurations (or, equivalently, disparity for binocular configurations). The essential goal of matching problematic is to select the unknown values of depth (or disparity) parameters to minimize r given a suitable radiometric relation R_k of color/intensity information between $\mathcal{I}_0(\mathbf{x})$ and $\mathcal{I}_k(\mathbf{x}_k)$, which are our *data-cost* values. Hence in this section, we will present several ideas for choosing R_k and we consider, for the sake of simplicity, only gray images. However, it is important to note that in the general case, $\mathcal{I}, r$ can be also vectors and R a multi-dimensional map.

There are many other different cost functions mentioned in [69] to which interested readers can refer, but here we only want to give a short overview about cost functions we work with in order to perform robust depth estimation from a video sequence.

2.2.1 L_p-based functions

The simplest assumption, namely $\mathcal{I}_0(\mathbf{x})$ and $\mathcal{I}_k(\mathbf{x}_k)$ are approximately the same, means that the cost function $c(\mathbf{x})$

$$c(\mathbf{x}) = \left\|\mathcal{I}_k(\omega(\mathbf{x}_k)) - \mathcal{I}_0(\omega(\mathbf{x}))\right\|_p, \quad \text{where} \;\; p \geq 1 \tag{2.3}$$

must be small. Here ω is a small correlation window ω around points of interest needed to cope with rounding errors. Note that with increasing value of p, more weight will be given to outliers in the correlation window, which can deteriorate results for pixels near occlusions or dead pixels in infrared images (pixels with constant luminance values, similar to salt-and-pepper-noise). These are clearly undesired effects and this is why usually $p = 1$ or $p = 2$ are used. The cost functions corresponding to $p = 1$ and $p = 2$ are *Sum of Absolute Differences* and *Sum of Squared Differences*, abbreviated by SAD and SSD, respectively. In order not to give too much importance to non-plausible changes of luminance, one can use truncated cost functions, therefore e. g. , for SAD, we will use

$$c(\mathbf{x}) = \left(\frac{1}{\varepsilon_{\max}}\right) \sum_{\mathbf{y}\in\omega(\mathbf{x})} \min\left(|\mathcal{I}_0(\mathbf{y}) - \mathcal{I}_k(\mathbf{y}_k)|, \varepsilon_{\max}\right) \tag{2.4}$$

instead of (2.3) in Chapter 4. Here $\varepsilon_{\max}$ is a real-valued scalar, and by division by $\varepsilon_{\max}$, the cost function is scaled between 0 and 1. This cost function is sampling-sensitive because for non-integer coordinates of $\mathbf{y}_k$, the value $\mathcal{I}_k(\mathbf{y}_k)$ depends on the rounding procedure, so efforts can be made to make (2.4) sampling-insensitive (see [13]).

2.2.2 Other parametric cost functions

Due to the different viewing angles of P_0 and P_k onto the object's surface, there are luminance gain $a > 0$ and offset b in the intensity of the both images, in other words:

$$\mathcal{I}_k(\mathbf{y}) = a\mathcal{I}_0(\mathbf{y}) + b. \tag{2.5}$$

This equation can be explained by considering the *Phong lighting model* (see [33], pp. 306-311) when the total intensity is expressed in terms of two summands[4]: *ambient term* $\mathcal{L}_a$ and *diffusion term* $\mathcal{L}_d$, which is proportional to the intensity of the reflected light emanating from the common source $\mathcal{L}_d$ as well as to the angle between the surface normal and the viewing direction. From the relations $\mathcal{I}_0(\mathbf{y}) = \mathcal{L}_a + b(\mathbf{y})\mathcal{L}_d, \mathcal{I}_k(\mathbf{y}) = \mathcal{L}_a + b_k(\mathbf{y})\mathcal{L}_d$, we obtain (2.5). In order to achieve invariance with respect to linear transformations without knowledge of a and b, one can apply the function of *Normalized Cross Correlation*, denoted also by (*Zero-mean*) NCC or (Z)NNC:

$$\begin{gathered}\tilde{c}(\mathbf{x}) = \frac{\sum_{\mathbf{y}\in\omega(\mathbf{x})} \left(\mathcal{I}_0(\mathbf{y}) - \bar{\mathcal{I}}_0(\mathbf{y})\right) \cdot \left(\mathcal{I}_k(\mathbf{y}_k) - \bar{\mathcal{I}}_k(\mathbf{y}_k)\right)}{\sqrt{\sum_{\mathbf{y}\in\omega(\mathbf{x})} \left(\mathcal{I}_0(\mathbf{y}) - \bar{\mathcal{I}}_0(\mathbf{y})\right)^2 \cdot \sum_{\mathbf{y}\in\omega(\mathbf{x})} \left(\mathcal{I}_k(\mathbf{y}_k) - \bar{\mathcal{I}}_k(\mathbf{y}_k)\right)^2}}, \\ c(\mathbf{x}) = \frac{1 - \tilde{c}(\mathbf{x})}{2}\end{gathered} \tag{2.6}$$

[4] We omit here the Non-Lambertian specular component.

Here $\bar{\cdot}$ is the averaging operator. In order to avoid calculation of square roots, $c(\mathbf{x})$ from (2.6) can be replaced by:

$$\frac{1-\tilde{c}(\mathbf{x})|\tilde{c}(\mathbf{x})|}{2}, \tag{2.7}$$

which is also scaled between 0 and 1. This kind of correlation is quite sensitive to outliers since a local Taylor series expansion around zero describes a quadratic polynomial.

2.2.3 Nonparametric cost functions

In the case of complex radiometric relationships, one can still use assumptions about intensity ordering of gray values or even formulate implicit functions of probabilities of assigning gray values (mutual information).

Intensity-ordering-based functions

If not the magnitude but rather the order of intensities in quadratic windows is of interest, *the Census filter* [136] around a pixel can be considered. It defines a logical vector variable where each entry corresponds to a certain pixel $\mathbf{y} \in \omega(\mathbf{x})$. This entry is true if and only if $\mathcal{I}_0(\mathbf{y}) < \mathcal{I}_k(\mathbf{y}_k)$. Thus, Census not only stores the intensity ordering, but also the spatial structure of the local neighborhood. The computation of dissimilarity can be measured by Hamming-distances. Using similar descriptor vectors around salient points in gradient space, like SIFT [92] or SURF [8], theoretically can be generalized for dense sets of points. These descriptors however do not contain a reliable information in the regions of weak texture and their computation requires a very high computational cost.

Mutual information

The key idea of *Mutual information* is to quantify the extent to which two random variables are dependent by computing the entropy of the joint probability distribution $\mathcal{H}_{1,2}$ and subtracting it from the sum $\mathcal{H}_1 + \mathcal{H}_2$ of entropies of single probability distributions (see [133] for further details). To do this, an assumption about correspondences must be made on a coarser level (initialization). If we know that $\mathbf{x} \in \mathcal{I}_0$ and $\mathbf{x}_k \in \mathcal{I}_k$ are corresponding points, we increase the probability $P(m,n)$ where $m = s(\mathcal{I}_0(\mathbf{x})), n = s(I_k(\mathbf{x}_k))$ and s is a discretization function, that is, a suitable number of intensity levels. For example, if two 16-bit images are given, it makes more sense to convert them to 8-bit and consider $m, n = \{0, 1, 2, ...255\}$ than computing probabilities for each $m, n = \{0, 1, 2, ...2^{16}-1)\}$. From $P(m,n)$, we compute

$$\begin{gathered} P_1(m) = \textstyle\sum_n P(m,n), P_2(m) = \sum_m P(m,n), \\ \mathcal{H}_1(m) = \log(\tilde{P}_1(m)), \mathcal{H}_2(n) = \log(\tilde{P}_2(m)), \mathcal{H}_{1,2} = \log \tilde{P}(m,n), \end{gathered}$$

where $\tilde{\cdot}$ is the (one- or two-dimensional) Gaussian smoothness function. The cost function given by Mutual Information (MI) is computed according to:

$$c(\mathbf{x}) = -MI(m,n) = \tilde{\mathcal{H}}_{1,2}(m,n) - \tilde{\mathcal{H}}_1(m) - \tilde{\mathcal{H}}_2(n), \tag{2.8}$$

$m = s(\mathcal{I}_0(\mathbf{x})), n = s(I_k(\mathbf{x}_k))$. The values of $MI(m,n)$ are scaled between 0 and 1 and stored in a square matrix, see Fig. 2.3. The pixel-wise accumulation of costs from (2.8) within a window can be performed as well, e. g. by averaging costs of entries. The question of initialization without image pyramids will be the topic of Sec. 4.5.1.

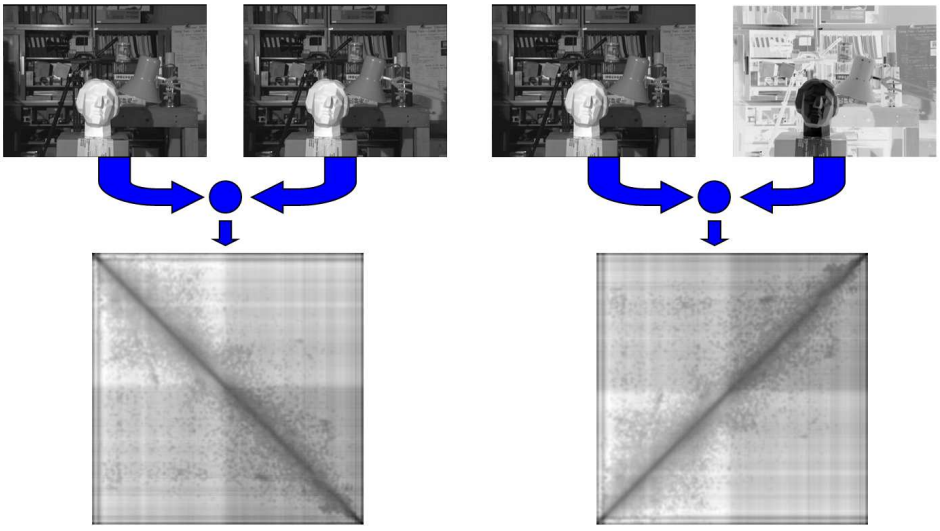

Figure 2.3: Mutual information MI as a cost function stored in a 256×256 square matrix (matching table). It can handle simple changes in illumination: in the pair of images on the top left, the lower cost entries mostly lie near the main diagonal of the matrix (bottom left). If we replace the second image by its negative (as in the pair of images on the right), the entries of the matching table change in the suitable way (bottom right). Fig. courtesy of P. Wernerus.

2.3 Image-based-methods – smoothness functions

Correct assignment of correspondences by minimizing one of the cost functions of the previous section can be carried out, in the majority of practical situations, only for a small number of points in textured areas. As we will see in Chapter 4, mismatches from local algorithms happen due to radiometric deviations, repetitive patterns of texture and weakly textured areas as well as many other factors. Since we want to obtain 3D coordinates for pixels homogeneously distributed in the image, we must make additional assumptions about scene geometry. In practice, surfaces observed are piecewise continuous, which means neighboring pixels usually have similar disparities. Belhumeur formulates in [10] the goal of matching as a Bayesian problem:

$$P(S|D) \simeq P(D|S)P(S), S \text{ denotes Scene}, D \text{ denotes Data.}$$

In other words, to maximize the probability of a scene given some data, not only data generated from the scene but also prior information about the scene have to be considered. Taking the logarithm of the last formula yields the well-known energy function

$$E = \sum_{\mathbf{x}} \left(E_{data}(\mathbf{x}, S) + E_{smooth}(\mathbf{x}, S) \right). \tag{2.9}$$

The most popular way to impose the smoothness penalty on the *disparity* or *depth*, denoted by d in this work, is to punish the disparity or depth jumps of neighboring points[5]. In other words,

$$E_{smooth}(\mathbf{x}, S) = E_{smooth}(\mathbf{x}, d_{\mathbf{x}}) = \sum_{\{\mathbf{x},\mathbf{y}\} \in N} f(d_{\mathbf{x}}, d_{\mathbf{y}}, \mathbf{x}, \mathbf{y}),$$

[5]From here on, d is the unknown we use in order to parametrize the Scene S. We leave this parameterization and also a discretization of depth values, which is usually imposed for dense reconstruction, until Chapter 4.

where $\{\mathbf{x}, \mathbf{y}\} \in N$ (or, alternatively, $\mathbf{y} \in N(\mathbf{x})$) if and only if $\|\mathbf{x}-\mathbf{y}\|_1 = 1$, $d_\mathbf{x}$ is the unknown parameter of depth at $\mathbf{x}$ and f is a scalar non-decreasing function of $\|d_\mathbf{x} - d_\mathbf{y}\|$. We give here several possible cost functions f some of which can be found in related works cited in Sec. 3.1.2.

$$f_1(d_\mathbf{x}, d_\mathbf{y}) = \lambda_1 U(d_\mathbf{x} \neq d_\mathbf{y}) = \begin{cases} 0 & \text{if } d_\mathbf{x} = d_\mathbf{y} \\ \lambda_1 & \text{otherwise} \end{cases} \tag{2.10}$$

$$f_2(d_\mathbf{x}, d_\mathbf{y}) = \begin{cases} 0 & \text{if } d_\mathbf{x} = d_\mathbf{y} \\ \lambda_1 & \text{if } 0 < |d_\mathbf{x} - d_\mathbf{y}| \leq d_0 \\ \lambda_2 & \text{otherwise} \end{cases} \tag{2.11}$$

$$f_3(d_\mathbf{x}, d_\mathbf{y}, \mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } d_\mathbf{x} = d_\mathbf{y} \\ \lambda_2 & \text{if } |\mathcal{I}_0(\mathbf{x}) - \mathcal{I}_0(\mathbf{y})| \leq g_0 \\ \lambda_1 & \text{otherwise} \end{cases} \tag{2.12}$$

$$f_4(d_\mathbf{x}, d_\mathbf{y}) = \lambda_1 |d_\mathbf{x} - d_\mathbf{y}|^{g_0} \tag{2.13}$$

$$f_5(d_\mathbf{x}, d_\mathbf{y}) = \lambda_1 \left(1 - \frac{d_0^2}{(d_\mathbf{x} - d_\mathbf{y})^2 + d_0^2}\right). \tag{2.14}$$

Here $\lambda_1 < \lambda_2, g_0, d_0$ are positive numbers called *smoothness parameters*, and numerous references can be found about optimal choice of smoothness parameters. See, for example, [28, 101, 79] (Sec. 3), [59] and references therein.

We review here the differences in expressions (2.10)-(2.14). In (2.13), the depth discontinuities are punished hard because the penalty function increases monotonically with the difference of depth values. As a result, the depth map is expected to be oversmoothed near occlusions. On the other hand, Eq. (2.10) punishes all discontinuities equally. Merely two cases of small and big differences of depth are considered in (2.11): for big differences it is a constant value. A smooth change between small cost for small differences and constant cost for big differences is modeled in (2.14). Finally, if two neighboring pixels have similar intensities, they are less likely to belong to different segments and so the disparity cost for such a pair of pixels should be larger, which justifies (2.12).

Now suppose that we have a path $\mathbf{v}$ and want to enable depth values of points to increase or decrease linearly along the path $\mathbf{v}$ instead of (possibly) incurring too many occlusions. This approach results in the next kind of smoothness term, which includes triplets of neighbors:

$$f_6(d_\mathbf{x}, d_{\mathbf{x}-\mathbf{v}}, d_{\mathbf{x}+\mathbf{v}}) = \lambda_1 |d_{\mathbf{x}-\mathbf{v}} + d_{\mathbf{x}-\mathbf{v}} - 2d_\mathbf{x}|. \tag{2.15}$$

It is also possible to combine (2.14) with one of penalty terms acting on neighboring pixels only, for example, f_1 of (2.10) or f_2 of (2.11).

Besides smoothness terms in the image space, we give an example of an object-based smoothness term from [79], see p. 63. The author uses the term *interaction*: pixels $\mathbf{x}$, $\mathbf{x}_k$ in two images of $\mathcal{I}$ and $\mathcal{I}_k$ can only interact when the reprojection rays from $\mathbf{x} \in \mathcal{I}, \mathbf{x}_k \in \mathcal{I}_k$ nearly intersect in space; the interaction $i = \langle \mathbf{x}, \mathbf{x}_k, d \rangle$ is set *active* if the intersection point is close to the object surface. Here d is a depth or disparity value, which, as we will see in Sec. 4.1, uniquely defines the 3D coordinate. For active interactions i, the boolean variable

$U(i)$ is set to true. Two interactions $i = \langle \mathbf{x}, \mathbf{x}_k, d_\mathbf{x} \rangle$ and $i' = \langle \mathbf{y}, \mathbf{y}_k, d_\mathbf{y} \rangle$ are neighbors ($\{i, i'\} \in N_2$) if and only if $d_\mathbf{x} = d_\mathbf{y}$ and $\|\mathbf{x} - \mathbf{y}\|_1 \leq 1$. The object-based smoothness term

$$E_{smooth} = \sum_{\{i,i'\} \in N_2} \lambda(i, i') U(U(i) \neq U(i')) \tag{2.16}$$

with a scalar function λ and U as in Sec. 1.4 is not quite the same as one of the single-image-based disparity terms (2.10)-(2.15).

2.4 Shape reconstruction

We now consider the shape reconstruction portion of the reconstruction pipeline. The task it to perform polygonization of an input point cloud which means either compression of very dense point clouds (as a result of substep 2.2 of Alg. 1.1, if it took place) and/or interpolation of point clouds with moderate density (if that step was omitted). It is clear that not every surface can be exactly modeled by triangles. Therefore we assume a surface $\mathcal{F}$ interpolating or approximating such a point cloud $\mathcal{X}$, and our task will be to find a polygonization homeomorphic[6] to $\mathcal{F}$. The necessary theoretical background about surface polygonization without explicit computation of $\mathcal{F}$ will be given in Sec. 2.4.1 while several possible ways of meshing of surfaces will be given in Sec. 2.4.2. Note that an elaborate survey of previous work on surface computation will be given in Sec. 3.2.

2.4.1 Direct polygonization of point clouds

Given a set of 2D points in a plane, there are plenty of ways to connect (some of) them by means of straight line segments. However, depending on the configuration, one way may appear more compact or more natural from a physical point of view than another one. As an example, all four options for connecting points in left hand side portion of Fig. 3.1, p. 34 are possible and have geometric justification (as we will see below), but the first one – which does the best job of recognizing that the shape consists of two rings – seems somehow more probable; intuitively, its probability will increase with the point density within two rings. In 3D, the situation is clearly even more complicated. If we imagine a surface $\mathcal{F}$ passing through the 3D point cloud $\mathcal{X}$ and wish to generate a triangular mesh $\mathcal{T}$ homeomorphic to $\mathcal{F}$, it becomes clear that the point sets must have special properties with respect to their density (a term to be explained below) and noise: their density must exceed a given threshold and noise level must be low. Amenta and Bern [4] give a sufficient criteria for sampling in order to make a triangular surface homeomorphic to the original one.

Here the definitions of *medial axis* (points in space which have at least two nearest neighbors on $\mathcal{F}$ in the Euclidean sense), *local feature size* (distance from point to medial axis, denoted as *lfs*) as well as ρ-*sample* $\mathcal{X}$ such that $\mathrm{dst}(\mathbf{r}, \mathcal{X}) < \mathit{lfs}(\mathbf{r})\rho$ for each $\mathbf{r} \in \mathcal{F}$) are given. The main result, stated in [4], considers noise-free ρ-samples, $\rho \leq 0.1$. Then it is possible to reconstruct the triangular mesh homeomorphic to $\mathcal{F}$. Note that a ρ-sample does not require the point density to be uniformly constant. From the definition of the medial axis, it must only be high enough in curved regions.

The approaches related to that in [4] have an advantage that they do not require explicit knowledge of $\mathcal{F}$ for computation of such a triangular irregular network (TIN) $\mathcal{T}$. This makes them very attractive for several openly and commercially available software packages such as *meshlab*. Therefore it will be worth reviewing these methods in Sec. 3.2.1. However, the main drawback of TINs is their extreme dependence on the sampling density of points.

[6] Two surfaces $\mathcal{F}, \mathcal{F}'$, are said to be *homeomorphic* if there is a mapping (*homomorphism*) $f : \mathcal{F} \leftarrow \mathcal{F}'$. Here f must be a continuous bijection, f^{-1} is also continuous.

Apart from the fact that there are only heuristic methods to estimate ρ without knowledge about the surface, it is rather impossible, because of difficulties of image-based algorithms to find correspondences in homogeneously textured areas or in the areas not sufficiently covered by the camera path, to satisfy the assumptions of [4]. In addition, the resulting mesh $\mathcal{T}$ will usually contain aesthetically unpleasant surface artifacts which have to do with noise and outliers in the data, since no explicit assumption about the smoothness of the surfaces underlying $\mathcal{T}$. Since we want to obtain polygonal meshes despite these negative properties and also be able to fill sparsely sampled regions in a plausible way, it will be necessary to deduce methods that lack, to a certain extent, a theoretical justification, but are good enough to be applied in the practical case. For this practical case, we may make use of assumptions for objects we are dealing with, such as orientation consistence, or one dominant direction which is given by the z-axis.

2.4.2 Polygonization of surfaces

Generation of meshes

Suppose that the function describing $\mathcal{F}$ is explicitly given. In the case of 2.5D "terrain skins", altitudes z are represented in terms of x and y coordinates as a function $z = f(x, y)$. Otherwise, there is a 3D parameterization $\mathbf{X}(u, v) := (x(u, v), y(u, v), z(u, v))$ in some coordinate system (u, v). In both cases, one can perform (e. g. Delaunay) triangulation of (x, y)-, respectively (u, v)-points.

Other methods have an implicit surface as input. It is usually given by a signed distance function sampled for points in space. Since sampling implicit surfaces goes beyond the scope of this work, we mention the most famous algorithms [39, 53, 66, 91, 107] and refer to (e. g.) Akkouche and Gallin [3] where a classification of these methods in three groups (surface meshing techniques, surface fitting techniques and surface tracking techniques) is made and also to [17] where several interesting refinements and more references of the existing methods are described. Our default method for implicit surface polygonization will be the well-known algorithm of *marching cubes* [91].

Mesh manipulation

Some kinds of surface tessellation routines described in the last paragraph often do not consider the (scalar or vector) properties of mesh vertices, as for example, the partial derivative values, color informations etc. A concept and examples of cost functions which can be minimized with local flipping algorithms are given in [41] for 2.5D surfaces. Usually, a combination of several basic procedures are chosen for mesh simplification, namely:

1. *vertex translation:* Vertices are transformed so that a total energy of the mesh is diminished. See Fig. 2.4, top.

2. *edge flip:* A spatial quadrilateral **ABCD** consisting of two triangles **ABC** and **ACD** is flipped to **BDC** and **BDA**. See Fig. 5.3, p. 77, right.

3. *edge collapse:* Two vertices are melt, that is, the edge between them disappears, the number of triangles is reduced by two and that of edges by three, as shown in Fig. 2.4, bottom.

4. *edge split:* A new vertex is added near an edge. If this is not a margin edge, then the new vertex is connected to other two vertices of the quadrilateral and so the number of triangles is increased by two.

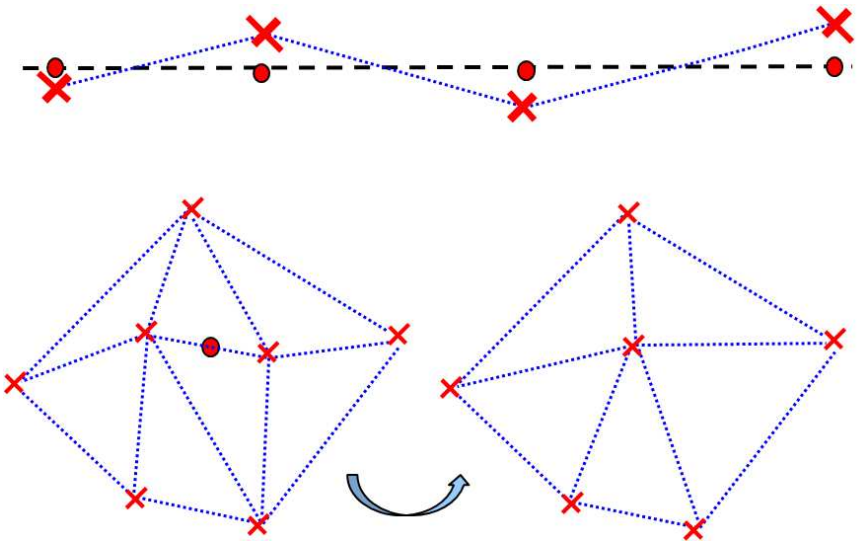

Figure 2.4: Top: To reduce an energy term (e. g. Laplacian), a dominant plane can be fitted. Bottom: To compress the mesh, an edge collapsing method is applied (inserting a new point marked by a red circle). Edge split and edge collapse are inverse procedures.

In [71], items 1-3 of those previously mentioned are selected in random order to perform mesh simplification. Other authors restrict themselves to one operation – edge flip in [103] or edge collapse in [89]. Some publicly or commercially available software packages are mentioned in [126].

While the four procedures mentioned above do not change the topology of the mesh, the procedure of *hole filling* usually has a topologically different mesh as output. A hole as a loop of boundary edges (i. e., those incident with exactly one triangle) has to be identified and filled with new vertices and edges. One algorithm to perform hole-filling is described in [134], the non-trivial part of the algorithm consists in reasonable choice of 3D coordinates for new vertices to be added to the mesh.

Chapter 3

Previous work

Tremendous amounts of work on scene reconstruction from video sequences have been done in the past decades. Even though it is hardly possible to survey the technical details for all existing algorithms, a detailed study of state-of-the-art is very important for us not only because an evaluation of our algorithms and comparable methods will be described in Chapter 6, but also in order to demonstrate that the innovations presented in this work are meaningful and robust to close the gap in the area of generic urban terrain reconstruction from aerial videos, often under non-cooperative conditions. Since our work consists of an image-based and an object-based module, we separately cover algorithms for depth estimation from a set of images and surface reconstruction algorithms in Sec. 3.1 and Sec. 3.2. Among numerous already existing pipelines that go the whole way from an image sequence to a textured reconstruction, we give in Sec. 3.3 a detailed description of three procedures [116, 48, 111] which turned out to be very instructive for our approach.

3.1 Previous work on depth map computation

The task of retrieving depth values for a relatively dense and homogeneously distributed set of pixels in the reference image can be accomplished by tracking sparse points as in Sec. 3.1.1 or by using data, smoothness and other assumptions, as mentioned in Sec. 3.1.2, 3.1.3, and 3.1.4, respectively.

3.1.1 Sparse tracking

We forget for a short moment the 3D aspect of the problem and solely wish to retrieve, for a pixel $\mathbf{x} \in \mathcal{I}_0$, the corresponding point $\mathbf{x}_k = \mathbf{x} + \mathbf{w}_k \in \mathcal{I}_k$. This kind of matching is closely related to the *optical flow* problem because in the approaches of e. g. [72, 94], a functional including a data and a smoothness term must be minimized over the translation parameters $\mathbf{w}_k$ by means of common numerical methods. For example, in [21], the data cost consists of a non-decreasing function Ψ of weighted differences of gray values and their Laplacians:

$$\Psi = \Psi\left(|\mathcal{I}_0(\mathbf{x}) - \mathcal{I}_k(\mathbf{x}+\mathbf{w}_k)| + \gamma|\nabla\mathcal{I}_0(\mathbf{x}) - \nabla\mathcal{I}_k(\mathbf{x}+\mathbf{w}_k)|\right), \gamma \in \mathbb{R} \tag{3.1}$$

and the smoothness term is the total variation of the flow field, which is given, in the case of two images, by the norm of spatial-temporal second-order derivatives. In order not to get stuck in local minima, image pyramids downscaled by an arbitrary factor between 0 and 1 are calculated and a steady-state solution of a linearized fixed-point-approximation of (3.1) determined for each pyramid level is used as the initial value for the next level.

The process of optical flow estimation can be generalized for a multi-view case [113]. Unfortunately, the computational cost is very high and so discretization of derivatives and using fixed-point numbers are necessary to perform minimization in a reasonable time. It is theoretically possible to detect moving objects by means of optical flow algorithms since the functionals do not prevent any point from being moved to any other point. For retrieving 3D structure, however, it will be indispensable to introduce geometric constraints and thus to reduce the search range for point correspondences to a one-dimensional space, namely the depth, which reduces the search space in the images to the (epipolar) line. Still, it is possible to use the features of the optical flow estimation pipeline for a sparse set of points, which later can allow either direct surface reconstruction or 2D meshing of points into triangles and classification of these triangles into consistent and non-consistent with the surface by considering pixels within these triangles.

The state-of-the-art method for computing correspondences for a sparse point set is the well-known algorithm of Lucas-Kanade-Tomasi (KLT, [94]) which iteratively searches for a (e. g. affine) transformation of a window around a point in the first image that produces a similar window in the second image. Usually the similarity is measured by the squared norm of the differences of the intensities within both windows; the optimization method can be gradient descent. The algorithm has one important advantage – no need for any prior information; hence a simple creation of image pyramids and the identity transformation as a starting value is usually a suitable approximation for the position of pixels in the next image. But it is also its disadvantage because the search range for point correspondences is theoretically unlimited. For this reason, efforts were made to incorporate the known camera positions. Trummer et. al. [130] consider the binocular case and support tracking of points along epipolar lines. The component perpendicular to the epipolar lines is supposed to compensate for uncertainties of camera poses. The algorithm is expected to perform worse for points that lie near edges parallel to epipolar lines. In order to make this approach more stable with respect to this problem, one can consider the work of Gruen [54, 55] as a generalization of this approach in the case of a multi-view system. In the system described in Eqs. 9-11 in [55], an affine transformation of points in images is supposed to compensate for rotations, so instead of considering relative orientation of cameras, he uses an over-parametrized system of equations for every point (six affine transformation parameters per camera and three spatial coordinates). In [54], an additional variable expressing radiometric deviation is introduced. A statistical test in order to eliminate unnecessary parameters from further calculations is performed afterwards. Note also that no use of information from already established correspondences is made in these approaches.

3.1.2 Considering the data term

Many existing approaches of stereo matching are mentioned in the survey of Scharstein and Szeliski [115]. Local methods compute depth maps pixel-by-pixel using the principle "winner takes all". For a pixel $\mathbf{x} = (x, y)$, values of a cost function (denoted by $c = c(\mathbf{x}) = c(\mathbf{x}, d)$) are obtained for candidates in a suitable rectangle

$$[x + d_{\min} - \varepsilon_x; x + d_{\max} + \varepsilon_x] \times [y - \varepsilon_y; y + \varepsilon_y],$$

where $\varepsilon_x, \varepsilon_y$ are needed to take into account uncertainties in the camera parameters and $d_{\min}$, $d_{\max}$ are the disparity ranges computed, for example, from already available points. The cost function can be SSD of gray values differences, NCC or some other distance function of Sec. 2.2. The point with the highest score is chosen to be the corresponding point if it satisfies some heuristics (for example, the value of the score must exceed a certain threshold). We can mention contributions due to [69] where disparities that failed the cross-check test (see Eq. (3.2) below) are marked as discarded and then filled by values propagated from

neighboring points, [18] where the window size for correlation was adapted according to the local geometric constellation (pixels with disparity similar to $d_{\mathbf{x}}$, i. e., lying near the fronto-parallel plane through $\mathbf{x}$, obtain larger weights in e. g. (2.3)) and [114] where a diffusion term was introduced.

Of course, these methods produce a large set of outliers among point correspondences in the regions of repeatable texture and homogeneously textured regions. This happens because no model assumptions about the surface are made and so not all available information is used. In order to extract only reliable, confident pixels, [112] suggests discarding ambiguous matches by selecting the *maximum stable component* along an epipolar line. This largest stable subset is proved to be unique, but – especially in areas of homogeneous texture, – it can be very sparse and even empty.

3.1.3 Considering the smoothness term

Since we want to retrieve a reliable set of correspondences homogeneously distributed in the images, we strive for an efficient minimization of (2.9). To reduce computing time, depth or disparity scales must be discretized into labels. For example, we assign for every integer disparity value (in pixels) one of $S+1$ values $j = 0, ..., S$. Even with this discretization, global minimization of (2.9) was shown to be an NP-hard problem [19, 51], which means that the order of magnitude of operations needed for computing an exact minimum *cannot* be less time-consuming than the brute-force procedure of $O(S^M)$ configurations, where M is the number of pixels in the images. We will sketch and discuss several methods of different complexity that allow determining a strong local minimum of (2.9).

Dynamic Programming, tree-based optimization

The method presented in [10] suggests minimizing the energy functional along all epipolar lines using a well known method of *dynamic programming*. We will use this method for multi-view optimization and, from a detailed description of this method in Alg. 8.2 of the Appendix, we will see that the complexity can be reduced to $O(MS)$ where M is the number of pixels in the images and S is the number of depth/disparity values. However, the distribution of costs in the adjacent epipolar lines can be completely different which usually leads to implausible bulges and convexities in the final result. We do not discuss here heuristics for additional optimization in the direction perpendicular to epipolar lines, but turn our attention to a generalization of this method given in [132] which uses a *minimum spanning tree* [82] from the weighted graph of absolute gray value differences of the neighboring pixels instead of (epipolar) lines. Since by including an edge between neighboring pixels $\mathbf{x}$ and $\mathbf{y}$ in the tree, one enforces the constraint that pixels $\mathbf{x}$ and $\mathbf{y}$ should have similar disparities, it is reasonable to weight the edge of the graph by $|\mathcal{I}_0(\mathbf{x}) - \mathcal{I}_0(\mathbf{y})|$ and then to create a minimum spanning tree of such a graph.

The algorithm starts at the leaves of the tree (as in [10], it starts in the first pixel of the epipolar line) and processes along the branches of the tree until the root is achieved. From the root, it is then possible to go to every leaf since the recursive information, which is the best disparity value of the *current* pixel (i. e. *child*) given a disparity value for the *previous* pixel (i. e. *parent*), is available; compare Alg. 8.2, p. 142. The algorithm has the property of being invariant with respect to image subdivision (since the minimum spanning tree of a union of disjoint sub-images is a union of minimal spanning trees of these sub-images [82]), which offers an elegant way to compute depth/disparity maps even from large images. However, also here bulges that correspond to the branches of the tree are inevitable in the final result.

2D Global approaches

As mentioned before, the process of finding a 2D global minimum of equation (2.9) is, unfortunately, a NP-hard problem, in contrast to both of the methods mentioned above, which obtain a *global* minimum of the 1D equivalent of equation (2.9). The algorithms of *alpha-expansion* [80] and *alpha-beta-swap* [19] based on *graph cuts* and *belief propagation* [77, 124] approximate this minimum by iterative procedures.

For example, given a depth map $\mathcal{D}$, an alpha-expansion (α-expansion) of $\mathcal{D}$, as described in Kolmogorov and Zabih [80], is a configuration $\mathcal{D}'$ with $\mathcal{D}'(\mathbf{x}) = \mathcal{D}(\mathbf{x})$ or $\mathcal{D}'(\mathbf{x}) = \alpha$. Now one can define a binary function f such as $f(\mathbf{x})$ is true if $\mathcal{D}'(\mathbf{x}) = \mathcal{D}(\mathbf{x})$ and false otherwise. It is possible to construct a graph that minimizes in a polynomial time the energy function for binary variables:

$$E(f(\mathbf{x}_1), f(\mathbf{x}_2)...f(\mathbf{x}_n)) = \sum_{i,j} E(f(\mathbf{x}_i), f(\mathbf{x}_j))$$

if and only if $E(0,0) + E(1,1) \leq E(0,1) + E(1,0)$. The procedure of construction and minimization of the binary graph is given in [81].

Now disparities from $d_{\min}$ to $d_{\max}$ are randomly ordered. The inner iteration consists of selecting a disparity j from the list and minimizing energy over all j-expansions of D via graph cuts. The outer iteration consists of repeating the inner iteration until no improvement in the value of energy function has been achieved.

Especially for Nadir flights, the graph-cuts approach turns out to be one of the best methods for removing noise without over-smoothing the edges. However, its main disadvantage is an extremely long computing time. Another drawback is that the method has problems in scenes with many slanted surfaces.

Semi-global approaches

Another procedure for minimization of (2.9) is the method of Hirschmüller [67], originally elaborated for disparity map computation from a stereo pair. Here paths from different directions leading into one pixel are accumulated. For only one path, the method becomes equivalent to the dynamic programming. The key idea of algorithm is here, similar to [10], to use the *previous* pixel $\mathbf{x} - \mathbf{r}$ in order to compute the disparity value for the *current* pixel $\mathbf{x}$. The difference is that the global value of the cost function is stored in a $M \times S$ array obtained by summing up costs of all paths of the same disparity and then the disparity which yields the lowest result is chosen.

The original approach of [67] consists of computing image pyramids, then to start using a random map and iteratively calculate improved maps, which are used for a new cost calculation by means of Mutual Information (see Sec. 2.2.3). Finally, images and corresponding disparity estimations are iteratively upscaled until the original scale is achieved. Since the final result usually looks too noisy because of discretization into a finite number of paths, the author suggests using a median filter to obtain the final result.

To find occlusions and mismatches (in the reference image $\mathcal{I}_1$), one first computes disparity map $\mathcal{D}_{12}$ from $\mathcal{I}_1$ to $\mathcal{I}_2$, then $\mathcal{D}_{21}$ from $\mathcal{I}_2$ to $\mathcal{I}_1$, after which pixels $\mathbf{x}$ with the property

$$\left|\mathcal{D}_{12}(\mathbf{x}) + \mathcal{D}_{21}\left(\mathbf{x} + [\mathcal{D}_{12}(\mathbf{x}),\ 0]^T\right)\right| > 1 \tag{3.2}$$

are marked as occluded. We will take a closer look at the implementation details for the *multi-view case* in Sec. 4.5.3 and we will see that it is also here possible to perform semi-global optimization in a linear time.

The semi-global approach has another advantage in comparison with the the graph-based algorithm, apart from computing time. In its original implementation, graph cuts approach assigns to pixels in the regions of homogeneous texture depth values from neighboring textured pixels and propagates these values, which leads to spurious disparities in whole regions. However, the semi-global approach solves this problem by considering different patches and thus smooths the final result, as we will see in Chapter 6.

In the last paragraph of this subsection, we mention other modifications of the semi-global matching. In the method due to [15], another sophisticated path choice was given and the authors of [62] generalized the semi-global method for the rectified configuration of three cameras. Finally, in [68], particular attention was paid to homogeneous segments. Mean-shift segmentation of the reference image was performed and included in the semi-global matching pipeline, with an assumption that homogeneous segments must have approximately the same disparity.

3.1.4 Other approaches

To end this section, several other methods for depth or disparity map computation will be listed here, especially those that use a set of more than two images and use already available sets of points. Many authors perform *image segmentation* in order to improve reconstruction in textureless areas [7, 14, 68, 77, 87]. For example [14], after performing color segmentation of one image of a rectified stereo pair and computing disparity from some reliable points, the authors store the three degrees of freedom of the homography induced by a scene plane for every segment in a vector $\mathbf{v}$. The disadvantage is that, in general, $\mathbf{v}$ does not have geometric meaning and depends, as we will see in Sec. 4.3.1, only on the way the images are rectified. For this reason, the authors state that the weak point of the algorithm lies in the grouping planar segments into layers by computing Euclidean distances of corresponding values of $\mathbf{v}$. Besides this nontrivial task of assigning planes to segments and typical artifacts arising from over- and under-segmentation, color segmentation is not possible for infrared images, which are actually very important in our applications. Furthermore, Szeliski and Coughlan [127] extracted depth maps by means of splines. In [105], the Delaunay triangulation[1] of points already determined is obtained; [103] proposes using edge-flip algorithms in order to obtain a better triangulation since the edges of the Delaunay-triangles in the images are not likely to correspond to the object edges, but the point correspondences obtained at that stage are usually too sparse.

Using more than two images usually does not allow joint image rectification; nevertheless it is possible to use depth instead of disparity values. Multi-view systems are known to be more robust against occlusions and patterns of repeatable texture because using redundant information from more than two images allows suppressing spurious local maxima of the cost function. One survey about handling occlusions in stereo- and multi-view systems can be found in [74]. A global graph-cuts-based algorithm for multi-view depth map extraction [80] makes use of an additional term that marks occlusions and takes on the value infinity for forbidden configurations. The work of Mayer and Ton [98] is a simplification of the reconstruction pipeline of Schlüter ([116], see Sec. 3.3.1). A coarse 2.5D triangular mesh of points in a reference image is given and pixels inside the convex hull are projected into other images in order to obtain the local minimum of the cost function and thus the correspondences. This approach has turned out to be rather unstable for more than three images.

In the work of [86], which makes up the *Google 3D* software, high-resolution images with enough overlap are used and depth maps are computed by means of [77]. This method is

[1] There can be several Delaunay triangulations for degenerate sets of points, however, we can always imagine a slightly transformed point set and so, for a general case, there is only one Delaunay triangulation.

known to perform well for many fronto-parallel surfaces. Model assumptions are then used to perform tessellation.

The well-known software described in [50] is a continuation of the Microsoft-based software *Photosynth*. The main goal is to obtain dense reconstructions from arbitrary images taken mostly by tourists from historic buildings and available in the Internet. Even more than the depth map computation itself, the authors are concerned about criteria for the choice of local neighbors of the reference image from which the depth map is computed. These are: *global criteria* such as the number of common (SIFT, [92]) features, angles between reprojection rays from these features and differences of the resolution, as well as (after rescaling images according to the resolution changes) *local criteria*, which include the changes of the y-coordinates in the camera positions (in order to stabilize depth computation near horizontal lines) and the matching scores of the local features with the ZNCC-matching function (2.6). The (non-zero mean) NCC is the cost function for the region-growth-based approach for depth maps computation, but an important feature here is that the color shift component (denoted luminance in Sec. 2.2.2, Eq. (2.5)) is forced to be the same for each image pair and hence is included in the optimization. The output of the procedure is a 3D point cloud. For our applications, a conclusion can be made that matching SIFT points cannot provide the desired resolution for spatial depth (because subpixel accuracy of matching is not given) and therefore tracking algorithms provide better subpixel coordinates for the characteristic points.

3.2 Previous work on shape reconstruction

Because of rapid progress in hardware development that allows processing large point sets, there are plenty of algorithms for generating models from scattered point sets. The goal of this section is to provide an overview of several surface reconstruction algorithms and to discuss their potential advantages and disadvantages for application on our point clouds. We will consider in Secs. 3.2.1, 3.2.2, 3.2.3, 3.2.4, respectively, examples of four main approaches of geometric reconstruction, namely, TINs (examples stemming from the general idea of Sec. 2.4.1), (implicit) iso-surface extraction, surface reconstruction by level sets, and surface reconstruction by explicit functions (tensor-product splines). Sec. 3.2.5 is dedicated to several alternative algorithms for surface reconstruction.

3.2.1 Polygonization of surfaces of unknown topological type by TINs

Motivated by the approach of Amenta and Bern, many approaches are based on the local sample density. One of the typical examples presented in Gopi [52] requires that the dot product between the normals of neighboring points must be approximately constant and bounded away from zero. Then a local (2D) Delaunay triangulation of every sample point in its local coordinate frame replaces the 3D Voronoi polygonization of [4]. Medeiros et. al. [100] even compress the point set (by fusing neighboring points into clusters) and apply also a local algorithm for triangulation. The method of Boissonnat [16] starts with a Delaunay tetrahedrization of 3D points, and deletes iteratively all tetrahedra which either have one border face and the vertex non-incident with this face as an interior point, or two border faces and one interior edge. Other criteria (as in our case, visibility criteria for the given camera locations and corresponding depth images) can be applied, too. Another method, called *ball pivoting algorithm* it proposed by Bernardini et. al. [12]. It starts with a ball around a fixed edge in the point set. Its radius is diminished until the next point is hit. The triangle formed by this triple of points is added to the list and the procedure is propagated from these recently added edges. Finally, α-shapes [43], a geometric tool widely used and

investigated for surface modeling, consist of all triples of points such that no further point of $\mathcal{X}$ lies in one of two spheres of radius α around these triples of points. Clearly, for large values of α, the convex hull of $\mathcal{X}$ will be obtained while for too small values of α, the resulting set of triangles will be empty (see Fig. 3.1, left, for visualization of these situations). To name some advantages of α-shapes, we mention that the size of the triangles is automatically regularized, α-shapes are easy to generalize for higher dimensions, and, since they are a subset of the Delaunay triangulation (or, in 3D, tetrahedrization) of $\mathcal{X}$, they are in principal easy to compute.

The concept of α-shapes can be generalized to the case when information about distribution and quality of points is available. Here, weighted α-shapes [44] can be used. The point $\mathbf{X}$ is given a weight ($w_{\mathbf{X}}$) such that the weighted distance between two points $\mathbf{X}, \mathbf{Y}$ is given by $\tilde{d}(\mathbf{X}, \mathbf{Y}) = \mathrm{dst}(\mathbf{X}, \mathbf{Y}) - w_{\mathbf{X}} - w_{\mathbf{Y}}$. Just as α-shapes are subsets of the Delaunay triangulation (tetrahedrization) of $\mathcal{X}$, weighted α-shapes are subsets of the so called regular simplicial complexes, which can be extracted in a manner similar to the way in which the Delaunay triangulation of $\mathcal{X}$ is generated.

Despite the advantages of α-shapes and other TINs-based methods, the reconstruction results produced by them suffer from the drawbacks mentioned at the end of Sec. 2.4.1. Even though e. g. [11] gives a necessary condition when a triangular mesh modeled by α-shape is homeomorphic to $\mathcal{F}$, in many practical cases, the surface is not topologically correct. For example, it is not guaranteed that an edge is shared by exactly two triangles. If α is too small, the resulting mesh will contain holes. If α is too large, it will connect points of topologically different fragments. Furthermore, noise around nearly planar regions in $\mathcal{X}$ will result in visually unpleasant artifacts.

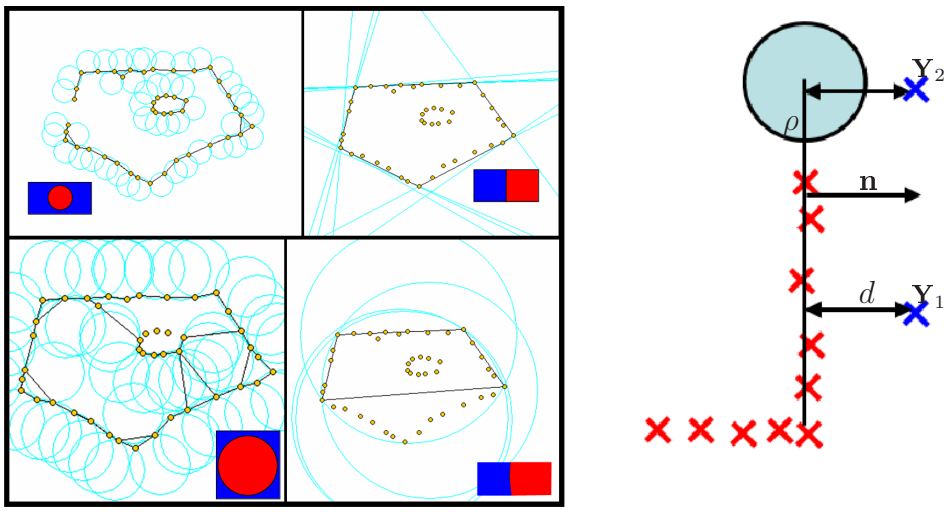

Figure 3.1: Left: Alpha-shapes (depicted by black line segments) for different values of α. The characteristic circles around segments that belong to the α-shape are depicted in cyan and their size is indicated by red circles on a blue background in the lower right of each portion. Right: Iso-surface extraction by [70]. Surface points $\mathbf{X}$ are depicted by red crosses and the nodes $\mathbf{Y}$ of the volumetric grid by blue crosses. The value d of the signed distance function is given by the length of the perpendicular from $\mathbf{Y}$ in the direction of "its" tangential plane (black horizontal line) if there is a sample point near the base-point, as in the case of $\mathbf{Y}_1$. Otherwise, as for $\mathbf{Y}_2$, it remains undefined. Problems are expected in the areas near sudden changes of normal vector field, see Fig. 6.33, p. 113.

3.2.2 Iso-surface extraction

An iso-surface is a surface in space that represents points of a constant value of a trivariate function $f(x, y, z)$. For the both state-of-the-art methods covered in this section, f represents the signed distance from the point to the surface and it is computed at the vertices of a tensor-product volumetric grid $(x_k, y_l, z_m), k = 0, ..., g_x, l = 0, ..., g_y$ and $m = 0, ..., g_z$ and g_x, g_y, g_z are the numbers of nodes in the grid, usually chosen in the way to guarantee approximately equal resolution of the grid in x, y and z direction. After extracting f, one performs meshing by means of one of the method mentioned in Sec. 2.4.2.

Hoppe's method

The method of Hoppe et. al. [70] is able to reconstruct a *smooth, orientable* surface of arbitrary topological type and consists of four steps (for schematic visualization, see Fig. 3.1, right). In the first step, the approximate computation of a surface *tangent plane center* and *normal vector* for every sample point takes part. The tangent plane consists of the surface normal $\mathbf{n}$ (always of length 1) and the plane center $\mathbf{C}$ that can be computed as an average of neighboring points. Then the surface normals are consistently oriented, which means that for neighboring points $\mathbf{X}$ and $\mathbf{Y}$, the dot product of the normals $\mathbf{n}_\mathbf{X}^T \mathbf{n}_\mathbf{Y}$ (which is expected to be close to ± 1 since the surface is piecewise smooth) should be rather close to 1 than to -1. An exact solution of an energy function minimization implies a graph-cuts-based minimization, but in [70], a sign-propagation approach is proposed. In the third step, the value of the *signed distance function* $\mathrm{dst}(\mathbf{Y})$ from each node $\mathbf{Y}$ of a volumetric grid is computed by projecting $\mathbf{Y}$ onto the tangent plane i, where the center of the plane i is the closest to $\mathbf{C}$. This function reflects the distance from $\mathbf{Y}$ to the closest point on the surface. Formally, we have $i = \arg\min(\mathrm{dst}(\mathbf{Y}, \mathbf{C}_i))$, the base-point

$$\mathbf{V} = \mathbf{Y} - \mathbf{n}_i \left(\mathbf{n}_i^T \cdot (\mathbf{Y} - \mathbf{C}_i)\right) \text{ and } \mathrm{dst}(\mathbf{Y}) = \mathbf{n}_i^T (\mathbf{Y} - \mathbf{C}_i) \tag{3.3}$$

is set to be the value of the signed distance function if and only if there is a sample point of $\mathcal{X}$ within a sphere of radius ρ around $\mathbf{V}$. Otherwise it is set to infinity. In the last step, triangles are extracted from the volumetric grid by one of the approaches described in Sec. 2.4.2.

Experiments show that the approach of [70] performs well in presence of moderate Gaussian noise. Its another advantages is the topological flexibility: there is no need to differentiate between 2.5D and 3D surfaces. But the approach has the following disadvantages: it is not immediately clear how to take the sample's accuracy (weighted points) into account. For a point $\mathbf{Y}$ quite far from the surface, a correct value of the signed distance function is hard to determine, especially if the surface has boundaries or there are uncertainties in the values of $\mathbf{n}$. Other problem can emerge near the points of the medial axis, where function values can differ from negative to positive and so ghost triangles can appear. Also, the approach does not perform well in regions of rapid curvature changes and non-continuous distribution of normal vectors.

Based on values of the signed distance function retrieved by [70], local adaptive [6] and global [42] methods were developed to support smoothing the function values at grid nodes and also at the intermediate points.

Applying the Fourier transform for water-tight surface extraction

Another well-known method of iso-surface extraction from water-tight surfaces (i. e., those that partition the space into two sets, one with positive and one with negative values of the signed distance function) is given in [75]. Given the point sample and normal vectors $(\mathbf{x}, \mathbf{n})$, the procedure first retrieves the Fourier transform of the characteristic function χ of

the surface ($\chi(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathcal{F}$ and $\chi(\mathbf{x}) = 0$ if $\mathbf{x} \notin \mathcal{F}$) from the point set S and the set of oriented normal vectors using Stokes's theorem.

$$\hat{\chi}(\mathbf{v}) = \int_{\mathbb{R}^3} \chi_{\mathcal{F}} e^{-i\mathbf{v}^T\mathbf{x}} \mathrm{d}\mathbf{x} = \int_{\mathcal{F}} e^{-i\mathbf{v}^T\mathbf{x}} \mathrm{d}\mathbf{x} = \int_{\partial\mathcal{F}} \mathcal{G}_{\mathbf{v}}(\mathbf{x})\mathbf{n}(\mathbf{x})\mathrm{d}\mathbf{x} \approx \sum_{\mathbf{x}\in\mathcal{X}} \mathcal{G}_{\mathbf{v}}(\mathbf{x})\mathbf{n}(\mathbf{x}),$$

where $\hat{\cdot}$ denotes the Fourier-transform, $\mathbf{v} = (k, l, m)$ is a triple of integer numbers and $\mathcal{G}$ is a vector function such as $\mathrm{div}(\mathcal{G}_{\mathbf{v}}(\mathbf{x})) = e^{-i\mathbf{v}^T\mathbf{x}}$ for all $\mathbf{v}$. In [75], the term $\mathcal{G}_{\mathbf{v}}(\mathbf{x}) = i\mathbf{v}e^{-i\mathbf{v}^T\mathbf{x}}/||\mathbf{v}||^2$ is proposed, because it is the only function that is invariant under rotations and translations and by which "no points influences its neighbor".

After obtaining χ by Fast Fourier Transformation, the resulting mesh may be obtained by any polygonization algorithm mentioned in Sec. 2.4.2. Of course, our models are not water-tight. Therefore, the resulting surface must be filtered in an additional step, e. g. by removing pieces of the surface outside the bounding box of $\mathcal{F}$.

3.2.3 Surface reconstruction by level sets

The key idea of the level set method is an exploration of the evolution of the open, possibly multi-connected set $\Omega \in \mathbb{R}^n$, bounded by a hyper-surface $\mathcal{F}$ under influence of a velocity field, see [106]. This velocity field can depend on position, time, geometry of $\mathcal{F}$ and many other factors. The function $\phi(\mathbf{X}, t)$, which (similar to the last section) is positive for $\mathbf{X} \in \Omega$, negative for $\mathbf{X} \notin \Omega \cup \partial\Omega$ and zero at the border $\partial\Omega$, is a kind of characteristic function for Ω. A significant advantage of the representation can be seen from Fig. 3.2: the three different curves in the top of the figure have completely different topology and can hardly be parametrized from a mere intuition. But, if one considers the three-dimensional counter-part of these graphics (in Fig. 3.2, bottom), an evolution principle becomes evident and clear.

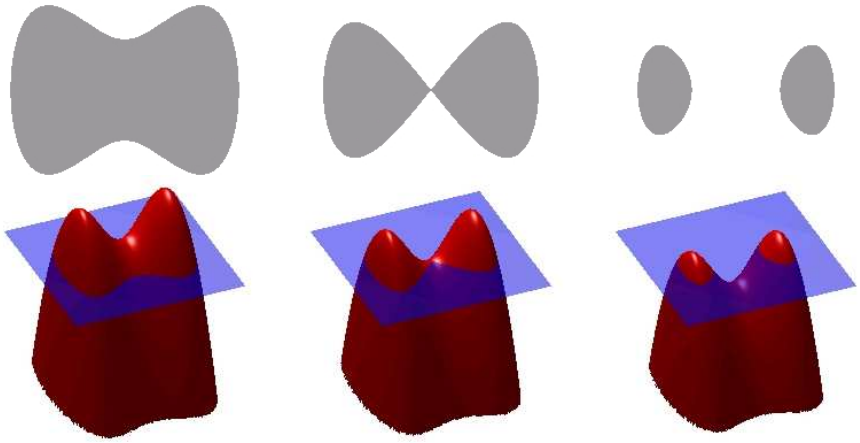

Figure 3.2: Top: the behavior of the level-set function is hard to describe by an explicit function. Bottom: in 3D, it is easy to observe how the level set function merely is moved downward and so parameterization is easier in 3D. Source: Wikipedia.

Reconstruction of open, water-tight surfaces is one of the applications of the level set method. The task is to obtain a *steady-state solution* for a partial differential equation (PDE) $\partial\phi/\partial t + f(\mathbf{X}, \phi, \nabla\phi) = 0$ with a suitable function f; the PDE representation means that the surface is assumed to be a time-dependent function that ideally converges to the correct solution for $t \to \infty$. This surface deformation approach has turned to be a very suitable method to optimize a surface given several kinds of information. A modification of that method will be presented in Sec. 3.3.2, and one of its best-known alternatives is the *snake*

algorithm (see, for example, [65]). The partial differentiation of the PDE mentioned above helps to obtain the Euler-Lagrange equation for its steady-state solution. The resulting functional consists of a data term (which can express, similarly to [106] the distance from the point cloud to $\mathcal{F}$ or some radiometric relations, compare [76]), a smoothness term, which according to [106] punishes the area of $\mathcal{F}$, and, additionally, a salience field, which is used to reduce the number and influence of outliers (see [93] for further information).

The technical details of the approach for obtaining a solution of PDE mentioned above are described in [106]. The first step is an approximation of ϕ and its derivatives on a discretized Carthesian grid. Then the solution of the discretized PDE can be obtained via TVD (total variance diminishing) by Runge-Kutta schemes.

The results of the level set approach are usually visually good even for a high percentage of outliers, especially after being extended with the salience term of [93]. However, the tensor voting procedure works only if the number of inliers is high and their distribution is homogeneous. In addition to the rather high computing time needed to solve the PDE, there are two other reasons why the level set in its straightforward implementation can hardly be applied on our problem. First of all, the model assumption of a C^2-function bias the results towards Gibbs artifacts (over-swinging near sharp edges and gradient discontinuities). The second problem consists of the fact the models to be instantiated in our applications are not necessarily water-tight.

3.2.4 Approximation of surfaces on two-dimensional tensor-product grids

If the assumption of the z-axis as a dominant direction holds (e. g., by flights at sufficiently high altitudes), it is possible to parametrize the terrain along its length and width by independent variables u and v and model the height

$$z_{i,j}(u,v) = A_{i,j}F_i(u)G_j(v), i = 0,...,I, j = 0,...,J \tag{3.4}$$

with basis functions $F(u), G(v)$ of independent parameters u, v and unknown scalars A_{ij}.

Gridfit

The simplest possibility is to let F and G be fixed and model $A_{i,j}$. In the case $F = G = 1$, $A_{i,j}$ represent the function values of z at the nodes (u_i, v_j) and will be denoted by $z_{i,j}$. For example, *gridfit*, a widely used modeling tool available in MATLAB (see [38]) can be applied for obtaining the unknown $z_{i,j}$ and thus a C^0-surface homeomorphic to a plane. The resulting surface has to approximate the points $\mathbf{X} = (x, y, z)$ in the least square sense and the interpolated method can be either:

1 Bilinear: for $u_i < x < u_{i+1}, v_j < y < v_{j+1}$, we have

$$z(x,y) = t\left(sz_{i,j} + (S-s)z_{i+1,j}\right) + (t_j - t)\left(sz_{i,j+1} + (S-s)z_{i+1,j+1}\right), \tag{3.5}$$

where $s = x - u_i, s_i = u_{i+1} - u_i, t = y - v_j, t_j = v_{j+1} - v_j$ (see Fig. 3.3, left).

2 Triangular: here we use the *local barycentric coordinates* of (x, y) in the triangles obtained after tracing the diagonal $z_{i,j}z_{i+1,j+1}$ of the spatial quadrilateral $z_{i,j}z_{i,j+1}$ $z_{i+1,j+1}z_{i+1,j}$. We have:

$$z(x,y) = \begin{cases} \mathcal{U}z_{i,j} + \mathcal{V}z_{i+1,j} + \mathcal{W}z_{i+1,j+1} & s/t \geq s_i/t_j \\ \mathcal{U}z_{i,j} + \mathcal{V}z_{i,j+1} + \mathcal{W}z_{i+1,j+1} & s/t < s_i/t_j. \end{cases} \tag{3.6}$$

The condition $s/t \geq s_i/t_j$ means that (x, y) lies in the upper triangle of Fig. 3.3, right, made by points $(u_i, v_j), (u_{i+1}, v_j)$ and (u_{i+1}, v_{j+1}) with $\mathcal{U}, \mathcal{V}, \mathcal{W}$ are the local barycentric coordinates corresponding to this vertices, while the condition $s/t < s_i/t_j$ is equivalent to (x, y) is incident with the bottom triangle.

3 Nearest neighbor: the coordinates of x and y only have to be rounded towards the nearest vertex of the rectangle. Clearly, this kind of interpolation will be sub-optimal in the majority of cases, but it helps to save computing time.

The first two options for interpolation mentioned in the previous paragraph also have their drawbacks. For example, in (3.5) the result will be different, in general, for curvilinear rectangles, if we replace s, s_i and $z_{i+1,j}$ by t, t_j and $z_{i,j+1}$ respectively, because two lines in space do not necessarily intersect. In (3.6), the result will be different if the other diagonal of the quadrilateral is chosen. The optimization process consists of solving an over-determined system of equations with a sparse, banded-structured left-hand-side matrix $\mathcal{A}$ using well-known methods of linear algebra.

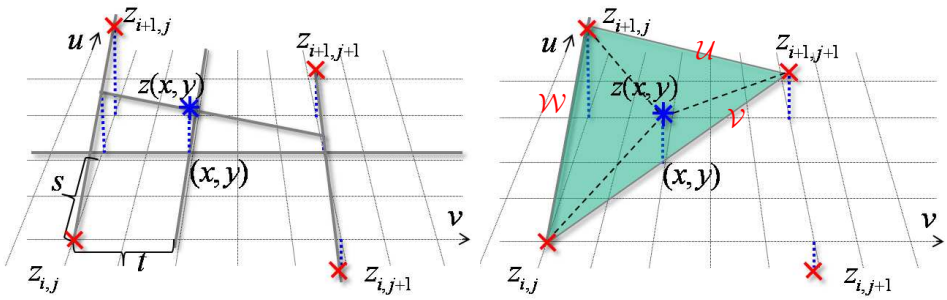

Figure 3.3: Left: Bilinear interpolation of a point (x, y) and (unknown) function values in the grid nodes. Right: Triangular interpolation. See text for further explanation.

Splines

In order to cope for the negative effects mentioned at the end of previous paragraph, one has to use other functions $F(u), G(v)$ as the basis functions in (3.4). Hoschek and Lasser [73] consider in Chapter 6, among others, bicubic polynomial splines

$$z_{i,j} = \sum_{k=0}^{3} \sum_{l=0}^{3} A_{i,j,k,l} (u - u_i)^k (v - v_j)^l. \tag{3.7}$$

Since these splines will be very important for our applications in Sec. 5.2, we now provide the necessary theoretical background about bicubic splines. A bicubic C^1-spline is uniquely determined by the values of the function z and its partial derivatives $\partial z/\partial u, \partial z/\partial v$ at the grid vertices (u_i, v_j) which we denote by $z_{i,j}, (z_{i,j})_u, (z_{i,j})_v$, respectively.

The integration of a data point (x, y, z) into the matrix $\mathcal{A}$ succeeds by assigning it to one of four triangles built by the diagonals of the cell containing (x, y) and computing its Sibson-element [57, 85]:

If (x, y) lies in the triangle specified by:

$$\left\{ (u_i, v_j), (u_{i+1}, v_j), \left(\frac{u_i + u_{i+1}}{2}, \frac{v_j + v_{j+1}}{2} \right) \right\}$$

then

$$\tilde{x} = (x - u_i)/s_i, \tilde{y} = (y - v_j)/t_j \text{ with } s_i = u_{i+1} - u_i, t_j = v_{j+1} - v_j$$

lies in the triangle T_0 specified by vertices $(0,0),(1,0)$ and $(1/2,1/2)$ and we can express the function value for z in terms of function and derivative values at the vertices of the corresponding rectangle by means of the following equation.

$$\begin{aligned} z(x,y) &= \left[1-3\tilde{x}^2+2\tilde{x}^3-3\tilde{y}^2+3\tilde{x}\tilde{y}^2+\tilde{y}^3\right] z_{i,j} \\ &+s_i\left[\tilde{x}-2\tilde{x}^2+\tilde{x}^3-\frac{\tilde{y}^2}{2}+\frac{\tilde{x}\tilde{y}^2}{2}\right](z_{i,j})_u+t_j\left[\tilde{y}-\tilde{x}\tilde{y}-\frac{3\tilde{y}^2}{2}+\tilde{x}\tilde{y}^2+\frac{\tilde{y}^3}{2}\right](z_{i,j})_v \\ &+\left[3\tilde{x}^2-2\tilde{x}^3-3\tilde{x}\tilde{y}^2+\tilde{y}^3\right] z_{i+1,j}+s_i\left[-\tilde{x}^2+\tilde{x}^3+\frac{\tilde{x}\tilde{y}^2}{2}\right](z_{i+1,j})_u \\ &+t_j\left[\tilde{x}\tilde{y}-\frac{\tilde{y}^2}{2}-\tilde{x}\tilde{y}^2+\frac{\tilde{y}^3}{2}\right](z_{i+1,j})_v+\left[3\tilde{y}^2-3\tilde{x}\tilde{y}^2-\tilde{y}^3\right] z_{i,j+1} \\ &+s_i\left[\frac{\tilde{y}^2}{2}-\frac{\tilde{x}\tilde{y}^2}{2}\right](z_{i,j+1})_u+t_j\left[\tilde{x}\tilde{y}^2-\tilde{y}^2+\frac{\tilde{y}^3}{2}\right](z_{i,j+1})_v \\ &+\left[3\tilde{x}\tilde{y}^2-\tilde{y}^3\right]z_{i+1,j+1}+s_i\left[-\frac{\tilde{x}\tilde{y}^2}{2}\right](z_{i+1,j+1})_u+t_j\left[-\tilde{x}\tilde{y}^2+\frac{\tilde{y}^3}{2}\right](z_{i+1,j+1})_v . \end{aligned} \tag{3.8}$$

Expressions for the Sibson element in the other three triangles can be created by mapping these triangles onto T_0. For instance, for the triangle with vertices $\{(x_{i+1},y_j),(x_{i+1},y_{j+1}),$ and $(x_i+x_{i+1},y_j+y_{j+1})/2\}$ (or, equivalently, $(\tilde{x},\tilde{y})$ vertices $(1,0)$, $(1,1)$, and $(1/2,1/2)$), one adjusts (3.8) by replacing $\tilde{x}$ by $1-\tilde{y}$ and $\tilde{y}$ by $\tilde{x}$ and by rotating the indices of the four vertices of the cell, replacing $z_{i,j},z_{i+1,j},z_{i+1,j+1}$ and $z_{i,j+1}$ (with derivatives) by $z_{i+1,j},z_{i+1,j+1},z_{i,j+1}$ and $z_{i,j}$, respectively.

Smoothing Surfaces

The surfaces described above are fitting surfaces, in other words, they assume a point set of high accuracy more or less regularly distributed over the parameter domain $[u_0;u_I]\times[v_0;v_J]$. The result of these routines applied for point sets with sparsely covered regions will be poor since the matrix $\mathcal{A}$ will have a multi-dimensional null-space. Similar to Sec. 3.1, we will have to extend the data term:

$$\|z-z(x,y)\|, \tag{3.9}$$

(where $\mathbf{X}=(x,y,z)$ is a sample point and $z(x,y)$ as in (3.4)) by a smoothness term. In the case of gridfit, three possibilities are given:

1 A *Diffusion*, or *Laplacian* term is the weighted norm (weight λ) of the numerical Laplacian Δ of neighboring grid points. For example, for a point i,j such that $i,j>0, i<I, j<J$,

$$\Delta=\left[2z_{i,j}-z_{i-1,j}-z_{i+1,j}\;\; 2z_{i,j}-z_{i,j-1}-z_{i,j+1}\right]^T, \tag{3.10}$$

which contributes two new rows to the matrix $\mathcal{A}$. The weight λ balances data fidelity and hypothesized properties of the surface. For the grid points on the margin of computation domain but not in the corner, only one row of (3.10) is added. The total number of equations thus obtained is $2(I-1)(J-1)+2(I-1)+2(J-1)$.

2 The *Gradient* strategy suggests minimizing the norm of the gradient and is subtly different from what we saw before, since here the directional derivatives are biased to be smooth across cell boundaries in the grid. The total number of equations here is $(I+1)J+(J+1)I$.

3 *Springs* minimizes springs between neighboring nodes as well as between data points and the nodes of the grid. In this case, the nodes drag the surface toward the local mean of the data and therefore it is usually only a suboptimal choice. The total number of equations here is $2m + (I+1)J + (J+1)I$.

One of the first two terms is usually applied in the case of more complicated basis functions, as described in the previous section. Here the balance parameter λ as in (3.10) plays a role similar to that in equations of Sec. 2.3. Theory to guide the choice of λ is not yet well developed.

In the case of conventional splines [73, 122], regions with sharp changes of curvature often cannot be reconstructed correctly. For regions of rapid change of curvature (e. g., corners of building), overshoot (Gibbs) artifacts emerge if the smoothness parameter λ is too small while oversmoothing occurs if λ is too large. One possibility to solve this problem is presented in [20], where reduction of the smoothness parameter near the characteristic edges in the images is proposed; however, these edges have to be identified in advance. Alternatively, the *L_1-spline-based approach*, originally elaborated by Lavery for approximation of 2.5D surfaces, allows non-overshooting and non-oversmoothing reconstruction of regions of sharp change of curvature without requiring additional information, albeit at additional computation cost [84, 85]. In addition, L_1 splines provide accurate terrain reconstruction even in cases with considerable noise and outliers. The remaining problem is thus to generalize this approach for our applications – reconstruction of a fully 3D surface represented by a vector function $\mathbf{X}(u, v)$ under the assumption that the surface is "nearly" 2.5D with the z-axis as dominant direction.

Summarizing the contents of this section, we state that smoothing splines on tensor-product grids are often used to retrieve plausible surfaces approximating noisy point clouds. However, because videos of the urban terrain recorded from a moderate height cannot be represented by a function $z = z(x, y)$ but rather require representation by a parametrized 3D-vector function $\mathbf{X}(u, v)$, the question of parameterization must be solved. Typically, the parametrization by u and v is unknown a priori. If we succeed in finding a suitable parameterization, the probability of obtaining good results is high.

3.2.5 Other methods

Here we will describe several approaches of meshing point clouds that can be applied for the kind of data obtained from our image-based methods. For example, in [99], a constrained Delaunay triangulation [119] of sparse points and endpoints of characteristic edges in every reference image is obtained and afterwards a visibility constraint for every triangle is checked. The triangles in a new reference view that occlude a point obtained in an old reference view are discarded. This approach leads to holes in the mesh and to artifacts resulting from noise and outliers in the data. The group of *space-carving methods* [83] also uses the *power principle*: the more photographs are available, the more difficult it is for 3D points to satisfy either spatial or radiometric constraints and once a surface point fails to satisfy these constraints, no new image of that point can re-establish the reliability of this point.

Several authors [1, 78] (see also contributions mentioned in these two papers) perform surface reconstruction by modifying the well-known *Shepard method* (Hoschek and Lasser [73], Chapter 9) for scattered point approximation. They interpolate on a volumetric grid $\mathbf{Y} = (x_k, y_l, z_m)$ the 3D function

$$\mathbf{a}(\mathbf{Y}) = \frac{\sum_i w_i(\mathbf{Y})\mathbf{X}_i}{\sum_i w_i(\mathbf{Y})} \quad \text{where} \quad w_i(\mathbf{Y}) = -\exp\left(\frac{\|\mathbf{Y} - \mathbf{X}_i\|^2}{\sigma}\right),$$

or some other function that has a maximum at $\mathbf{X}_i$ and decreases toward zero in all directions. Here, σ is a scalar that depends on the distribution and quality of the points. The resulting surface is the zero set of the function

$$f(\mathbf{Y}) = \mathbf{n}^T(\mathbf{Y})\,(\mathbf{Y} - \mathbf{a}(\mathbf{Y})) \quad \text{and} \quad \mathbf{n}(\mathbf{Y}) = \arg\min \left(\sum_i \mathbf{n}^T(\mathbf{Y})\,(\mathbf{X}_i - \mathbf{a}(\mathbf{Y}))\, w_i(\mathbf{Y}) \right)$$

is the (oriented) normal vector field to be estimated. Intuitively, the point sets are locally approximated by planes and the size of the local neighborhood is given by the potentials w_i. Different approaches make use of topological relations between the points and variation of norm (since the L_2-norm is known to be sensitive to noise and outliers).

The approaches of [97, 34] and [117] are dedicated to extracting special kinds of surfaces. The work [97] searches for vertical planar segments from sparse 3D points clouds, since many ghost planes may appear if the assumption of vertical segments identifying building walls is dropped. On the other hand, [34] fits conics in the depth maps. Finally, [117] searches for geometric primitives in laser point clouds using RANSAC with an octree-based evaluation cost function.

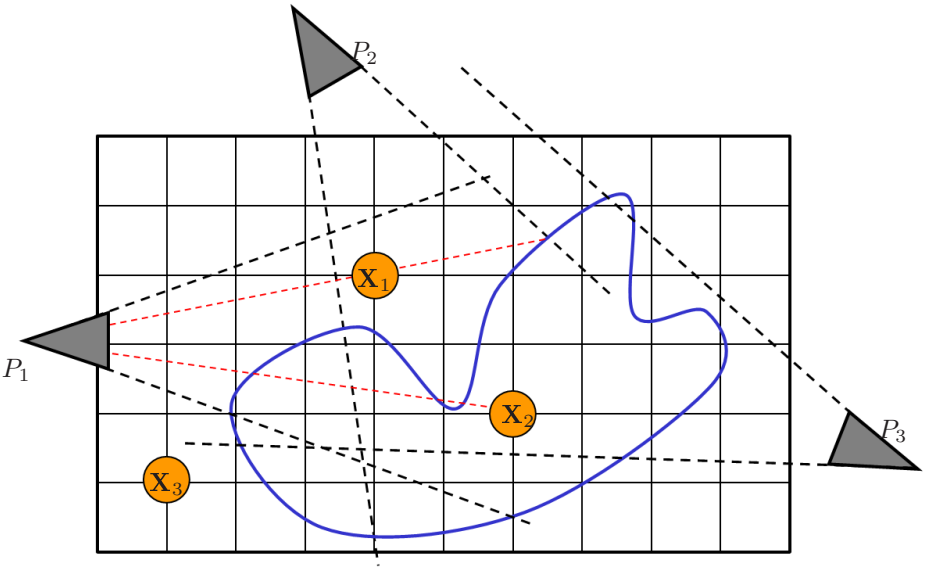

Figure 3.4: A typical approach of surface (illustrated by the blue curve) reconstruction from dense range images. As an approximation of the absolute value of the signed distance function at $\mathbf{X}$ (nodes of a volumetric grid, denoted, in selected cases, by orange circles), one takes $\min(|\delta_i(\mathbf{X})|)$ over all reference images (identified by the corresponding camera matrices P_i) with the sign $+1$ if and only if all $\delta_{\mathbf{X}}$ are positive (as for the point $\mathbf{X}_2$). At the points for which $\delta_i(\mathbf{X})$ cannot be calculated (for instance, $\mathbf{X}_3$), the value of the signed distance function is left undefined.

Curless and Levoy [36] have a set of depth maps $\mathcal{D}_i$ corresponding to several reference images as input and calculate, in a volumetric grid, a signed distance function consisting of a weighted sum of signed distances to the surface in the direction of the camera view. The problem is the choice of function values "behind the surface" which may lead to multi-sheet surfaces. A possible solution consists in keeping track of the union of all regions behind the surface and setting its signs after all depth maps are processed [80]. A typical constellation is shown in Fig. 3.4, where, for each grid point $\mathbf{X}$ and each reference camera P_i, the term $\delta_i(\mathbf{X}) = |\mathbf{C}_i\mathbf{X}| - \mathcal{D}_i(P_i\mathbf{X})$ can be calculated. (Note that, in an manner analogous to Fig. 3.4, we can write instead of the distance $|\mathbf{CX}|$ between $\mathbf{X}$ and camera center, the depth value

$d_{\mathbf{X}}$; see Sec. 5.1.1.) The sign of the signed distance function at $\mathbf{X}$ is positive if and only if all $\delta(\mathbf{X})$ are positive. Turk and Levoy [129] remove the redundant parts of the meshes, connect their boundaries and, finally, update the positions of the vertices. We assume that the results of this algorithm will be similar to the iso-surface extraction, since the mesh is not expected to be topologically consistent and the values of signed distance function are not exact. As an example, one sees that the bad approximation of the signed distance function in $\mathbf{X}_1$ and $\mathbf{X}_2$ resulting from the depth map at P_1 (specified by red dashed lines in Fig. 3.4) can be corrected by P_2. However, in reality, such a P_2 is either not necessarily given or may be occluded by another object.

3.3 Overview of three existing reconstruction pipelines

There is quite a large amount of work on textured 3D reconstruction from images and videos because of the importance and elegance of this area. In this section, we will present several of approaches and discuss their applicability for our data. In particular, we will learn form Schlüter's dissertation of Sec. 3.3.1 how to create, starting from a coarse 2.5D triangulation, a 3D description of surface patches in *local coordinates*. The idea of *enriching* a sparse 3D point set by means of radiometric relations and then fusing an enriched point set into a 3D surface is presented in Sec. 3.3.2 and a *real-time oriented incremental approach* of local tessellations from depth maps is given in Sec. 3.3.3. For additional relevant work on reconstruction pipelines that go the complete way from image sequences to textured 3D models, we refer to [120].

3.3.1 Schlüter's thesis

The approach of [116] generalizes global methods for 2.5D surface-fitting on a rectangular grid by several images [63, 135] and uses multi-grid method to obtain dense 3D models without prior knowledge about the surface. The 3D nodes are vertices of a global triangulation to be defined for every pyramid level. The observations are defined for every pixel in every image that covers a patch $\mathcal{F}$ of the surface. The selection of images succeeds by means of visibility constraints previously computed. Both the point of intersection of a reprojection ray with $\mathcal{F}$ and its local barycentric coordinates within the corresponding triangle in space can be computed, which allows computing surface normals and main curvature directions. The solution of the resulting differential equation presupposes updating $\mathcal{F}$ by means of sliding 3D points in the direction of their normal vectors. Of course, updating the position of every single point can lead to completely wrong results, since the interactions of neighboring pixels are not considered. Therefore, a regularization term that consists of a distance function between local tangent planes for adjacent points is added.

The bottleneck of the method is the choice of the initial triangulation. While the author claims the 2.5D Delaunay triangulation of x and y coordinates of the available 3D points is good enough for the initialization, it is clearly not sufficient for our applications where the sensor platform may be located near the walls of the building, so that, in the case of balconies and overhanging roofs, projection of points into the xy-plane will not correspond to correct topological relations between the points. Varying density of the 3D points obtained by photogrammetric methods does not contribute to the stability of such a triangulation. Since minimization parameters include both geometry and color information (together with the local values of brightness and contrast), the parameter matrix becomes rather large and the solution cannot easily be computed for a large number of images covering a broad scene. Therefore the 3D data presented in [116] include only a few high-resolution intensity images around a small object (a single house) and not a complete, theoretically infinite video

sequence with a lot of redundant information. For large data sets, it will be an advantage to split the process up into the image-based and point-based stage.

3.3.2 Reconstruction by Furukawa and Ponce

The key idea of this work [48] is to obtain a set of patches that are parts of the object surface using a sophisticated region-growing system. Every patch is characterized by its center and normal in the direction of its reference image. These two parameters are obtained by minimizing the NCC-score. Initial guesses are given by matching algorithms for characteristic features [92, 60] along epipolar lines in the images. At the initial stage, the patch must be visible in at least two images and not be occluded by other patches in other images that can potentially see it. At the expansion stage, neighbors of already reconstructed patches must be added to the reconstruction. For accomplishing this task, images are partitioned into quadratic cells, each of which can potentially contain several patches. The empty cells the neighbors of which contain already reconstructed patches are explored. The next stage is filtering, where first patches that occlude more than n patches and finally patches that are occluded by more than m other patches (n, m are automatically calculated thresholds) are deleted.

Since, until this stage, the algorithms are local, many outliers are expected and a subsequent filtering stage is indispensable. Since patches are sparse in space and even more holes will be left after the filtering process, Furukawa and Ponce propose a post-processing optimization that is described in [47]. An energy function that includes a smoothness term for minimization the second derivatives of local parameterizations of mesh nodes, a photometric consistency term based on the reconstructed patches in the first phase, and a visibility term that is additionally inserted in the case accurate silhouettes are available, is minimized in the last step.

Similar to Schlüter's method, the authors strive to use all available information at the same time. Using already available point correspondences while expanding patch sets (which will be partly inferred in Chapter 4) and considering color/intensity information while post-processing makes results more robust. In the current implementation, this method produces a combinatorial explosion for a large number of images (which is given in our case because we deal with theoretically infinite video sequences with uncertainties in camera positions), but can be modified for incremental processing. Another drawback of this method is insufficient investigation of its performance for critical motions, such as forward/backward motion, where not all points are situated in front of all cameras. Moreover, the post-processing step without visibility is biased towards shrinking models, which can produce the empty set as output; in the case of water-tight models, Furukawa and Ponce prefer using Kazhdan's method ([75], see Sec. 3.2.2) to perform the post-processing step.

3.3.3 Reconstruction algorithm by Nistér et. al.

The system presented by Nistér et. al. in [111] can create textured models from a geo-registered video taken from a moving ground vehicle. The process is incremental, so model generation can be performed in real time. There are four parts of the reconstruction pipeline that are interesting for our purposes. First, a plane-sweep algorithm that allows obtaining depth maps from several images is presented. Then the concept of fusing depth maps, which has several simple depth maps as input, is described. Then a triangular mesh from a reference image is obtained. Finally, interaction of several such triangular meshes is obtained by deleting wrong and redundant triangles.

The concept elaborated in [111] will be partly adopted for our work. However, there are several significant differences: while [111] assumes the set of several cameras to be fixed in the ground vehicle and uses an internal navigation unit, model assumptions can be made

that facilitate, clearly, the reconstruction. For example, directions of dominant planes are given by the ground plane and facades whose approximate positions can be easily determined (Sec. 6.2 of [111]). Moreover, resolution of depth does not change that dramatically as for the aerial view, as one can see in Figs. 6.3 and 6.45 on pp. 84 and 124, respectively, of this present thesis, since the distance between the points on the surface corresponding to adjacent pixels can differ by up to several meters. The question is, consequently, that of finding a post-processing routine that allows computation a global mesh connecting points in different parts of surface.

Chapter 4

Multi-view algorithms for depth maps estimation

The goal of this chapter is to obtain a dense 3D point set from a set of images, corresponding camera matrices and also a sparse, but precise and reliable set of 3D points used for retrieving relative orientation of cameras. Of course, such points can come from other sources, like LIDAR points or manually measured ground control points. However, in our case, these points are automatically extracted from the images and so usually stem from rather textured areas and have extremely low density in the untextured regions. Each short subsequence of 5 to 10 images that we consider in this chapter has a reference frame $\mathcal{I}_0$, typically in the middle of the subsequence. It can be assumed that the Non-Lambertian specular components can be neglected in relations between corresponding pixels in different images of the subsequence. The desired output is the depth information of (almost) every pixel of $\mathcal{I}_0$ with maximum accuracy. We do not care about the (theoretically unlimited) length of the video stream, but will show in the next chapter how the outliers can be successfully removed by using several of reference images and simple geometric constraints.

The proposed pipeline of point homogenization consists of two optional steps. The first step concerns characteristic points whose positions in 3D space are to be determined with maximum accuracy. This process, used for *enriching* the already available point set, is called *sparse tracking and triangulation*. The Delaunay triangulation of these points in images will support the second step, namely, the *pixel-wise depth computation* for which the smoothness constraints as in Sec. 2.3 must be enforced.

Derivation of the most important relations for point-projection in multi-view configurations, choice of characteristic points, initial values of the unknown depth by means of triangular meshes, sparse tracking and triangulation, and dense matching will be described in Secs. 4.1, 4.2, 4.3, 4.4 and 4.5, respectively. We shall make a difference between a rectified binocular configuration and a multi-view configuration (and thus subdivide Secs. 4.3-4.5) not only in order to describe simple, but reliable heuristics for outlier rejection in the case of geometrically less stable binocular configurations, but for the sake of differences in terms of disparity and depth estimation, since for the rectified binocular case, we do not need 3D points and can work only in terms of disparities.

It is important to emphasize that either of the two steps mentioned above can be omitted, usually at the cost of reduced accuracy of the reconstruction. Sparse tracking and triangulation can be omitted and the (Delaunay) triangulation $\mathcal{T}$ of the already available points in $\mathcal{I}_0$ can be thus the input for Sec. 4.5, but then $\mathcal{T}$ will probably consist of very small triangles in textured areas of $\mathcal{I}_0$ and large triangles far away from the surface in textureless

areas of $\mathcal{I}_0$. As a consequence, the evaluation of triangles into consistent or inconsistent with the surface and thus rendering local tessellations will not have much sense. If the second step, the dense estimation, is omitted, only the (enriched) point cloud and the triangles of $\mathcal{T}$ will be output of this chapter. However, even though some triangles that do not reflect correct depth information can often be filtered by considering further reference frames and local methods for shape reconstruction, which we will describe in Sec. 5.1, the assumptions of many surface reconstruction methods [4] will generally not be satisfied because of the low density of points in textureless areas.

4.1 Multi-view geometry

The goal of this section is to establish fast point projection relations that can be used for projecting millions of pixels into dozens of images for dense reconstruction. The best way to parametrize spatial coordinates of points with a minimum of unknown parameters is to consider the depth values d of pixels in the *reference image* $\mathcal{I}_0$ of a sequence, because the search space for point correspondences is one-dimensional and the explicit computation of 3D points is not required. We denote the camera corresponding to I_0 by P_0, as visualized in Fig. 4.1, and call P_0 the *reference camera* of the sequence. If P_0 is a classical pinhole camera, then the depth $d := d_{\mathbf{x}}$ of the 3D point $\mathbf{X}$ corresponding to a pixel $\mathbf{x} = (x, y) \in \mathcal{I}_0$ is the distance from $\mathbf{X}$ to the image plane of $\mathcal{I}_0$ and is given by (see e. g. [61]):

$$d(\mathbf{X}) = (d_{\mathbf{x}}) = \ \mathrm{sgn}(\det(M))P_0^3\mathbf{X}/\|M^3\|, \tag{4.1}$$

where $\cdot^3$ is the third row of $\cdot$, and $M = P_0^{\{4\}}$ is the 3×3 matrix obtained after omitting the last column of P_0. Throughout this work, the camera matrix P_0 will be normalized, i. e. divided by the quantity $\mathrm{sgn}(\det(M))\|M^3\|$. In homogeneous coordinates, we denote the vector $[x\ y\ 1]^T$ by $\check{\mathbf{x}}$ and we prove, starting from (4.1), the following result:

Result 1: The coordinates of the 3D point $\mathbf{X}$ corresponding to $\mathbf{x}$ are given by:

$$\mathbf{X} = d \cdot M^{-1}\check{\mathbf{x}} + \mathbf{C}_0 \tag{4.2}$$

(as a function of d) while the reprojection of $\mathbf{x}$ into the image $\mathcal{I}_k$ will be induced by the transformation:

$$\check{\mathbf{x}}_k(d) \simeq H_{0,k}\check{\mathbf{x}} + \frac{\mathbf{e}_k}{d} \tag{4.3}$$

where $H_{0,k} = P_k^{\{4\}}M^{-1}$, $\mathbf{e}_k = P_k\check{\mathbf{C}}_0$ are the infinite homography and the epipole, respectively. Since (4.3) denotes equality up to a multiplicative constant, one can perform a further substitution (with an arbitrary real scalar d_0) in order to reduce point projection to addition of 2D points:

$$\mathbf{x}_k(d) = \hat{\mathbf{h}}_k + t\hat{\mathbf{e}}_k \ \text{ where } \ t = \frac{d - d_0}{d + e_k^3/h_k^3} \ \text{ and} \tag{4.4}$$

$$\hat{\mathbf{h}}_k \simeq H_{0,k}\check{\mathbf{x}} + \mathbf{e}_k/d_0,\ \hat{\mathbf{e}}_k = \frac{1}{h_k^3\,(d_0h_k^3 + e_k^3)} \begin{bmatrix} h_k^1e_k^3 - h_k^3e_k^1 \\ h_{ik}^2e_k^3 - h_k^3e_k^2 \end{bmatrix}.$$

Proof: Since the fronto-parallel plane π at distance d from the image plane has the equation:

$$\pi(d) = P_0^3 - (0\ \ 0\ \ 0\ \ d)\,,$$

the coordinates of the 3D point $\mathbf{X}$ are given by:

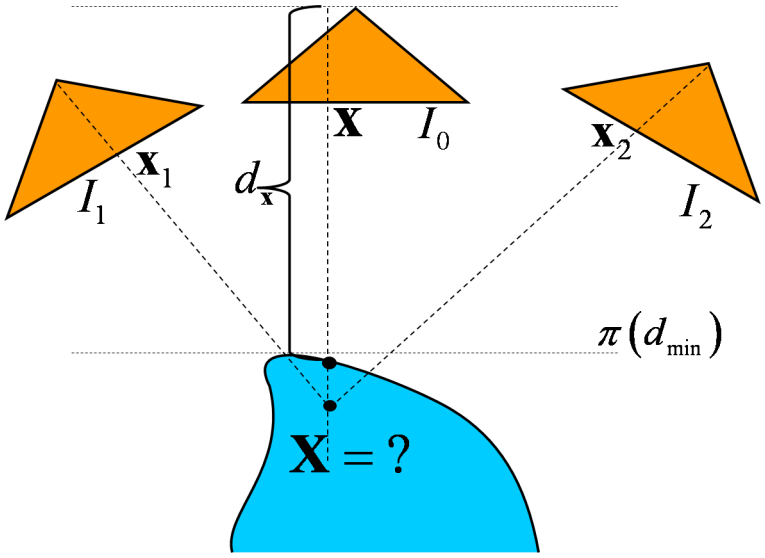

Figure 4.1: Point projection in multi-view configurations. Cameras are depicted by orange pyramids on the top, the object surface is below. A point $\mathbf{x} \in \mathcal{I}_0$ with depth $d_{\mathbf{x}}$ induces a 3D point $\mathbf{X}$ that can be projected to images $\mathbf{x}_1 \in \mathcal{I}_1$ and $\mathbf{x}_2 \in \mathcal{I}_2$. Matching can then succeed by comparing color/intensity values of $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2$.

$$\check{\mathbf{X}} = \begin{bmatrix} P_0 \\ \pi(d) \end{bmatrix}^{-1} \begin{bmatrix} \check{\mathbf{x}} \\ 0 \end{bmatrix} = \left(\begin{bmatrix} M^{-1} \\ \mathbf{0}_3^T \end{bmatrix} + \begin{bmatrix} \mathbf{0}_4 & \mathbf{0}_4 & \frac{\check{\mathbf{C}}_0}{d} \end{bmatrix} \right) \check{\mathbf{x}}$$

and thus $\mathbf{X}$ is given by (4.2). Moreover, (4.3) is also easily obtained:

$$\check{\mathbf{x}}_k(d) = P_k \cdot \mathbf{X} = P_k \cdot \left(\begin{bmatrix} M^{-1} \\ \mathbf{0}_3^T \end{bmatrix} + \begin{bmatrix} \mathbf{0}_4 & \mathbf{0}_4 & \frac{\check{\mathbf{C}}_0}{d} \end{bmatrix} \right) \check{\mathbf{x}} \simeq H_{0,k}\check{\mathbf{x}} + \frac{\mathbf{e}_k}{d}$$

with notations for $H_{0,k}$ and $\mathbf{e}_k$ mentioned above. According to (4.3), the reprojection of a point into the image k can be performed by adding two homogeneous quantities, since the values of $\mathbf{h}_{ik} = H_{0,k}\mathbf{x}_i$ can be saved for every pixel $\mathbf{x}_i$. In order to derive (4.4), the subscripts $\cdot_k$ can be dropped and, because of a strong analogy in the calculations, it is enough to consider only the x-coordinates:

$$\hat{\mathbf{h}} + t\hat{\mathbf{e}} = \frac{d_0 h^1 + e^1}{d_0 h^3 + e^3} + \frac{d - d_0}{d + \frac{e^3}{h^3}} \cdot \frac{h^1 e^3 - h^3 e^1}{h^3 \left(d_0 h^3 + e^3\right)} =$$

$$\frac{d d_0 h^1 h^3 + e^1 e^3 + d h^1 e^3 + d_0 h^3 e^1}{\left(d_h^3 + e^3\right)\left(d_0 h^3 + e^3\right)} = \frac{d h^1 + e^1}{d h^3 + e^3},$$

which completes the proof.

From the already available point correspondences, we approximately know the depth ranges ($d \in [d_{\min}; d_{\max}]$), which allows us obtaining depth ranges for t:

$$t \in \left[0; \frac{d_{\max} - d_{\min}}{d_{\max} + e_k^3/h_{ik}^3}\right], \; t = \frac{d - d_{\min}}{d + e_k^3/h_{ik}^3}, \; d = \frac{t \cdot e_k^3/h_{ik}^3 + d_{\min}}{1 - t}.$$

We now describe the properties of Eqs. (4.2)-(4.4).

If we know the spatial depth of an arbitrary number of points $\mathbf{x}_i \in \mathcal{I}_0$, using (4.2) represents an extremely fast way for obtaining their spatial coordinates, because multiplication

and addition can be performed simultaneously and column-wise. The same argument can be applied for Eqs. (4.3) and (4.4). There are more time-consuming algorithms for obtaining 3D points from point correspondences, which, however, consider uncertainties in camera parameters and point coordinates. Here, the DLT[1] solution by means of singular value decomposition of a $2K \times 4$ matrix for every 3D point, see [61], Chapter 12. The solutions for error-free camera configurations and noisy point correspondences are presented in [61] for two-view configurations and in [123] for three views.

According to (4.3), point projection from image to image can be performed by adding two homogeneous quantities if one stores the values for $\mathbf{h}_{ik}, \mathbf{e}_{ik}$ for every pixel of interest $\mathbf{x}_i$. This fact will be extensively used in Sec. 4.5 when d is a common optimization parameter in arbitrary multi-view configurations and dense sets of pixels. Since t depends on the camera index k in equation (4.4), we cannot, unfortunately, generalize these considerations for t as a common optimization parameter in (4.4), unless $K = 1$ or images $\mathcal{I}_0, \mathcal{I}_k$ are rectified to epipolar geometry. However, for epipolarly rectified images, $e_k^3 = 0$. Hence, t does not depend on k anymore, the transformations concern only the x-coordinates, the time-consuming conversion of (4.3) into inhomogeneous coordinates is not required and, since the influence of all rotation angles except the one around the baseline $\mathbf{C}_0\mathbf{C}_k$ has been compensated, the algorithms of Sec. 4.4 and Sec. 4.5 are made more invariant against rotations. There are also disadvantages of image rectification: First of all, it can be carried out using a linear transformation only if the epipole $\mathbf{e}$ lies outside the image domain and significant distortions of images are inevitable if it is close to the image border. Moreover, due to interpolation errors in the course of image transformation, gradient calculations are less reliable. Throughout this work, we have a *rectification option* opt.r; if (and only if) its value is true and the epipoles are bounded away from the image borders, we rectify the images by means of the algorithm proposed in [90] (see Sec. 2.1). As the result, we have several rectified pairs of images and pairs of homographies. For example, if we rectify $\mathcal{I}_0$ and $\mathcal{I}_k$, we have the rectified images $\mathcal{I}_{0k}^R, \mathcal{I}_{k0}^R$, the homographies H_{0k}^R, H_{k0}^R, and, for every pixel of interest $\mathbf{x}_i \in \mathcal{I}_0$, we store $H_{0k}^R\mathbf{x}_i$, $H_{k0}^R H_{0,k}\mathbf{x}_i = \hat{\mathbf{h}}_{ik}$ as well as $\hat{\mathbf{e}}_{ik}$ in $2K \times N$ matrices and always can perform a sum of 2D points for projection of points.

For two images rectified to epipolar geometry, the first coordinate of the left hand side of (4.3) can be formulated as:

$$\begin{aligned} x_k(d_R) &= \frac{\left(H_{0,k}^{1,1} + e_k^1/d_R\right)x + H_{0,k}^{1,2}y + H_{0,k}^{1,3}}{H_{0,k}^{3,3}} \text{ abbreviated by} \\ x_k(d) &= \mathbf{v}\check{\mathbf{x}}, \text{ where } \mathbf{v} = \frac{1}{H_{0,k}^{3,3}}\left[H_{0,k}^{1,1} + e_k^1/d_R \;\; H_{0,k}^{1,2} \;\; H_{0,k}^{1,3}\right], \end{aligned} \tag{4.5}$$

d_R is the new value of depth in terms of $H_{0k}^R P_0$ and $\mathbf{v}$ is a 1×3 vector.

From (4.5) we can obtain depth (in the terms of rectified images) from the disparity value $j = x_k - x$:

$$d = \frac{e_k^1}{(x+j)H_{0,k}^{3,3} - H_{0,k}^1\mathbf{x}}. \tag{4.6}$$

We illustrate in Fig. 4.2 fast ways of calculating 3D coordinates, depth values of points in terms of original and rectified images as well as disparity values. The time-consuming process for obtaining 3D points from point correspondences requires applying the DLT-algorithm.

[1] Direct Linear Transformation

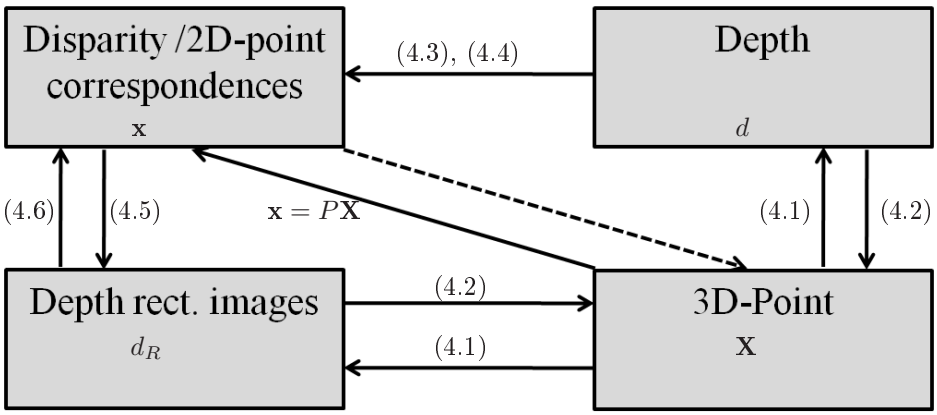

Figure 4.2: Reprojection equations for multi-view configurations. The time consuming pixel-wise triangulation is denoted by a dashed line.

4.2 Choice of characteristic points

If the original point cloud is too sparse and not distributed regularly in the image, we need to obtain 3D coordinates for some additional points. The criteria of state-of-the-art feature extraction procedures must therefore be modified in order to incorporate the given knowledge of camera matrices. In this case, the search range for points is reduced by the one-dimensional epipolar line as indicated by equations (4.3), (4.4) and (4.5). We are interested in points whose neighborhoods have strong intensity changes in the direction parallel to the epipolar lines.

We subdivide the reference image into small squares (e. g. 10×10 pixels) and select, for every square, a point with a maximum response of some *cornerness operator* $C(\mathcal{I})$. For the two-camera case, the authors of [29] considered the *structural tensor* (compare [46, 60]) $A(\mathcal{I})$ for a given image as well as the term

$$\bar{C}(\mathcal{I}) = \operatorname{trace}(A(\mathcal{I})) - 0.04 \det(A(\mathcal{I})), A(\mathcal{I}) = \begin{bmatrix} \tilde{\mathcal{I}}_x^2 & \tilde{\mathcal{I}}_x\tilde{\mathcal{I}}_y \\ \tilde{\mathcal{I}}_x\tilde{\mathcal{I}}_y & \tilde{\mathcal{I}}_y^2 \end{bmatrix}, \tag{4.7}$$

where $\mathcal{I}_{x/y}$ are image gradients given e. g. by the Sobel operator, $\tilde{\cdot}$ is the optional Gaussian smoothness operator. The response of the term $\bar{C}(\mathcal{I})$ given by (4.7) consists of points near corners of the intensity image and so the probability of finding them in the second image is relatively high. We use points obtained by (4.7) mostly in the binocular case. In order to save computing time, we rely, instead of on the structural tensor, only on the gradient operator, namely,

$$C(\mathcal{I}) = (1-\alpha)\tilde{\mathcal{I}}_x^2 + \alpha\tilde{\mathcal{I}}_y^2, \tag{4.8}$$

where $\alpha \in [0, 1]$ is a positive scalar needed to give more support to pixels with intensity changes parallel to the epipolar lines. For example, if the x and y coordinates of the axes in the images approximately coincide with the corresponding coordinates in 3D space and the height of the sensor platform remains approximately constant, the angle between x-axis and epipolar lines is usually small. Therefore α should be chosen close to 1; but even the choice $\alpha = 0.5$ is reasonable. For multi-view configurations, we usually apply (4.8) instead of (4.7). An illustration of the operator $C(\mathcal{I})$ for an infrared image is presented in Fig. 4.3.

If several points with known depth are available, we always compute the Delaunay triangulation of these points in $\mathcal{I}_0$ and replace $C(\mathcal{I}_0)$ of (4.8) by $C(\mathcal{I}_0)C_1(\mathcal{I}_0)$, where $C_1(\mathbf{x})$ is 0

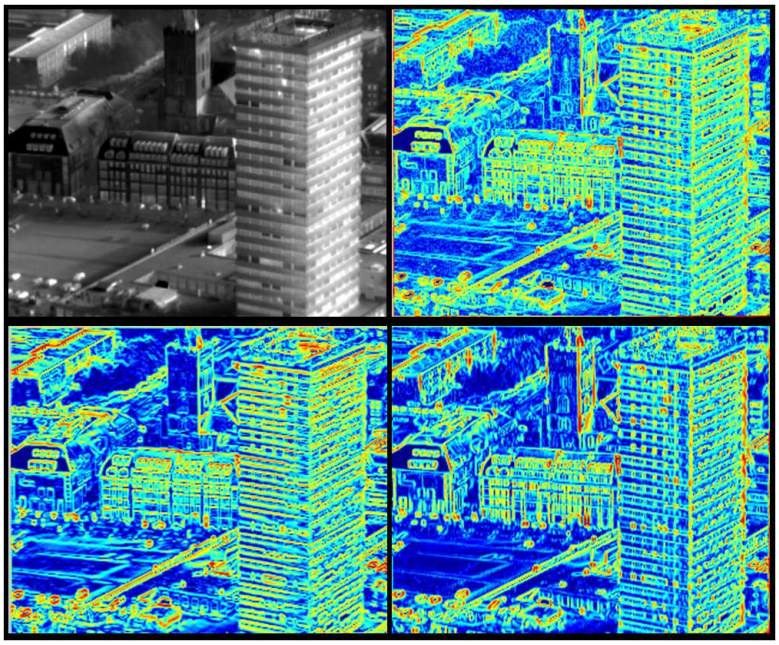

Figure 4.3: Top left: A reference frame of the video sequence *Infrared*. The other three pictures represent $\log(C(\mathcal{I})+1)$ for different choices of α: top right: $\alpha = 0.5$, bottom left: $\alpha = 0.2$, bottom right: $\alpha = 0.8$. As a consequence, horizontal lines are highlighted in the bottom left image and vertical lines are highlighted in the bottom right image.

if the point $\mathbf{x}$ lies within the convex hull of these points and the area of the incident triangle is smaller than a threshold (150-300 pixels in our experiments) and 1 otherwise; so new points will be found in the areas not yet sufficiently covered. The points with response of the cornerness operator below a certain threshold are excluded from further consideration.

4.3 Choice of initial values by means of triangular meshes

Our goal is to obtain depth values for characteristic points from the previous section. This is done by the iterative algorithms of Sec. 4.4, which require initialization. If a characteristic point lies outside the convex hull of points in $\mathcal{I}_0$ with available depth values or no points at all are assigned 3D coordinates, a brute-force procedure consists of evaluating a suitable data cost function (see Sec. 2.2) for several values of an unknown parameter and taking the one that leads to a global minimum. This approach is less sensitive to local minima, but it is time-consuming. A faster method can be applied if several points have already been reconstructed. We obtain a 2D triangulation $\mathcal{T}$ of points already available and consider the support planes of each triangle in 3D space. The initialization consists of intersecting the reprojection ray of a pixel $\mathbf{x}$ with the support plane of the triangle T incident with $\mathbf{x}$. In the following three sections, we describe 1) the process of obtaining the initial disparity (without explicit calculation of the corresponding 3D point) in the two-camera case (Sec. 4.3.1), 2) the process of obtaining initial depth values from multi-view configurations (Sec. 4.3.2) and, finally, 3) the methods used for obtaining a suitable triangulation $\mathcal{T}$ and incidence relations for pixels in $\mathcal{I}_0$ (Sec. 4.3.3) with respect to triangles in $\mathcal{T}$.

4.3.1 Binocular configuration

Suppose two rectified images as well as a set of sparse point correspondences $\mathbf{p}_1$ and $\mathbf{p}_2$ are given. We can assume that the percentage of outliers among these points is low because most of the outliers are supposed to be eliminated by robust methods in Step 1 of our reconstruction pipeline (Sec. 1.2, Alg. 1.1). We are interested in computing correspondences of all points inside the convex hull of the points already available. Consider a triangulation $\mathcal{T}$ of the point set and a triangle $T \subseteq \mathcal{T}$. Suppose that the triangle T is consistent with the object surface, in other words, the surface enclosed by three vertices of T can be nearly replaced by the support plane of T. Then for any point $\mathbf{x} = (x_1, y) \in T$, the corresponding point in the second image is given by:

Result 2: Let $\mathbf{p}_{1,T}, \mathbf{p}_{2,T}$[2] be triplets of corresponding points in two epipolarly rectified images. The homography induced by T maps $\mathbf{x}_1$ onto the point $\mathbf{x}_2 = (x_2, y)$, where $x_2 = \mathbf{v}\check{\mathbf{x}}_1$, $\mathbf{v} = x_{2,T}\left(\check{\mathbf{p}}_{1,T}\right)^{-1}$, $\check{\mathbf{p}}_{1,T}$ is the 3×3 matrix formed by the columns of the projective coordinates of $\mathbf{p}_1$ and $x_{2,T}$ is the row vector consisting of x-coordinates of $\mathbf{p}_{2,T}$.

Proof: Since triangle vertices $\mathbf{p}_{1,T}, \mathbf{p}_{2,T}$ are corresponding points, their correct locations are on the corresponding epipolar lines. Therefore, they have pairwise identical y-coordinates. Moreover, the epipole is given by $\mathbf{e}_2 = [1, 0, 0]^T$ and the fundamental matrix is $F = [\mathbf{e}_2]_\times$. Inserting this information into Result 13.6 on p. 331 of [61] proves, after some simplification, the statement of **Result 2**.

We wish to understand the nature of the parameter $\mathbf{v}$, first mentioned in Eq. (4.5). A scene plane π (visualized by one of the two red segments in the left hand side portion of Fig. 4.4 connecting points with already available 3D coordinates) induces an image-to-image homography H_π which has three degrees of freedom [61]. These three degrees of freedom stem from a plane equation and are stored in $\mathbf{v}$. On the other hand, π can be defined by three non-coplanar points, which can be interpreted as three vertices of a triangle T in space. By **Result 2**, we have a relation that connects the vertices of T and the vector $\mathbf{v}$ without mentioning intermediate results π or H_π.

According to **Result 2**, the disparity in the second image is given by

$$j_{T,\mathbf{x}} = \mathbf{v}\check{\mathbf{x}} - x_1, \quad , \mathbf{v} = x_{2,T}\left(\check{\mathbf{p}}_{1,T}\right)^{-1}$$

which not only provides an initialization for the algorithms of Sec. 4.4, but also a coarse approximation for the disparity/depth map itself, especially in areas where the surface is approximately piecewise planar and does not have many self-occlusions, as illustrated in the example of Fig. 4.5. To compute this approximation $\mathcal{D}_\mathcal{T}$, it is sufficient to determine and store the entries of $\mathbf{v}$ for each triangle; the disparities of any other point – with not necessarily integer coordinates – are computed according to **Result 2**. An optional step for improving the quality of the initial depth map is to fit planes by clustering the values of $\mathbf{v}$ while considering neighborhood relations. This will be a subject of future work.

4.3.2 Multi-view configuration

From the already available 3D points, we can obtain the depth values by equation (4.1). The depth value of a point induced by the triangulation is given by a linear combination of depth values at the vertices of the corresponding simplex (epipolar line endpoints in Fig. 4.4, left, for 2D and triangle vertices T in Fig. 4.4, right, for 3D). In the two-dimensional analogy of triangular interpolation, Fig. 4.4, left, the coefficients of the linear combination are given by proportions $\mathcal{U}, \mathcal{V}$ of lengths of small segments vs. the total segment length. In 3D, these

[2] Here $x_T, y_T, \mathcal{P}_T$ etc. are $x, y, \mathbf{x}$-coordinates (respectively) of triangle vertices specified by a triplet of integer numbers T.

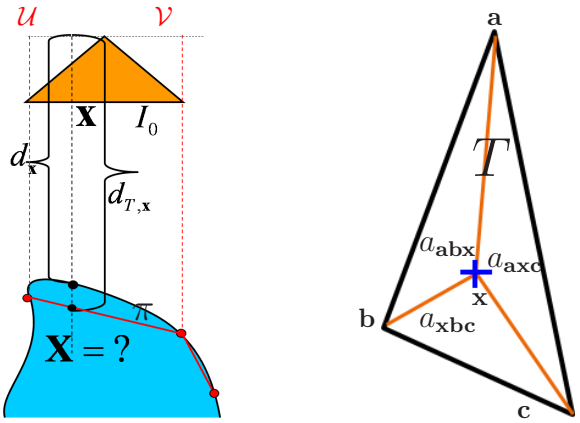

Figure 4.4: Left: The idea of triangular interpolation (same symbols as in Fig. 4.1). Several already reconstructed points are denoted by red circles and the triangulation by solid red lines. The initial estimation of $d_{T,\mathbf{x}}$ is retrieved from triangular meshes of already available points (whose depths are indicated by red dashed lines) either by means of local homographies given by the plane π (as in **Result 2**) or by means of *local barycentric* coordinates of $\mathbf{x}$ within triangle T (see text to Sec. 4.3.2). To obtain the local barycentric coordinates, the areas of small triangles, must be divided by the area of T, as depicted on the right.

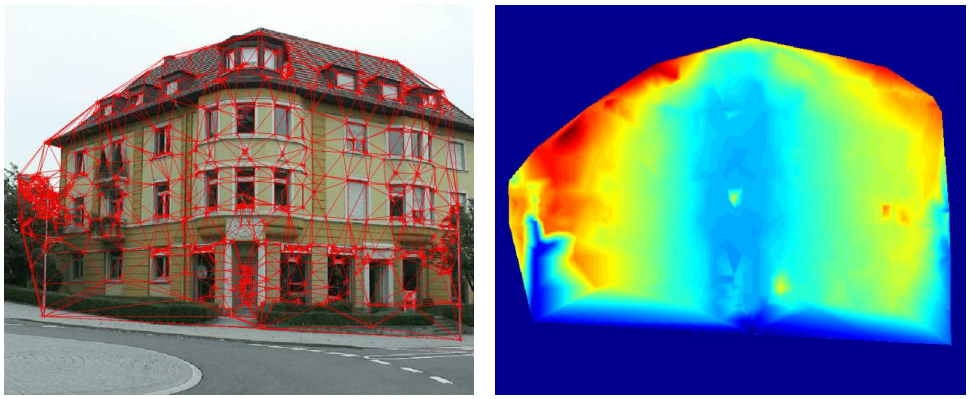

Figure 4.5: Left: A reference frame of the video sequence *House* and a sparse point set (resulting from the structure from motion algorithm [22]) with Delaunay triangles depicted in red. One sees the abundant density of points in highly textured regions (e. g. on a tree) and in the door lattice while the density of points in textureless areas (road and roof) is relatively low. Right: Initialization of the depth map obtained from the depth values of the points on the left and Eq. (4.9). There are also some outliers on the house walls visible by sudden depth changes with respect to their neighbors. These outliers usually stem from reflections in the windows.

are proportions $\mathcal{U}, \mathcal{V}, \mathcal{W}$ of the areas of the small triangles vs. the total area, as illustrated in Fig. 4.4, right, p. 52. This proportions are the the well-known local barycentric coordinates $\mathcal{U}, \mathcal{V}, \mathcal{W}$ of $\mathbf{x}$ in T. Formally we have:

$$d_{T,\mathbf{x}} = \mathcal{U} d_{\mathbf{a}} + \mathcal{V} d_{\mathbf{b}} + \mathcal{W} d_{\mathbf{c}}, \;\; \mathcal{U} = \frac{a_{\mathbf{xbc}}}{a_{\mathbf{abc}}}, \mathcal{V} = \frac{a_{\mathbf{axc}}}{a_{\mathbf{abc}}}, \mathcal{W} = \frac{a_{\mathbf{abx}}}{a_{\mathbf{abc}}} \tag{4.9}$$

and a denotes the area of a triangle. Equation 4.9 shows the advantage of parameterizing the 3D points according to their depth, not according to their distance to the projection center. Similar to Sec. 4.3.1, $d_{T,\mathbf{x}}$ will from here on denote the depth value resulting from triangular interpolation.

4.3.3 Choice of triangulation and establishing incidence relations

The remaining questions for this section are which kind of triangulation to apply (since we already know, for example, from (3.6) that the results of the interpolation depend on the triangulation) and how to assign to a point $\mathbf{x} \in \mathcal{I}_0$ the incident triangle in $\mathcal{T}$. The *Delaunay* triangulation was chosen because of its easy availability in many software packages and because the max-min principle allows excluding more (visually unpleasant) long and thin triangles. There is one more reason – actually an answer to the second question – for choosing Delaunay triangulations. Suppose we want to determine in which triangle $T \in \mathcal{T}$ a point $\mathbf{x}$ lies. There exist algorithms that allow finding T in linear time when $\mathcal{T}$ is the Delaunay triangulation. For example, one can calculate the vertex of the *Voronoi-polygonization* that is the closest to $\mathbf{x}$.

If the cardinality of the point set is large, using these algorithms for each point becomes computationally expensive. An alternative, on which we follow up in this work, is to create a segmented image where a triangle is labeled by its number. The points outside the convex hull are labeled by -1. The process of labeling is very fast and it also has an advantage that the barycentric coordinates or any other scalar value (for example, the area of the incident triangle, mentioned in the last paragraph of Sec. 4.2) can be stored for each pixel once and for all. The result of this routine works quite well (especially if the images and point coordinates are upscaled by a factor of 2 to 4, depending on image size) and with only several mismatches near the border of rather skinny triangles.

4.4 Sparse tracking and triangulation

The task of this section is to *enrich* the point set, in other words, to find the correspondences for new characteristic points obtained in Sec. 4.2. From the resulting, extended point set, we will again use Delaunay triangulation to determine the set of triangles consistent with the surface.

4.4.1 Binocular configuration

We will first turn our attention to the binocular case. This configuration is rather unstable for obtaining point correspondences because of spurious matches in the textured areas and in the image regions near occlusions. We assume that the images $\mathcal{I}_1, \mathcal{I}_2$ are rectified to epipolar geometry and we search for point matches within corresponding epipolar lines.

As visualized in Fig. 4.6, for a point $\mathbf{x} = (x_1, y)$ in a triangle $T \in \mathcal{T}$, the search window can potentially be reduced to

$$\begin{gathered} W_s = [x_1 + x_{\min}; x_1 + x_{\max}] \times [y - \varepsilon_y; y + \varepsilon_y], \\ x_{\min} = \max(j_{\min} - \varepsilon_x, \min(s_T)), x_{\max} = \min(j_{\max} + \varepsilon_x, \max(s_T)), \end{gathered} \tag{4.10}$$

where $\varepsilon_x = \varepsilon_y$ are fixed scalars which cope for uncertainties in camera orientations, s_T are the x-coordinates of at most six intersection points between the epipolar lines at $y, y-\varepsilon_y, y+\varepsilon_y$ and the edges of $\mathbf{p}_{1,T}$ and $j_{\min}, j_{\max}$ are the estimates of disparities ranges which can be obtained from the point coordinates already available.

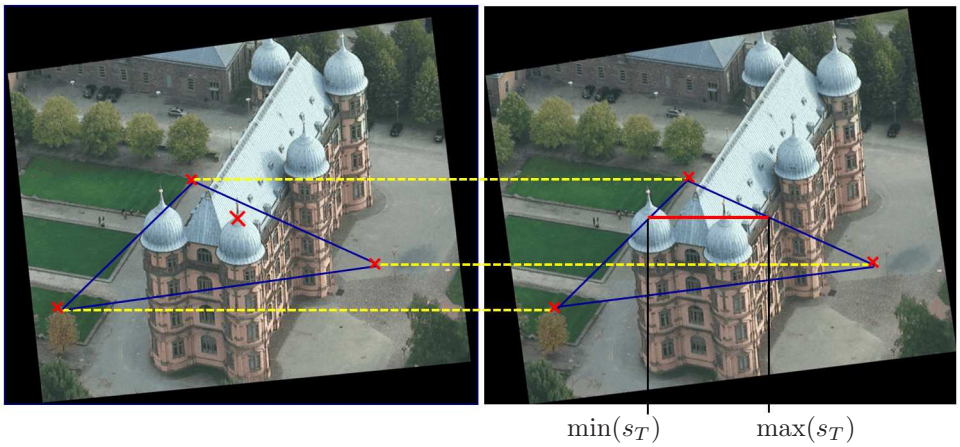

Figure 4.6: Matching supported by triangular meshes in binocular case. An exemplar triangle T from triplets of corresponding points $\mathbf{p}_{1,T}, \mathbf{p}_{2,T}$ (small red crosses) is depicted by thin blue lines in both images. The search range for correspondences within T (a point marked by a big red cross) can often be further constrained by taking into account intersection points of epipolar lines (denoted for a selected point by a thick red line) with edges of T. In degenerated cases of occlusions in triangles inconsistent with the surface, this assumption does not hold, but mismatches are usually excluded by applications of one of three filters imposed on putative correspondences.

The search for correspondence points can succeed be means of any data cost function mentioned in Sec. 2.2. In [29], it was the *Normalized Cross Correlation* (NCC), see Eq. (2.6) between quadratic windows $\mathcal{I}_1(\omega(\mathbf{x}_1))$ of size between 5 and 21 pixels and $\mathcal{I}_2(\omega(\mathbf{x}_2))$. Application of NCC is reasonable here since we can assume a piecewise linear transformation between luminance values of both images, see (2.5). However, in order to avoid including mismatches in the set of correspondences, three filters on the result are imposed before the correspondence $\mathbf{x}_1, \mathbf{x}_2$ is added:

1. The luminance difference between the windows is bounded, i. e. $\|\mathcal{I}_1(\omega(\mathbf{x}_1)) - \mathcal{I}_2(\omega(\mathbf{x}_2))\|_1 < w u_{\max}$ where w is the number of pixels in the window and $u_{\max} = 15$ in our experiments,

2. The correlation coefficient $c_0(\mathbf{x}) = \min_j(\mathbf{x}, j)$ of the winner is low enough (for example, below the threshold 0.5), and

3. In order to reject ambiguous correspondences, c_0 must be low enough with respect to the neighbors. Let c_1 be the best matching coefficients in the sub-windows

$$([x_1 + x_{\min}; x_2 - 2] \cup [x_2 + 2; x_1 + x_{\max}]) \times [y - 1; y + 1].$$

   If the ratio c_0/c_1 (best to second-best) exceeds a threshold (which is usually 0.9), the match is rejected.

The coordinates of corresponding points can be refined to subpixel values. We first check whether $j_{\mathbf{x}} \approx j_{T,\mathbf{x}}$, which can be the case if T is consistent with the surface. The

disparity $j_{T,\mathbf{x}}$ is assigned to $\mathbf{x}$ if and only if $|j_{\mathbf{x}} - j_{T,\mathbf{x}}| < 1$. Otherwise, the subpixel value of j can be assigned according to one of the four methods discussed in [128]. For the sake of computing time, subpixel coordinates for correspondences are computed according to correlation parabolas (second-order curves fitted into the cost distribution function). We denote by c_- and c_+ the correlation values in the pixels to the left and right of x_2. The correction term $\hat{x}_2$ in the x-direction is then given by

$$\hat{x}_2 = x_2 - \frac{c_+ - c_-}{2(c_- + c_+ - 2c_0)}.$$

In Fig. 4.7, new correspondences obtained from binocular sparse tracking are shown.

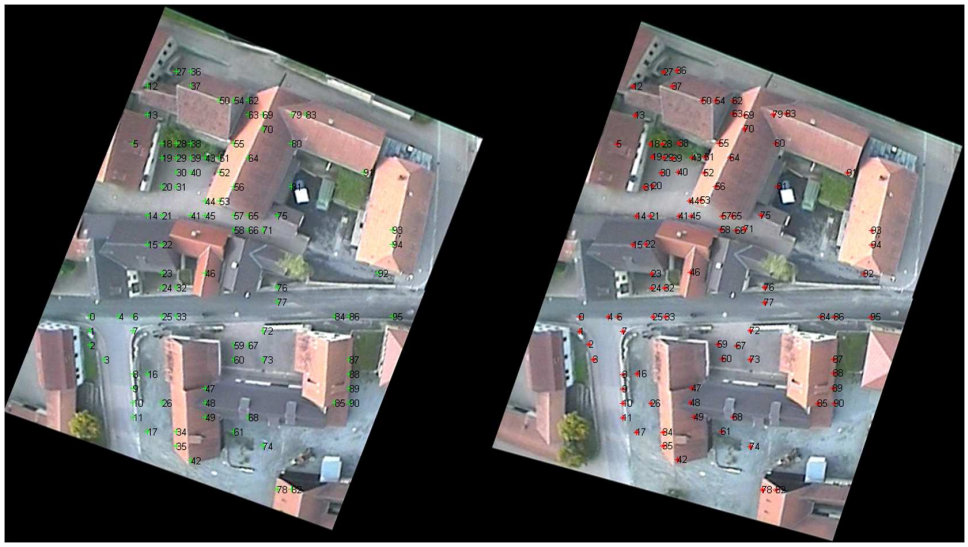

Figure 4.7: Two rectified images of the sequence Bonnland and a point set (marked in green) detected by means of (4.8) in a window of 20×20 pixels in the left image. In the right image, correspondences obtained by the local method are marked in red.

After performing this algorithm for all points, an additional heuristic can be applied in order to reject mismatches. A point with a deviation of disparity values of more than one pixel from all its neighbors is rejected. Here, the neighborhood relation is defined by common edges within the triangulation $\mathcal{T}$. We efficiently apply this procedure once prior to and once after the expansion.

Of course, the process of triangulation and matching can be carried out several times for a narrower matching search space given by (4.10), varying (diminishing) step and (increasing) window sizes until a new, refined disparity map is obtained. An alternative of using a *constrained Delaunay triangulation* (see [119]) with seeded edges stemming from the old, coarser mesh allows evaluating triangles of the coarser mesh with respect to the surface consistency (to be defined in Sec. 4.5) once for all, but has a significant disadvantage of having many long, skinny triangles.

4.4.2 KLT-epipolar and simultaneous tracking policies fur multi-view configurations

Obtaining point correspondences as described in the previous section usually works well for data sets with many fronto-parallel surfaces. In the case of airborne sequences with many

slanted surfaces (which we discuss in Chapter 6), a deviation of one pixel in the image space (disparity) sometimes results in a deviation of several meters in object space. In order to increase accuracy, we consider redundant information from several images by incorporating into the standard KLT-tracking algorithm [94] the knowledge of camera matrices. Because of a strong analogy in the calculations, we will concentrate on the case when the rectification option opt.r of Sec. 4.1 is set to zero in the explanations of this section. For a characteristic point $\mathbf{x}$, we have to compare the intensity distributions of $\mathcal{I}(\mathbf{x})$ and $\mathcal{I}_k(\mathbf{x}_k(d)), k = \{1, ..., K\}$ as in Eq. (4.3). The total error $\breve{\mathbf{c}}$ is composed of $\mathbf{c} = [\mathbf{c}_1, ..., \mathbf{c}_K]$ (the radiometric deviation term) and (optionally) $\mathbf{g} = [g_1, ..., g_K]^T$ (the uncertainties in the camera parameters). Here, the radiometric deviation can be described by differences of gray values since changes of luminance are small in neighboring images of the video sequence. Overall, we have

$$\breve{\mathbf{c}} = [\mathbf{c} \;\; W\mathbf{g}]^T, \; \mathbf{c}_k = \mathcal{I}_k \left(\omega(\mathbf{x}_k(d) + g_k \mathbf{e}_k^{\perp})\right) - \mathcal{I}_0 \left(\omega(\mathbf{x})\right), \tag{4.11}$$

where W is a diagonal weight matrix, ω is a small window around $\mathbf{x}$, $\mathcal{I}_k(\omega(\mathbf{x}_k(d) + g_k \mathbf{e}_k^{\perp})$ is computed by bilinear interpolation and $\mathbf{e}_k^{\perp}$ is the normalized perpendicular component to the epipolar line $\mathbf{e}_k$:

$$\mathbf{e}_k^{\perp} = \begin{bmatrix} e_k^2 \\ -e_k^1 \end{bmatrix} / \sqrt{(e_k^1)^2 + (e_k^2)^2}.$$

The Jacobian of derivatives $\breve{\mathcal{J}}$ is sparse and has the following structure:

$$\breve{\mathcal{J}} = \begin{bmatrix} \mathcal{J} & \bar{\mathcal{J}} \\ 0 & WI \end{bmatrix}, \text{ where } \mathcal{J} = [\mathcal{J}_1, ..., \mathcal{J}_K]^T, \bar{\mathcal{J}} = \begin{bmatrix} \bar{\mathcal{J}}_1 & ... & 0 \\ & ... & \\ 0 & ... & \bar{\mathcal{J}}_K \end{bmatrix},$$

$$\mathcal{J}_k = \frac{\partial \mathbf{c}_k}{\partial d} = [(\mathcal{I}_k)_x \;\; (\mathcal{I}_k)_y] \frac{\partial \mathbf{x}_k(d)}{\partial d}, \; \bar{\mathcal{J}}_k = \frac{\partial \mathbf{c}_k}{\partial g_k} = [(\mathcal{I}_k)_x \;\; (\mathcal{I}_k)_y] \, \mathbf{e}_k^{\perp}, \; I_k = \frac{\partial g_k}{\partial g_k} = 1,$$

w is the number of pixels in the window ω and $\mathbf{x}_k(d)$ of (4.3) is differentiated by the quotient rule. The system of normal equations can be solved for the parameter update $\mathbf{p} = [\Delta d \;\; \Delta \mathbf{g}]^T$, for example, by the Levenberg-Marquardt algorithm (with a small scalar λ and identity matrix I):

$$\left(\breve{\mathcal{J}}^T \breve{\mathcal{J}} + \lambda I\right) \mathbf{p} = -\breve{\mathcal{J}}^T \breve{\mathbf{c}},$$

followed by sparse matrix techniques for linear equation systems. In this work, the uncertainties in camera parameters g_k are not further considered. We thus have $\breve{\mathbf{c}} = \mathbf{c}, \breve{\mathcal{J}} = \mathcal{J}$ and

$$\Delta d = -\mathcal{J}^T \mathbf{c} / \mathcal{J}^T \mathcal{J}(1 + \lambda). \tag{4.12}$$

In this iterative minimization procedure, the initial value of d is refined until a given tolerance in parameter updates is achieved. In our implementation, we considered two policies of optimization: In the *KLT-epipolar* policy, points are sequentially tracked from image to image; pairs of images are optionally rectified and the error function is minimized according to (4.12). Point correspondences are triangulated linearly as described in [61] and rejected if the total reprojection error in pixels exceeds 1. Since $\mathcal{I}_0$ is usually chosen in the middle of the subsequence, the algorithm is modified by *forward* and *backward* tracking. For the second policy, *simultaneous tracking*, we project $\mathbf{x}$ into all images by (4.3) or (4.4) and use Levenberg-Marquardt optimization. If the algorithm converges and the value of $\breve{\mathbf{c}}$ of (4.11) without considering camera uncertainties lies below a fixed threshold $\varepsilon_{\max}$, the point is said to be tracked reliably and its 3D coordinates are computed from the depth value by means of (4.2).

4.5 Multi-view dense matching using triangular meshes

The task of dense matching is to assign a depth to each pixel of the reference image $\mathcal{I}_0$. The algorithms of the previous section cannot be applied to every pixel because of susceptibility to converge to local minima for pixels in areas of homogeneous texture and because of a high computational cost of a non-linear iterative minimization algorithm. Therefore, on the one hand, the values of unknown parameters must be discretized; for the binocular case, the discretization labels are given in the natural way by the integer disparity values. On the other hand, smoothness assumptions must be used in order to propagate the information from already reconstructed points or points where the correct depth value can be reliably obtained to those textureless regions.

The initialization of the depth map with $\mathcal{D}_\mathcal{T}$ from Sec. 4.3 can be used as a soft constraint in order to bias the depth values of the pixels – especially in areas of weak texture – to those resulting from triangular interpolation. To do this, we introduce a *triangulation-based* smoothness term and a process of *evaluation of triangles*. In Sec. 4.5.1, we will use $\mathcal{D}_\mathcal{T}$ as initialization for two non-local algorithms, namely the global algorithm of graph cuts with α-expansions [81] and semi-global optimization used by [67] with Mutual Information as cost function. Such depth maps obtained from pairs of images can be fused into the *median depth map* described in Sec. 4.5.2, which has the advantage of a much lower percentage of outliers and points with non-assigned depth values. Since calculation of median depth maps is computationally intensive, a framework of local and global simultaneous computation of depth maps will be presented in Sec. 4.5.3. Finally, we present in Sec. 4.5.4 an approach for automatic selection of the smoothness parameter λ, which as we have learned in Sec. 2.3, represents a trade-off between the properties of the data given a scene (photo-consistence assumptions) and hypothesized properties of the scene (piece-wise smoothness assumptions).

4.5.1 Binocular configuration

Triangulation-based smoothing

As previously indicated, the evaluation of pixel costs is carried out by means of one of the cost functions $c(\mathbf{x}, j) = E_{data}(\mathbf{x}, j)$ of Sec. 2.2 for every integer value of disparity. A significant difference of this approach with many state-of-the-art approaches is that we extensively use a large point set that is (after applying tracking routines described in Sec. 4.4) homogeneously distributed in $\mathcal{I}_0$. We assume that the non-occluded parts of the scene can be piecewise approximated by triangles. The point is that, if a correct evaluation is made about which triangles are nearly consistent with the surface and which are not, we will not only be able to avoid mismatches in areas of repetitive patterns of textures and homogeneous texture, but also be able to obtain depth values of *all points* within these triangles with subpixel accuracy. This subpixel calculation, performed in order to avoid discretization errors (see Fig. 4.8, left) actually does not depend on the choice of the cost function (see [128]) and it replaces segmentation of images as in [14, 68, 77, 87].

In [28], the local smoothness term

$$E_T(\mathbf{x}, j) = A(\mathbf{x}, T) D(j, T, \mathbf{x}) \tag{4.13}$$

is introduced. Here D can be practically any scalar nondecreasing function in terms of $|j - j_{T,\mathbf{x}}|$. The weight function $A(\mathbf{x}, T)$ should be zero outside the convex hull, reflect the reliability for the coordinates of points at the vertices of a triangle T and become smaller in its interior (as, for instance, in Fig. 4.8, middle and right). One possible choice, followed up in this paper, is

$$A(\mathbf{x}, T) = A_0 \exp\left(-\frac{g(\mathbf{x}, T)}{\sigma}\right),\ D(j, T, \mathbf{x}) = -1 + \min\left(\frac{|j_\mathbf{x} - j_{T,\mathbf{x}}|}{j_0}, 1\right), \tag{4.14}$$

where $\mathbf{x} \in T$, the amplitude A_0 (which in the future will be denoted by A) and j_0 are two non-negative constants and the descent parameter $\sigma \in [0; \infty]$. By $g(\mathbf{x}, T)$, we denote the minimum distance from $\mathbf{x}$ to the vertices of T. For j_0, the value 2 is a reasonable choice. It is clear that for small values of σ, only the depth values for characteristic points are made unlikely to change (which can be good when such points are provided by other sources – such as LIDAR-data – and thus possibly lie in the weakly textured areas). On the other hand, for $\sigma \to \infty$, the whole convex hull $\bigcup T \in \mathcal{T}$ will be affected:

$$A(\mathbf{x}, T) = \begin{cases} A & \text{if } g(\mathbf{x}, T) < 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } \sigma = 0,$$

and

$$A(\mathbf{x}, T) = \begin{cases} A & \text{if } \mathbf{x} \in \bigcup T \in \mathcal{T} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } \sigma = \infty.$$

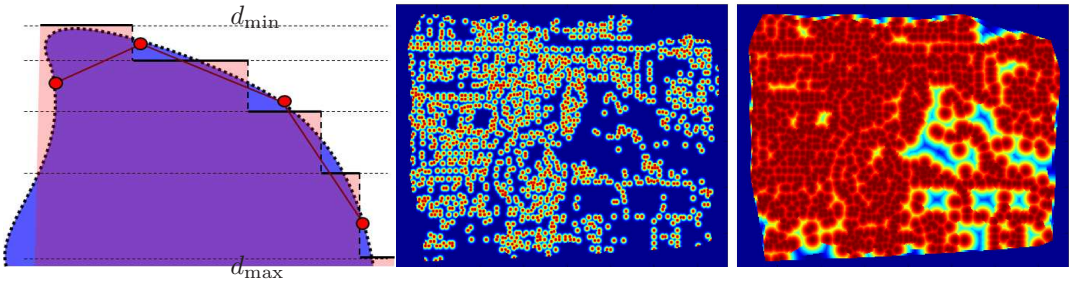

Figure 4.8: Left: Discretization of depth labels deteriorates the visual quality of the dense reconstruction even in the case of error-free matching. The problem can be solved by considering triangular meshes from 3D points already obtained rather than by increasing the number of labels, because, in the latter case, mismatches appear due to limited resolution, the smoothness term of (4.23) tends to lose its sense and computation cost increases dramatically. Middle and right: Weights $A(\mathbf{x}, T)$ from (4.14) propagated from already available points with a small/large value of σ (on the left/right, respectively) for the reference image of sequence *Tsukuba* (see [115]).

In addition to the parameters A and σ, a third triangulation-based parameter $\gamma \in [0; 1]$ is introduced in [28]. If the percentage of pixels consistent with the surface within a triangle exceeds γ, then all pixels $\mathbf{y}$ of such triangles are assigned the value $d_{T,\mathbf{y}}$. The definition of a pixel $\mathbf{x}$ consistent with the surface is given by the ratio

$$r(\mathbf{x}) = \frac{c_0(\mathbf{x})}{\min\left\{c\left([j_{T,\mathbf{x}}]\right), c\left([j_{T,\mathbf{x}}]+1\right)\right\}}, \tag{4.15}$$

where $c_0(\mathbf{x}) = \min_j(c(\mathbf{x}, j))$ is the best cost value and $[j_{T,\mathbf{x}}]$ is the "floor value" of $j_{T,\mathbf{x}}$ (the largest integer smaller than $j_{T,\mathbf{x}}$). Point $\mathbf{x}$ is said to be consistent with the surface if $r(\mathbf{x}) = 1$ for a global algorithm and $r(\mathbf{x}) > 0.8$ for a local algorithm.

Using similarity information of triangles in RGB-images

The influence of parameters A, σ and γ helps to overwrite, at different stages of the algorithm, the disparity values of a set of pixels with those stemming from triangular interpolation. The performance of this approach depends directly on the quality of the triangular meshes. In the case of color images $\mathcal{I}_1, \mathcal{I}_2$, the authors of [29] propose a similarity analysis of triangles based on color information and histogram evaluations: Each color contains values from 0

to 255 and thus each color histogram has b bins with a bin size of $256/b$. Let the number of pixels in a triangle be N. In order to obtain the probability of this distribution and to make it independent of the size of the triangle, we obtain for the l^{th} bin of the *normalized histogram*

$$H_T(l) = \frac{1}{N} \cdot \# \left\{ \mathbf{p} \,\middle|\, \mathbf{p} \in T \text{ and } \frac{256 \cdot l}{b} \leq \mathcal{I}_0(\mathbf{p}) < \frac{256 \cdot (l+1)}{b} \right\}.$$

The three histograms H_T^R, H_T^G, H_T^B represent the color distribution of T. It is also useful to split big, inhomogeneous triangles that are inconsistent with the surface into smaller ones. To perform splitting, characteristic edges [30] are found in every candidate triangle and saved in the form of a binary image $G(\mathbf{p})$. To find the line with maximum support, the radon transformation [37] is applied to $G(\mathbf{p})$:

$$\breve{G}(u,\varphi) = R\{G(\mathbf{p})\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\mathbf{p})\delta(\mathbf{p}^T\mathbf{e}_\varphi - u)d\mathbf{p} \text{ where } \delta(x) = \begin{cases} \infty & x = 0 \\ 0 & \text{otherwise} \end{cases}$$

is the Dirac delta function and line parameters $\mathbf{p}^T\mathbf{e}_\varphi - u$, where $\mathbf{e}_\varphi = (\cos\varphi, \sin\varphi)^T$ is the normal vector and u the distance to the origin. The strongest edge in the triangle is found when the maximum of $\breve{G}(u,\varphi)$ exceeds a certain threshold for the minimum line support. This line intersects the edges of T in two points. We disregard intersection points too close to a vertex of T. If new points are found, the original triangle is split in two or three smaller triangles. These new, smaller triangles consider the edges in the image.

Next, the similarity of two neighboring triangles has to be calculated by means of the color distribution. There are a lot of different approaches for measuring the distance between histograms, see [31]. We define the distance between two neighboring triangles T_1 and T_2 as follows:

$$\text{dst}(T_1,T_2) = w_R \cdot d\left(H_{T_1}^R, H_{T_2}^R\right) + w_G \cdot d\left(H_{T_1}^G, H_{T_2}^G\right) + w_B \cdot d\left(H_{T_1}^B, H_{T_2}^B\right) \tag{4.16}$$

where w_R, w_G, w_B are weights for the colors that are all set to be $1/3$ in our method. The distance d between two histograms in (4.16) is the SAD of their bins. There are two possible ways to define neighboring relations on a set of triangles: two triangles can be declared neighbors if they either share one or two common vertices (i. e. a common edge in this latter case). The value of $\text{dst}(T_1,T_2)$ is set to infinity if T_1 and T_2 are not neighbors.

In the last step, disparity values in the vertexes of triangles that are inconsistent with the surface are corrected. For such a triangle T_1, another triangle

$$T_2 = \arg\min_{T \in \mathcal{T}_0} \text{dst}(T_1, T)$$

was defined in [29]. Here, $\mathcal{T}_0$ denotes the set of triangles consistent with the surface. If $\text{area}(T_2) > 30$ pixels and $\text{dst}(T_1,T) < 2$, then T_1 and T_2 are likely to belong to the same (planar) region of the surface and therefore the disparities of pixels in T_1 are recomputed with $\mathbf{v}_{T_2}$ according to **Result 2**. The more reliable, though time-consuming approach, not followed in [29], consists of expanding the already precomputed cost function $E_{data}(\mathbf{x})$ by the recalculated triangle-based term $E_T' = A(\mathbf{x},T_2)D(j,T_1,\mathbf{x})$ from (4.13) and (4.14).

Refinement with global and semi-global optimization algorithms

The values of the function $c(\mathbf{x}) = E_{data}(\mathbf{x},d) + E_T(\mathbf{x},d)$, computed for each pixel and each disparity value, can be stored in a $S \times M$ matrix $\mathcal{A}$ where S is the number of disparity labels and M is the number of pixels. The result $\mathcal{D}_{loc}$ of a local algorithm assigns to the pixel $\mathbf{x}_i$ a

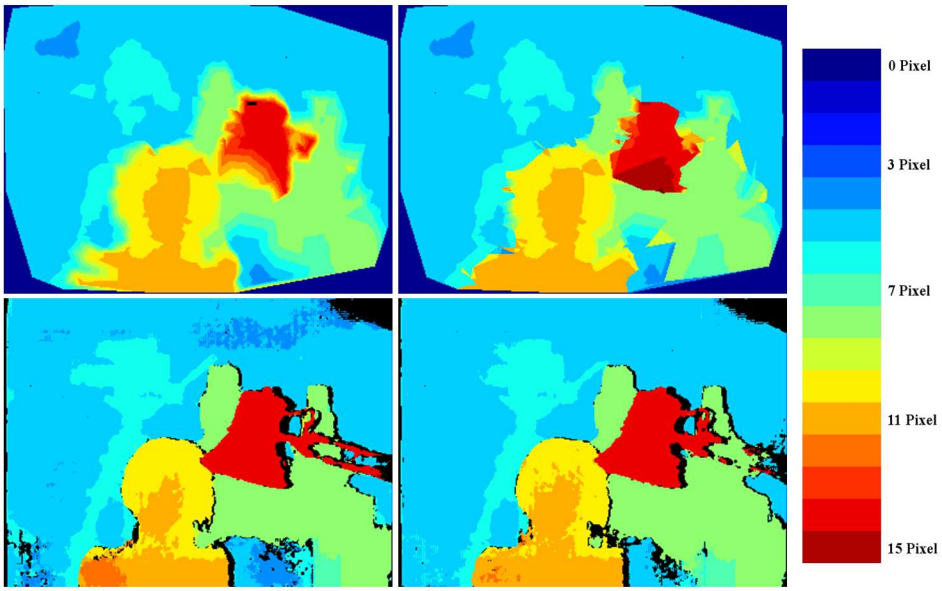

Figure 4.9: Top row, left: Initialization of the disparity map created from the triangular mesh. Top row, right: Result of correction of triangles as suggested in [29] for a pair of images from the sequence *Tsukuba*. Bottom row, left and right: Results of semi-global estimation of the disparity map without and with initialization of the disparity map, respectively. Right: Color scale representing different disparity values.

label corresponding to the minimum value within a column i of $\mathcal{A}$ (followed by γ-smoothing of surface-consistent triangles). Two possibilities are now opened up: to use either $\mathcal{A}$ or $\mathcal{D}_{loc}$ (or alternatively $\mathcal{D}_{\mathcal{T}}$) as an initialization of a (semi-)global optimization algorithm with one of the smoothness energy terms of Sec. 2.3. Before we go into the details of these two kinds of optimization, we consider two examples that justify each of two approaches. An example of advantages of initialization with $\mathcal{D}_{\mathcal{T}}$ is in the case of unclear luminance relations (such that $\mathcal{A}$ cannot be rendered). By calculating intensity correspondences with $\mathcal{D}_{\mathcal{T}}$ (see [29], Sec. 2.4), one can determine values for the *mutual information* matching table $MI(m, n)$ of Eq. (2.8) and does not have to consider image pyramids. This helps save computing time. On the other hand, suppose we have several very exact (e. g., LIDAR) 3D points. In this case, we use a very high value of A and a low value of σ in (4.14) in order to fix the disparity values of the ground control points in $\mathcal{A}$ and propagate these values to neighboring points using smoothness terms.

As explained in Sec. 3.1.3, the main feature of the algorithm of [81] is an α-expansion that expects a (depth) image $\mathcal{D}$ as input. The output $\mathcal{D}'$ is either identical with $\mathcal{D}$ or some pixels of $\mathcal{D}'$ are assigned the value α. In other words, if we have a good initialization, the energy computed at the beginning already takes on a large negative value and so, on average, fewer expansion moves need to be taken. This allows reducing computing time. On the other hand, initializing the semi-global optimization with a result of $\mathcal{D}_{\mathcal{T}}$ allows omitting image pyramids without significant visible and quantitative adverse affects on the results, as illustrated in Fig. 4.9, bottom.

The second alternative, namely, to consider $\mathcal{A}$, works in a slightly different way and will be covered for the multi-view case in Sec. 4.5.3 on the examples of dynamic programming and semi-global optimization.

4.5.2 Median-based depth estimation

A depth map produced in the previous section usually has several outliers and artifacts, especially in areas of reflections, occlusions, and homogeneously textured regions. To increase accuracy, it is necessary to use all available information from several images and several disparity maps obtained from $\mathcal{I}_0$ and $\mathcal{I}_k$ $(k = 1, ..., K)$.

One can ask why it makes sense to compute pairwise depth maps from pairs of frames in a subsequence of the given video sequence if a multi-view reconstruction algorithm (presented in Sec. 4.5.3 below) that can handle all images simultaneously is available. The answer is that the method described in this section has a possibility of self-control since, for pixels without reliable depth, undefined values are likely to occur while the algorithm of Sec. 4.5.3 has the advantage of being fast although, since geometric control is not given (that is, the algorithm always delivers *some* depth map), it is possible to have some outliers because of radiometric irregularities in the reference image, low accuracy in the position of cameras, etc. We can compare the ideas behind the algorithms of Sec 4.5.2 and Sec. 4.5.3 with the epipolar and simultaneous tracking of Sec. 4.4. The second important point is that the majority of the (semi)-global state-of-the-art methods available online (such as the graph-cuts method, belief propagation, etc.) works only for rectified image pairs. If we search for a certain advantage of these algorithms and are interested in obtaining a stable result with few outliers, we must be able to work with several disparity maps from a set of images rather than with an oriented subsequence (with external data provided by camera matrices). The situation covered in this section is schematically visualized by Fig. 4.10, left.

The algorithm starts by computing depth values $d_{k,\mathbf{x}} = d_k$ for a pixel $\mathbf{x} \in \mathcal{I}_0$ from disparity maps between $\mathcal{I}_0$ and $\mathcal{I}_k$ obtained in the previous section and use the chain of equations (4.6) $\mapsto$ (4.2) $\mapsto$ (4.1) (compare Fig. 4.2). But which of these values d_k should be chosen? Clearly, if a cluster with several values of d_k can be identified, we can assign to $d_{\mathbf{x}}$ the median of these values. In other words,

$$\bar{d} = d_{\mathbf{x}} = \text{median}_k \{d_k \,||d_k - d| < \varepsilon\} \tag{4.17}$$

for some positive ε. Conversely, if for example, $|d_k - d'_k| > \varepsilon$ for all $1 \leq k < k' \leq K$ and no prior information (such as the confidence of disparity maps or information about whether $\mathbf{x}$ is consistent with the surface) about the depth value at $\mathbf{x}$ is available, the depth at $\mathbf{x}$ is left undefined.

Equation (4.17) is recursive. In order to identify the set of values in a cluster, one can iteratively approximate $\bar{d}$ by the weighted average

$$\bar{d} = \frac{1}{W} \sum_k d_k w_k \quad \text{with} \quad W = \sum_k w_k \tag{4.18}$$

and with initial weights $w_k = 1$ if d_k is not occluded and 0 otherwise. In the next iteration, we set $w_k = w_k(d_k - \bar{d})^{-\beta}$ where β is a positive scalar ($\beta = 2$ is used for our applications). After the last iteration, we compute d from the inliers among the values of d_k by (4.17) and accept this value when the number of inliers is not smaller than $\max(K/3)$. Several remarks can be made here:

1 If $\mathcal{D}_{\mathcal{T}}(\mathbf{x})$ is available, it can be used as an additional observation in Eq. (4.17) and (4.18). The counter is now $K + 1$ and the initial weight for the triangular term is larger than 1 since the probability that the triangle incident with $\mathbf{x}$ is consistent with the surface is rather high.

2 Since the points in the background are obtained with lower accuracy than those in the foreground, one replaces the ε on the right of (4.17) by $\varepsilon\bar{d}$. Note that this right-hand side only influences the results of the final iteration.

3 A total of 3 to 5 iterations can be used in the algorithm. Because of the structure of (4.18), the loop over the pixels can be avoided, so the computation of weights and $\bar{d}$-values can proceed simultaneously. Therefore, the time for computing the median depth map is comparing with the time for computing depth maps.

The last remark concerns the choice of initial weights. Especially in the case of a low number of views, it makes sense not to set all $w_k = 1$, but to obtain, for one single pixel $\mathbf{x}$, the confidence of the depth value at $\mathbf{x}$. The confidence is expected to be high if the cost function has a single sharp minimum and low if there are several local minima (in other words, there are several plausible possibilities to match $\mathbf{x}$ in the corresponding epipolar line). The confidence map is calculated in a manner similar to that used in [111]:

$$\mathcal{C}_0(\mathbf{x}) = \left(\sum_{\tilde{d} \neq d_{\mathbf{x}}} \exp \left\{ - \frac{\min \left(c(\mathbf{x}, d_{\mathbf{x}}) - c(\mathbf{x}, \tilde{d}), 0 \right)^2}{\sigma^2} \right\} \right)^{-1}, \tag{4.19}$$

where σ is an empirically determined constant. We use the confidence function if the number of available views is low and select the match of highest confidence.

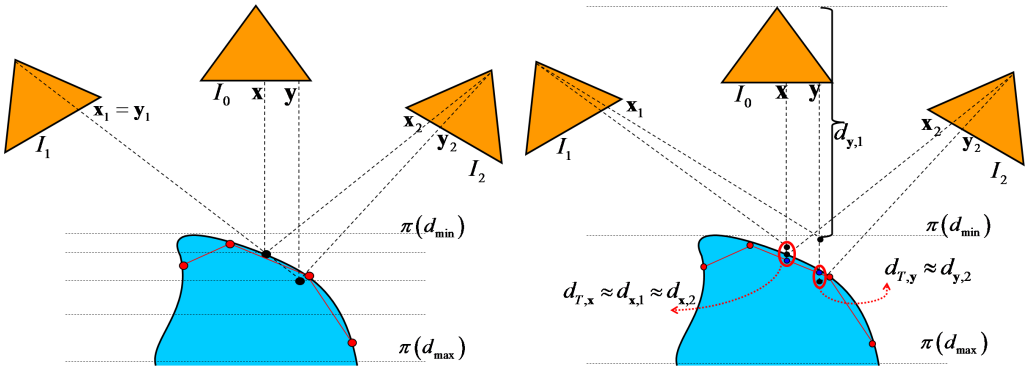

Figure 4.10: Left: Median-based computation of depth maps. In order to find $d_{\mathbf{x}}$, one can take into account depth values $d_{\mathbf{x},1}, d_{\mathbf{x},2}$ resulting from images $\mathcal{I}_1, \mathcal{I}_2$ and also $d_{T,\mathbf{x}}$ (since these values lie in a cluster specified by the ellipse on the left). For the point $\mathbf{y}$, only $d_{\mathbf{y},2}, d_{T,\mathbf{y}}$ must be taken into account and $d_{\mathbf{y},1}$ is an outlier. Right: Schematic visualization of simultaneous multi-view dense estimation. Pixels have to be assigned labels using cost and smoothness penalty functions. The triangulation from the enriched set of points is shown by red circles and lines. A forbidden configuration of interactions $\{(\mathbf{x}, \mathbf{x}_1), (\mathbf{y}, \mathbf{y}_1)\}$ should be excluded either by adding an occlusion term as in (2.9) or by modification of the aggregation function.

4.5.3 Fast simultaneous computation of depth maps for multi-view configurations

Discretization

First, the equations (4.3) and (4.4) of Sec. 4.1 must be modified by discretizing d or t into labels d_j and t_j, respectively; here, $j = 0, ..., S$ and $S+1$ is the number of bins (labels). The discretization chosen is inverse-linear,

$$d_j = \frac{S}{(S-j)/d_{\min} + j/d_{\max}}$$

(schematically visualized in Fig. 4.10, right), which is more suitable than the linear one, namely,

$$d_j = ((S-j)d_{\min} + jd_{\max})/S,$$

because, in the inverse-linear case, the projections of the corresponding 3D points lie nearly equidistant at epipolar lines and so the decrease of the resolution is treated in a more natural way. The resolution of depth for points near the camera is then higher (and so is the accuracy of the depth computation) than in the background further away from the camera. The number of labels corresponds to the length (in pixels) of the longest epipolar line after all available points are projected into images $\mathcal{I}_1, ..., \mathcal{I}_K$ by means of Eq. (4.3) using values $d_{\min}, d_{\max}$ (or, respectively, Eq. (4.4) and values of $t_{\min}, t_{\max}$).

Choice of data and aggregation function

Analogously to binocular configurations, cost functions for each depth label and each pixel must be computed. As a default cost function, the truncated SAD (2.4) is used. Experiments were also carried out for NCC as in (2.6) and MI as in (2.8). In contrast to the situation with the image pair rectified to epipolar geometry, where the cost evaluation proceeds by fast convolution methods between windows of type (4.10), we need here also the inner loop of depth values (labels), which presupposes extracting quadratic windows around reprojected (e. g. by (4.4)) points by means of bilinear interpolation as in (3.5) (actually, bilinear interpolation is performed if and only if the option opt.i is activated; otherwise $\mathbf{x}_k(d)$ is determined by rounding procedure). Between this inner loop over depth values and the outer loop over pixels, there is a loop over interactions, i. e. which pairs of windows must be compared to each other. Since there are $K(K-1)/2$ possible kinds of interactions $i = \langle \cdot, \cdot \rangle$ and we want our algorithm to be linear in the number of views, a subset of i must be selected. One possibility, followed up in the current implementation, is to aggregate costs between the reference image $\mathcal{I}_0$ and other images. This choice differs from [79] which proposes to use neighboring images. The latter approach, we admit, could help us to treat all images symmetrically and avoid error resulting from radiometric irregularities in the reference image (reflections, small moving objects, dead pixels, etc.), but we decided, similar to what was done in [111], to compute costs from the reference image to other images, because in doing so, a higher value of S and therefore a higher depth accuracy can be obtained. In [111], the minimum of sums of data cost functions on the left and on the right of the reference image has been chosen. An occlusion term, important in [79], can be omitted in the majority of practical situations if the choice of the *cost aggregation function* is robust against occlusions, in which case not every pixel $\mathbf{x} \in \mathcal{I}_0$ must be seen in all images $\mathcal{I}_1, ..., \mathcal{I}_k$, but, at the same time $\mathbf{x}$ is encouraged to be observed in a large number of images (see Fig. 4.10, right). For example, in [22], where care was taken to exclude all triangles inconsistent with the surface, it was enough to consider the sum of costs c_k over k. For a more sophisticated choice of aggregation function, we denote by $K(\varepsilon_{\max})$ the number of interactions (of $\mathbf{x}$) where the cost function does not exceed a constant $\varepsilon_{\max}$. Now, for example, the aggregation function "average error per interaction"

$$E_{data}(\mathbf{x}) = \frac{\sum_k \{c_k | c_k \leq \varepsilon_{\max}\}}{K(\varepsilon_{\max})}$$

also tends to be small if only for a few images c_k is small at d, which is of course, unstable. Therefore we used positive constants $b, \varepsilon_{\max}$ and K_0 to increase the denominator for large $K(\varepsilon_{\max})$ and the aggregation function chosen in this work was

$$E_{data}(\mathbf{x}, d) = \begin{cases} \dfrac{\sum_k \{c_k | c_k \leq \varepsilon_{\max}\}}{(1+b)\,(K(\varepsilon_{\max}) - K_0) + 1} & \text{if } K(\varepsilon_{\max}) > K_0 \\ +\infty & \text{otherwise.} \end{cases} \tag{4.20}$$

The concept of dense pixel matching is explained in Alg. 8.1 of the Appendix.

Considering triangular meshes

Equations (4.13) and (4.14) for the triangulation-based term can, as in Sec. 4.5.1, be written in terms of depth instead of disparity. We again use the triangulation-based smoothness term

$$E_T(\mathbf{x}, d) = A(\mathbf{x}, T)D(d, \mathbf{x}, T), \text{ where} \tag{4.21}$$

$$A(\mathbf{x}, T) = A\exp\left(-\frac{g(\mathbf{x}, T)}{\sigma}\right),\ D = -1 + \min\left(\frac{|d_{\mathbf{x}} - d_{T,\mathbf{x}}|}{d_0}, 1\right) \tag{4.22}$$

with constants A, σ, d_0 and function $g(\mathbf{x}, T)$ defined analogously to (4.14).

The values of the function $E_{data}(\mathbf{x}, d) + E_T(\mathbf{x}, d)$, computed for each pixel and each depth level, are again stored in a $S \times M$ matrix $\mathcal{A}$. Similar to the binocular case, the local algorithm, in order to obtain $d_{\mathbf{x}_i}$ compares the lowest cost within the column i (that is, $j = \arg\min_{j'} \mathcal{A}(j', i)$) with costs at rounded $d_{T,\mathbf{x}_i}$ and assigns $d_{\mathbf{x}_i} = d_{T,\mathbf{x}_i}$ if T is consistent with the surface and d_j otherwise. Furthermore, almost any algorithm for non-local optimization mentioned in Sec. 3.1.3 can now be applied for the matrix thus obtained. We show two examples of the non-local optimization in the next section. After a depth level $d_{\mathbf{x}}$ for a pixel $\mathbf{x}$ (a result of a local or global algorithm) has been retrieved, we can compute cost functions at $d_{\mathbf{x}}$ and $d_{T,\mathbf{x}}$; if the ratio $r(\mathbf{x})$ as in (4.15) is below a threshold, the pixel $\mathbf{x}$ is marked as consistent with the surface. The percentage of pixels consistent with the surface allows a decision about triangles: if the percentage exceeds a constant scalar γ, all pixels $\mathbf{y}$ of such triangles are assigned the value $d_{T,\mathbf{y}}$. The influence of the parameters A, σ and γ will be evaluated in Sec. 6.3 along with other items.

Two examples of non-local optimization

As two examples of non-local optimization, the 1D optimization algorithm of dynamic programming [10] and semi-global optimization as in [67] were considered. For both approaches, E_{smooth} is chosen as in [67] (and, as in Eq. (2.11) on p. 24, d_0=1):

$$E_{smooth}(\mathbf{x}, j) = \lambda_1 \cdot N_{\mathbf{x}}(1) + \lambda_2 \cdot \sum_{j=2}^{\infty} N_{\mathbf{x}}(j), \tag{4.23}$$

where λ_1 and λ_2 with $\lambda_1 \leq \lambda_2$ are penalties for depth discontinuities and $N_{\mathbf{x}}(j)$ is the number of pixels $\mathbf{y}$ in the 4-neighborhood of $\mathbf{x}$ for which the absolute difference of depth/disparity values at $\mathbf{x}$ and $\mathbf{y}$ is equal to j. This choice of E_{smooth} is reasonable, because penalty terms monotonically increasing with differences of depth levels result in over-smoothing occlusions.

In the case of dynamic programming considered for an (epipolar[3]) line with M pixels, the data cost matrix $\mathcal{A}_{j,i}$ is denoted, as done previously in Sec. 2.2, by $[c(1, j), ...c(M, j)]$ for each value $j = 1, ..., S$ and the smoothness cost matrix with entries is denoted by $c_s(j_1, j_2), ...,$ $c_s(j_{M-1}, j_M)$. The smoothness term can also depend on the intensity levels of relevant pixels (see (2.12)) and should be denoted by $c_s(j_{M-1}, j_M, \mathcal{I}_0(M-1), \mathcal{I}_0(M))$. However, this slight misuse of notation does not lead to misunderstanding and is, therefore, not critical. The task is to minimize

$$\sum_{i=1}^{M} c(i, j_i) + \sum_{i=1}^{M-1} c_s(j_{i-1}, j_i) = \sum_{i=1}^{M-1} (c(i, j_i) + c_s(j_{i-1}, j_i)) + c(M, j)$$

[3]Originally, dynamic programming is used for a rectified stereo pair, so that in our applications, epipolar lines coincide with horizontal (scan)lines if and only if opt.r = 1

over all S^M possible configurations of j_i. This is carried out by computing and storing the best path $P(i,j)$ from 1 to i for each value of j_{i+1}, as explained in the Alg. 8.2.

The complexity of Alg. 8.2 is actually $O(MS^2)$ (instead of the S^M complexity of the brute-force procedure which considers every configuration), because computing $C(j)$ by minimization over j' is itself an $O(S)$ procedure. By a suitable choice of smoothness function, one can achieve a complexity of $O(MS)$. Such a smoothness function λ must depend as little as possible on j (although dependence on $\mathcal{I}$, as in (2.12), is not a problem). For example, in order to compute $C_1(j)$ with a disparity term given by (2.10) or (2.12), we need only to compare $C(j)$ and $C(P(j)) + \lambda(i))$. For the smoothness term mentioned in (4.23), four values of must be compared (see (4.24)). The generalization of the Alg. 8.2 for (2.15) (in which the smoothness term involves j_i, j_{i+1}, and j_{i+2}) is straightforward. The difference with Alg. 8.2 is just that we need to compute $C_1(j+2)$ in order to know the best path $P(i,j)$. For example in order to know the optimal choice of the label j_1 for every value of j_3, we must compute

$$\min_{j_1} \left(c(1,j_1) + c(2,j_2) + \lambda_1 |j_1 + j_3 - 2j_2| \right),$$

for every j_2 and j_3, a procedure of $O(MS^3)$ complexity. Also in this case, of course, the complexity can be reduced for special kinds of depth terms.

As for the semi-global optimization algorithm, the NP-hard 2D problem (2.9) was solved by approximating the term E_{smooth}. As stated in [67], at least eight paths (two horizontal, two vertical and four diagonal) are necessary to provide good coverage of $\mathcal{I}_0$. Throughout our experiments, up to 16 paths are used. A global accumulation of all possible paths is replaced by paths emanating from each pixel along a straight line. Suppose we have a pixel $\mathbf{x}$ and a path direction $\mathbf{r}$ such that the previous pixel $\mathbf{x} - \mathbf{r}$ is denoted by $\mathbf{y}$. With (4.23), the path cost at $\mathbf{x}$ at depth label j in the direction $\mathbf{r}$ is recursively defined by

$$L'_{\mathbf{r}}(\mathbf{x},j) = c(\mathbf{x},j) + \min \left[L'_{\mathbf{r}}(\mathbf{y},j), L'_{\mathbf{r}}(\mathbf{y},j \pm 1) + \lambda_1, \min_i L'_{\mathbf{r}}(\mathbf{y},i) + \lambda_2 \right]. \tag{4.24}$$

This recursive formula is initialized by corresponding values of $\mathcal{A}$ at the beginning of all paths. Because the value $L'_{\mathbf{r}}(\mathbf{x},j)$ always increases as the path is traversed, precautions must be taken to bound L. Thus, (4.24) is extended to

$$L_{\mathbf{r}}(\mathbf{x},j) = L'_{\mathbf{r}}(\mathbf{x},j) - \min_{j'} L_{\mathbf{r}}(\mathbf{y},j'). \tag{4.25}$$

Since $\min_{j'} L_{\mathbf{r}}(\mathbf{y},j')$ is constant for all j, the position of the minimum-cost depth does not change and $L_{\mathbf{r}}$ is bounded by $L_{\mathbf{r}} \leq \varepsilon_{max} + \lambda_2$. To compute the costs for a depth, the paths for all computed directions $\mathbf{r}$ are summed up to

$$C(\mathbf{x},j) = \sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{x},j).$$

The depth label $d_{\mathbf{x}}$ is then chosen as the label that yields the lowest overall cost: $\arg\min_j C(\mathbf{x},j)$. Since 16 paths are used in our experiments, the upper limit of C is $C < 16(\varepsilon_{max} + \lambda_2)$. By scaling the entries of the data cost matrix so that both ε_{max} and λ_2 are bounded by 2048, the size of C can be limited to 16 bits and thus a 16-bit integer vector is used throughout the computations. At the last step, outliers (which can sporadically emerge in the regions between the paths) are eliminated by means of a median filter. The subpixel calculation can proceed by fitting a correlation parabola to the values of the cost function, as we explained at the end of Sec. 4.4.1.

The semi-global optimization algorithm also has complexity $O(MS)$ (or, to be exact, $O(MSr)$ where r is the number of paths) for our special choice of cost function. In general,

its complexity is $O(MS^2r)$ (since the second summand of (4.24) is, in the general case, $\min[L(\mathbf{y}, j') + c_s(j, j', \mathbf{x}, \mathbf{y})]$ over j'). Applying semi-global optimization helps eliminate streaking artifacts without significant increase in computing time. It will, therefore, be our default method for the reconstruction pipeline.

4.5.4 Choice of smoothness parameters

As for the choice of smoothness parameters, the results presented in the next chapter show that the difference $\lambda_2 - \lambda_1$ should be bounded away from zero, since otherwise the algorithm prefers one big jump of the depth to its slow, continuous change that is characteristic for smooth surfaces. As a result, the depth maps become too noisy. On the other hand, if $\lambda_2 \gg \lambda_1$, the results easily become over-smoothed near occlusions and the deviations of depth in these areas become, consequently, very high. The best results were achieved for the ratio $\lambda_2/\lambda_1 = 2$ to 3. The choice of λ_1 is not trivial, but also not critical, since it is typical for global algorithms to produce results of comparable quality for quite a wide range of smoothness parameters. Due to equations (4.1) and (4.23), however, it is clear that λ_1 must not depend on image size while its order of magnitude must depend on the *differences of entries* in the data cost term.

The following strategy is applied: after the local algorithm is performed and a label j is assigned to a pixel $\mathbf{x} \in \mathcal{I}_0$, we calculate the term

$$\begin{aligned} \mathcal{C}_1(\mathbf{x}) = |c(\mathbf{x}, j) - c(\mathbf{x}, j+1)| + |c(\mathbf{x}, j) - c(\mathbf{x}, j-1)| = \\ \sum_{|\tilde{j}-j|=1} |c(\mathbf{x}, j) - c(\mathbf{x}, \tilde{j})| \end{aligned} \tag{4.26}$$

in order to estimate, quite rigorously, the confidence of $\mathbf{x}$. This quantity $\mathcal{C}_1(\mathbf{x})$ measures how well the cost function at d_j outperforms the cost functions of the previous and following labels, so that the depth value of $\mathbf{x}$ can be changed (oversmoothed) by λ_1. The quantity $\mathcal{C}_1(\mathbf{x})$ is a special case of

$$\mathcal{C}_2(\mathbf{x}) = \frac{1}{s} \sum_{\tilde{j} \neq j} |c(\mathbf{x}, j) - c(\mathbf{x}, \tilde{j})|, \tag{4.27}$$

and a simplification of the term $\mathcal{C}(\mathbf{x})$ of (4.19). Here again, we denote by $c(\mathbf{x}, j)$ the value of $E_{data}(\mathbf{x}, j) + E_T(\mathbf{x}, j)$. For illustration of this, see Fig. 4.11.

Now let us assume that typically not more than 10 to 20% of all points are characteristic enough that the depth can be estimated with a precision of one (depth or disparity) label (since the vast majority of points lies in areas of rather weak texture). Then it is sufficient to take a value of λ_1 corresponding to a quantile between the 80^{th} and 90^{th} quantiles of the histogram of $\{\mathcal{C}_1(\mathbf{x}) | \mathbf{x} \in \mathcal{I}_0\}$. Due to discretization effects, one could consider a lower quantile value $\mathcal{C}_2(\mathbf{x})$ of (4.27).

To explain the reason for our assumption, we go one step further and take into account pixels corresponding to smooth surfaces in object space. These are the pixels whose depth values we should be able to change by applying the smoothness term and which often lie in homogeneously textured areas. The question how many pixels we must be able to over-smooth is equivalent to the question how many pixels lie in homogeneous, *topologically connected* regions. This is the reason why the smoothness parameters the data set *Tsukuba* will turn out to be somehow lower than for data sets that are typical for our applications: there are not that many homogeneous regions in the reference image.

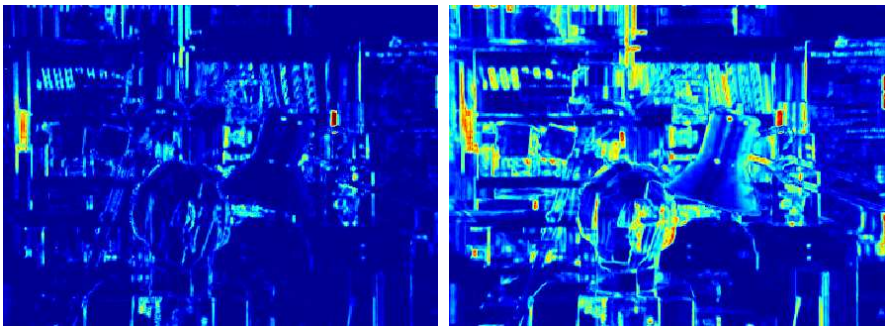

Figure 4.11: Confidence maps $\mathcal{C}_1(\mathbf{x})$ (left) and $\mathcal{C}_2(\mathbf{x})$ (right) from equations (4.26) and (4.27), respectively, of the data set *Tsukuba*. The reference image is depicted, for comparison of textured and homogeneous areas, in Fig. 6.4, p. 90, on the right.

Chapter 5

Shape reconstruction

The input of a shape reconstruction procedure consists of a 3D point cloud sampled from one or several depth maps obtained, as described in the previous chapter, from the corresponding reference image(s). The desired output is a discrete set of 3D points as well as triangles connecting these points. Actually, there are two tasks that face us here. First, we consider the urgent need for "close-to-real-time" algorithms and, consequently, their incremental character. In this case, we must make use of several reference images with corresponding depth maps and generate from them triangular meshes "up-to-now" without considering the global character of data. We describe the local incremental fusion of tessellations (LIFT) algorithm in Sec. 5.1. The second task will be unifying these results into a global mesh. To do this, we apply methods discussed in Sec. 5.2 and Sec. 5.3. Here the L_1-splines-based procedure of Sec. 5.2 is considered as our default method and represents the main innovation of our work. For comparison of the results on synthetic and real data, we implemented several methods mentioned in Sec. 3.2, namely, alpha-shapes, iso-surface extraction, grid-fit and conventional splines, details of which are reviewed in Sec. 5.3. Finally, the texturing procedure, described in Sec. 5.4, consists of choosing a reference camera for each triangle of the mesh.

5.1 Local tessellations from depth maps

The goal of this section is the description of an incremental procedure for compressing the data stemming from one or more reference images. We discuss first a method for tessellation of one reference frame (Sec. 5.1.1). If the number of reference frames is more than one, a naive approach is to consider the union of all tessellations. However, since such a tessellation usually contains spurious triangles, it is better to consider geometric constraints to remove these triangles. The *local incremental fusion of tessellations* (LIFT) algorithm will be explained in detail in Sec. 5.1.2.

5.1.1 Tessellation from one reference frame

We start our treatment of meshing with a minimum of information. Suppose we have one reference view and the corresponding depth map $\mathcal{D}$. If the depth map was retrieved according to Chapter 4, we already have a list of triangles consistent with the surface and can restrict ourselves to this list only. Since the vertices of these triangles were obtained in the process of (epipolar or simultaneous) tracking, some of them (especially those in textureless areas) get lost. As a consequence, the triangles have different sizes and, since many vertices lie in the textured areas, the number of triangles becomes unnecessarily high. In the rest of the section, we are concerned with compression and homogenization of the point set.

Starting from one single depth map, the simplest way to create a triangular mesh is to consider a canonical triangulation (see [105]): we subdivide the image into small squares of $q \times q$ pixels and further subdivide each square by one of its diagonals into two triangles. Two improvements of this approach are proposed and applied. The first improvement consists of choosing mesh vertices according to their accuracy. For every canonical vertex $\mathbf{x}$, we search, in a small window (some $q/4$ to $q/3$ pixels) around $\mathbf{x}$, for a point $\mathbf{y}$ with the maximum value of the confidence map (given, e. g., by (4.27)) and replace $\mathbf{x}$ by $\mathbf{y}$. The second improvement consists of subdividing a triangle with depth discontinuities into two smaller triangles along its symmetry axis. This kind of subdivision is very efficient (see Fig. 5.1) and preserves the angles of triangles. We have found out that the condition

$$\frac{d_{\max}(T) - d_{\min}(T)}{d_{\min}(T) + d} > \varepsilon$$

(where $d_{\max}(T), d_{\min}(T)$ are maximum and minimum depth values of a triangle and d, ε are positive constants) is a reasonable criterion for subdivision. The maximum possible order of iterative subdivisions (also called *generation* of triangles) is set to 4. In order to avoid *cracks* in the final surface (that result if a 2D mesh vertex is an inner point of an edge, because the corresponding 3D point is not necessarily incident with the an edge connecting the 3D endpoints of this edge) new vertices must be inserted, as in Fig. 5.1, bottom right. The process of inserting new vertices and subdividing triangles (which actually have passed the criteria mentioned above) to avoid cracks is called *restricted (top-down) quadtree triangulation* (RQT or RTDQT) and was introduced in [108]. The report [108] and the sources given there provide only hints about how to compute RTDQT. We describe in the two following paragraphs the basic terminology and the complete procedure for implementation of RTDQTs from the initial, canonical triangulation. For completeness, the procedure is formulated as pseudo-code in Alg. 8.3 of the appendix.

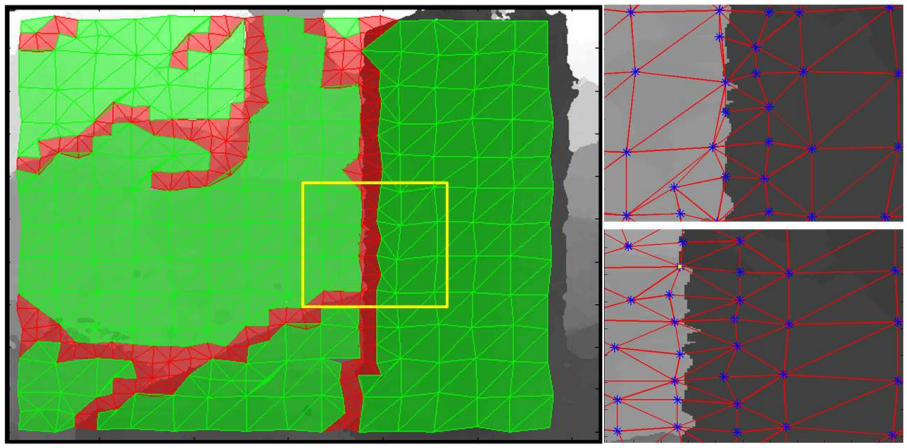

Figure 5.1: Left: The depth map of a reference frame from the sequence *Infrared* and the canonical triangulation of vertices corrected by the confidence map with pyramid-depth level 2; triangles with jumps of depth are shown in red, those without jumps in green. Top right: The edges and vertices of a part of the left image marked by the yellow rectangle. The cracks in the final surface are clearly visible. Bottom right: No cracks are visible if a *restricted triangulation* is performed.

The different levels of details for vertices and triangles correspond to generations g. On the coarsest level, $g = 0$, for a vertex at the midpoint of the largest edge of such a triangle, $g = 1$ and so on (see Fig.5.2, left). The generation of a triangle is given by the generation of

its youngest vertex, such that we can define for a triangle T (if $g(T) > 0$) its *parent* and two *children*. If the edge e of T opposite to its youngest vertex is not incident with the margin of the (rectangular) domain, then the triangle of the same generation sharing e with T is called, the *friend* of T. On the coarsest level, $g = 0$, these are just triangles which share the diagonal of the same rectangle. Note that two friends are not brothers (i. e. children of the same parent) unless $g = 0$ and that it is easily possible to compute for every triangle its friend by comparing the indexes of its vertexes.

The *activity status* s of a triangle can be active ($s(T) = 1$), if it is in the list, non-active ($s(T) = 0$), if a triangle of an older generation incident with T is active and lost($s(T) = -1$) if there is a chain of children of T ending up in an active triangle. Splitting a triangle T can always be performed by setting its status to 0 and the status of its children to 1 (see procedure **Split**(T) of Alg. 8.3). The RTDQT has the property that the generations of two triangles sharing the same edge differ at most by one, in other words, for every active triangle, either its friend, or the friend's parent, or its friend's child, is active (Alternatively, no vertex can be the inner point of a triangle's edge). The main idea of the **rtdqtSplit**(T), where T is the triangle to be split, is to identify the friend of T. If this triangle is active, it is split. If it is non-active, then, by definition, its parent P must be active and the procedure is repeated for P. Even if the algorithm is recursive, it will converge since the generation of P is necessarily lower than that of T and the moment must come when $g(P) = 0$. The process of refining starts with the canonic triangulation on the coarsest level. From level to level, the list of active triangle satisfying a splitting criterion is determined. For every triangle T of this kind (unless $g(T) > n_0$ where n_0 is a fixed number of maximum pyramid level), the procedure of **rtdqtSplit**(T) is performed and so a new set of active triangles is generated.

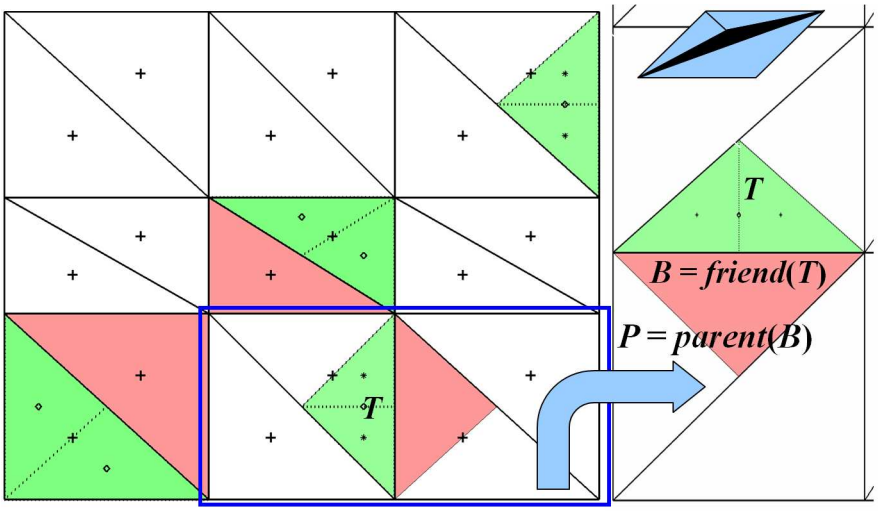

Figure 5.2: Left: Canonic triangulation of an arbitrary rectangular domain. The triangles of generation 0 are marked by crosses, for generation 1, by diamonds and dotted edges and for generation 2 by small stars. For four exemplar triangles marked in green, we show their children as well as their friends marked in red. There is no friend for a triangle near domain margin. Top right: Cracks are likely to emerge if restricted triangulation is *not* carried out. Bottom right: To perform the algorithm, one has to identify the friend B of a triangle T to be split, and if B is not active, then the same algorithm must be applied to the parent of B.

5.1.2 Tessellation from several reference frames

The union of local triangular meshes from different reference frames, as the output of the previous section, can be now considered. However, several triangles inconsistent with the surface may be included in the result. Also, there are many redundant triangles that emerge because the different reference images have a partial overlap (see, for instance, images on the top of Fig. 6.3, p. 84). After a local tessellation for the new reference view (denoted by $\mathcal{I}_m$) has been calculated, it is possible to reject several triangles that were incorrectly or redundantly assigned to the list of triangles consistent with the surface. In the following paragraph, we review the main ideas of the *local incremental fusion of tessellations* (LIFT) algorithm, which is also illustrated as pseudo-code in Alg. 8.4, p. 143.

From each pixel $\mathbf{x}$ of the current reference frame $\mathcal{I}_m$, we set the value of the boolean variable status to 1 and project the corresponding 3D point $\mathbf{X}$ (extracted by means of the corresponding depth map $\mathcal{D}_m$ at $\mathbf{x}$) into the other reference images $\mathcal{I}_1, ...\mathcal{I}_{m-1}$. (In [22], double indexing $\mathcal{I}_{r_1}, ..., \mathcal{I}_{r_{m-1}}$ was used to differentiate between the local approach within a subsequence and a global approach, where results from different subsequences are fused into a global mesh). Since we have depth values for these points ($\mathbf{x}_k$), we can compute the error term

$$\delta(\mathbf{x}) = d(\mathbf{X}) - \mathcal{D}_k(\mathbf{x}_k)$$

with depth $d(\mathbf{X})$ computed from P_k according to (4.1). For a positive constant ε (tolerance), $\delta(\mathbf{x}) > \varepsilon d(\mathbf{X})$ means that $\mathbf{X}$ occludes some point of $\mathcal{D}_k$; in this case, the *occlusion counter* $o(T)$ for the triangle T incident with $\mathbf{x}$ is increased. On the other hand, $|\delta(\mathbf{x})| < \varepsilon d(\mathbf{X})$ means that the pixel $\mathbf{x}$ was already processed at an earlier stage, so, in this case, the *redundancy counter* $r(T)$ of T is increased. In either of these situations, the variable status is set to be 0. After all pixels of the new reference image has been processed, we delete all triangles for which either $o(T)/a(T) > 0.1$ or $(o(T) + r(T))/a(T) > 0.99$ holds. Here $a(T)$ (*area counter*) denotes the number of pixels processed in every triangle. The starting values for the counters for $a(T), r(T), o(T)$ are all set to 0 for every triangle T.

A modification of this algorithm can be also found in [22] and it was originally applied on the Delaunay triangulations from the point sets in the reference images. The most significant difference between Alg. 8.4 and [22] is the following. Since Step 2.2 of our pipeline was completely omitted in [22], the evaluation of triangles took place *within* LIFT. For the case status $= 1$ after the inner loop in Alg. 8.4, the local approach with the aggregation function $\|c_{k'}(\mathbf{x})\|$ taken over *neighboring* images $k' = m \pm 1, m \pm 2, ...$, (not other reference images!) was performed; here $c_{k'}$ denotes the SAD-values from either gray or color values in a small window. If the value of the aggregated cost function exceeds a threshold, the pixel is declared as inconsistent with the surface. After all pixels of the new reference image have been evaluated, also triangles with a high percentage of pixels inconsistent with the surface are deleted as well. This has the advantage of performing a geometric and image-based evaluation on triangles in one step but the disadvantage of potential wrong classification of triangles. For example, large triangles from homogeneous, untextured regions are biased to be included into the list while triangles near occluded regions are biased to be excluded, since the aggregation function near occluded regions is less robust than the one chosen in (4.20).

We now refer to other differences between Alg. 8.4 and [22] as well as extensions of the LIFT algorithm. Fitting dominant planes into local tessellations and correcting points in the direction of normal vectors of these planes is a meaningful preprocessing step. The computation of dominant planes proceeds by means of the RANSAC procedure with the $T_{d,d}$-test (see [95]) until a sufficiently large consensus set is obtained. After the 3D points of this set are projected onto the plane according to (3.3), they are deleted (temporarily) from the point list and the procedure begins again. This has an advantage, beside improved

position of 3D points, that triangles lying in one of the dominant planes can be preferred by decreasing the maximum threshold for percentages of redundant and occluded pixels ($o(T)/a(T)$ and $r(T)/a(T)$, respectively) within them. In order to reduce computing time, the set of test points can be diminished from all pixels of the reference frame to the 3D points available up-to-now. This idea is proposed by [99]. Similar to [111], we also undertook efforts to avoid inconsistent meshes (i. e. those locally non-homeomorphic to a plane) and to reduce the number of vertices by fusing vertices of the new local tessellation with those of the previous one if they are too close. The closest point in the previous mesh is computed in the Hausdorff metric calculation (covered in Chapter 6). As the final step, we optionally delete triangles of the previous mesh that occlude the new mesh.

Clearly, for an increasing number of frames, it becomes quite expensive both to keep all reference images with the corresponding depth maps in memory and to process the new reference image while recalling *all* available reference images. The computational cost of such a procedure depends quadratically on the number of reference images. More sophisticated methods (for example, octree decomposition of the 3D space to be reconstructed) can process all tessellations simultaneously. These methods will certainly be a topic of future work. In the current implementation, in order to keep the cost of the procedure linear, we keep and process only a fixed number, between 2 and 5, of previous local tessellations. Other important parameters of the algorithm are the following:

1. The number of images in the subsequences is 5 to 7, as we will see in Chapter 6.
2. The number of frames between the frames within a subsequence varies between 2 and 12, depending on the sensor's velocity (see also Chapter 7).
3. The distance between subsequence in the current implementation is chosen so that two successive subsequences *almost overlap*, i. e., the number of frames between the last frame of the k^{th} and the first frame of $k+1^{\text{st}}$ subsequence is small.
4. The value of q (from Sec. 5.1.2) of the resolution on the finest level is 10-20 pixels. Consequently, it is 40-80 pixels for the coarsest level. The number of triangles in a tessellation in a subsequence usually does not exceed 10000.
5. Finally, the value of ε in Alg. 8.3 depends on the distance from the camera center to the object points, the baseline, and the focal length. Mostly $\varepsilon = 0.05$.

The procedure described in this section allows obtaining a close-to-real-time reconstruction in the form of (quite regularly distributed) sample points in the areas covered up to now and triangles that connect these points. This concept is sufficient for the majority of applications. However, the visual quality of models thus obtained is unfortunately not always sufficient. Two causes of insufficient visual quality are holes and other topological inconsistencies in the triangular mesh and noise in the triangle vertices. In order to solve these problems, we will consider the whole point cloud in the next sections.

5.2 L_1-splines-based procedure

The core element of our algorithm for shape reconstruction is the L_1-splines-based procedure, also described in [24]. Starting with a 2D tensor-product domain $(u_i, v_j), i = 0, ..., I, j = 0, ..., J$, our main task is to obtain a differentiable homeomorphism in the form of a cubic spline that approximates the point cloud. Explicitly, this means that the surface to be reconstructed must be homeomorphic to a plane. This puts restrictions on the topological variety of surfaces, but it is a plausible assumption for a flying sensor covering the urban terrain and thus eliminates, for a vast majority of cases, a large source of errors.

Since we strive for generic models automatically instantiated for data sets with irregular density of points, high percentage of outliers and sharp changes of curvature, we cannot rely on most least-squares-approaches. In order to obtain, on arbitrary grids, smooth approximations free from extraneous overshoot and oscillations, we adopted the ideas of L_1 approximating splines, whose main idea (see [85] and references therein) consists of replacing (the outlier-sensitive mean-based) L_2-norm by the median-based L_1-norm. Overall, our algorithm consists of four steps, namely,

1. Generation of a nonparametric 2.5D surface from the point cloud in form of a C^1 cubic spline
2. Creation of a parametrized data set using the latest 2.5D or 3D surface
3. Generation of a parametric 3D surface and return to Step 2 until a stopping criterion is satisfied
4. Tessellation of the 3D surface.

The point cloud serves as input of the algorithm while for texturing step, explained in Sec. 5.4, camera matrices and depth information are also needed. If one wants to reconstruct a smooth surface from depth maps or 3D character of the scene is not present, Steps 2 and 3 can be omitted. When vertical structures (like building walls have) to be reconstructed, Steps 2 and 3 are necessary and a pair of independent parameters u, v are to be determined. This is why we denote the surfaces obtained in Step 1 and Step 3 by nonparametric and parametric, respectively.

The four steps will be explained in the Subsections 5.2.1, 5.2.2, 5.2.3, and 5.2.4, respectively, of this section.

5.2.1 Functional and algorithm for 2.5D L_1 splines

The first step of the procedure is orientation of the point cloud $\mathcal{X} = \{\mathbf{X}_m | m = 1, ..., M\}$, since the nonparametric 2.5D representation assumes that one is able a priori to rotate the point cloud so that the z-axis coincides roughly with the physical vertical direction. For the data considered here, this assumption is reasonable, since the physical vertical direction can be estimated either by the normal vector of the plane robustly approximating the camera centers or by the dominating direction of vertical straight lines (detected by [30] in the images and triangulated by means of the DLT-method of [61]). This latter approach was successfully used in [97].

In this section, the problem of the 2.5D surface approximation given a set of sample points $\mathcal{X}$ is considered. Given a rectangular grid (u_i, v_j) (where $u_0 < u_1 < ... < u_I, v_0 < v_1 < ... < v_J$ and the $\{(x_m, y_m)\}$ of the data points are assumed to lie in the rectangle $[u_0; u_I] \times [v_0; v_J]$), we wish to approximate the data with a C^1 cubic spline $z(u, v)$ that best passes through the data points.

The (vertical) error of a single sample point $\mathbf{X}_m$ is $|z(x_m, y_m) - z_m|$, where $z(x, y)$ is given by (3.8) or by the analogous formula for one of the other triangles. The way to aggregate the error of the whole data set has a large influence on what surface one obtains. Unfortunately, the traditional choice which is the *least squares minimization*

$$\sum_m (z(x_m, y_m) - z_m)^2$$

virtually always produces inaccurate results with extraneous artifacts and oscillation in areas of rapid curvature change (for example, vertical discontinuities or near-discontinuities,

which are common in terrain). Evidence in the recent literature [85] suggests that surfaces calculated by minimizing sums of absolute values are more robust and have fewer artifacts than surfaces calculated by minimizing sums of squares. For this reason, we decided to minimize the sum of the absolute values (along with other terms) instead of the sum of squares.

The functional that we minimize to create an L_1 spline consists of a weighted sum of the absolute (vertical) deviations of the data from the surface, a smoothness term, similar to the Laplacian of Sec. 3.2.4 and a regularization term that resolves nonuniqueness when it occurs:

$$\begin{aligned}(1-\lambda)\sum_{m=1}^{M} w_m \left|z(x_m,y_m)-z_m\right| + \lambda \int \left(|z_{uu}| + 2\,|z_{uv}| + |z_{vv}|\right)\mathrm{d}u\,\mathrm{d}v \\ +\varepsilon \textstyle\sum_{nodes}\left(|z_u| + |z_v|\right) \longrightarrow \min .\end{aligned} \tag{5.1}$$

In the first term (data term) of (5.1), the weights w_m can be chosen to reflect uncertainty in the point coordinates. If there is no information on the uncertainty in the point coordinates, all of the w_m are set equal to 1. The parameter $\lambda \in [0;1]$ expresses the balance between how closely the data points are fitted and the tendency of the surface to be close to a piecewise planar surface, without extraneous, nonphysical oscillations. If λ is too small, the second term (smoothness term) of (5.1) becomes rather unessential and so the negative effects caused by outliers become clearly visible. If, however, it is too large, areas near characteristic edges become oversmoothed. In order to approximate the integral which makes up the smoothness term in (5.1) by a discretized value, each grid cell $[u_i;u_{i+1}] \times [v_j v_{j+1}]$ is divided into N^2 equal subcells ($N \geq 3$) and the sum of absolute values of the integrand at the midpoints of those sides of the subcells that are interior to the cell is computed. The value of the integrand is approximated by differential quotients of function values given by (3.8). The last term of (5.1), consisting of the sum of the absolute values of the first derivatives at the grid nodes, is added to the functional in order to prevent it from having a non-unique minimum. L_1 functionals are in general, non-convex and can have an infinite number of solutions. This third term is responsible for choosing from this set the most physically meaningful one. If ε is small enough, consideration of the last term in (5.1) does not change the minimum value of the functional.

The task is thus to solve an overdetermined system of equations $\mathcal{A}\mathbf{b} = \mathbf{c}$ in the L_1 norm. Formally:

$$\mathbf{b} = \arg\min_{\mathbf{b}'} \left(\| \underbrace{\mathcal{A}\mathbf{b}' - \mathbf{c}}_{\mathbf{r}} \|_1 \right), \tag{5.2}$$

where $\mathcal{A}$ is a coefficient matrix stemming from (5.1) that has $r = M + 6IJN(N-1) + 2(I+1)(J+1)$ rows and $3(I+1)(J+1)$ columns (recall that M is the cardinality of the point set, N is the number of grid cells used for discretization of the integral in (5.1) and $I \times J$ is the dimension of the grid). It can be assumed that $\mathcal{A}$ has the full rank. A linear program can be obtained from (5.2) by considering the residuals $\mathbf{r}$. We have to minimize

$$\mathbf{1}_{2r} \begin{bmatrix} \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} \text{ subject to } [\mathcal{A} \mid I_r \mid -I_r] \begin{bmatrix} \mathbf{b} \\ \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} = \mathbf{c} \text{ and } \begin{bmatrix} \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} > \mathbf{0}_{2r}$$

with $\mathbf{r}^+, \mathbf{r}^-$ as in Sec. 1.4. The minimization is carried out by means of a *primal-affine algorithm*. This is an interior-point method that starts with a least squares solution of (5.2) and, by iteratively updating the weight matrix W and computing the weighted least-squares

solution $W\mathcal{A}\mathbf{b} = W\mathbf{c}$, either converges to a L_1-solution of (5.2) (if parameter updates lie below a reasonable tolerance) or terminates if a maximum number of iterations is reached. The algorithm converges theoretically both for cases of unique minima [131] and for cases of multiple minima [2]. It is closely related with the robust least-squares approach because the outliers are supposed to be given smaller weights in the course of the minimization procedure. Consequently, it is possible to keep track track on outliers in the data; however, these outlier tests are not carried out in our approach.

The proof of the statement that primal-affine algorithm corresponds to a L_1-solution of (5.2) was given in [102]. The most time-consuming step is the least-squares solution of the overdetermined linear system, that is, solution of $\mathcal{A}^T W^2 \mathcal{A}\mathbf{b} = \mathcal{A}^T W^2 \mathbf{c}$. By properly ordering the unknowns, the symmetric, positive definite matrix $\mathcal{A}^T W^2 \mathcal{A}$ can have a minimal one-sided bandwidth (number of superdiagonals + 1 for the main diagonal) of $3\min(I, J)+9$. We give, for completeness, the pseudo-code for the primal-affine algorithm that we use in Alg. 8.5 of the Appendix and refer to [85] for further details.

5.2.2 Parameterization of data points

While the method presented in the previous section produces good results for 2.5D data, the question now is how to generalize it for a 3D point cloud. What we need is a global parametrization u, v that allows calculation of a triplet of splines $x(u, v), y(u, v), z(u, v)$, which we now denote by $\mathbf{X}(u, v)$. Such a parametrization usually exists for typical airborne video data of an urban scene, because the surface is usually homeomorphic to the plane. If the point density is sufficient and adaptive to curvature changes, one could apply methods of *multi-dimensional scaling* (see, for example, [35]) and (in the case of 3D to 2D dimensionality reduction) closely related *surface flattening*. These methods roughly consist of minimizing a norm of a matrix with observations

$$\mathrm{dst}\left((u_m, v_m), (u_n, v_n)\right) - \mathrm{dst}(\mathbf{X}_m, \mathbf{X}_n)$$

over $2M$ values of the parameters u_m, v_m and where $\mathbf{X}_n, n \in \{1, ..., N\}$ is a neighbor of $\mathbf{X}_m$. The choice of neighbors can be carried out by means of the approximate nearest neighbors (ANN) algorithm as described in [104]. In the context of surface reconstruction by bivariate B-splines, this approach was applied by Eck and Hoppe in [42]. Unfortunately, despite the band structure of the $MN \times 2N$ observation matrix, solving the system for (u_m, v_m) was noted to be an extremely time-consuming and unstable process. We project the data points $\mathbf{X}_m$ onto the (most recently generated) surface to obtain "corrected" points $\hat{\mathbf{X}}_m$ and use its coordinates $(u, v) = (\hat{u}_m, \hat{v}_m)$ as a parameterization for the surface $\mathbf{X}$ to be calculated next. The unknowns in this case are the (u, v)-coordinates of the point $\hat{\mathbf{X}}_m$ at the surface that is closest to $\mathbf{X}$. We use the Levenberg-Marquardt algorithm [49, 61], where the cost function ε and the Jacobian $\mathcal{J}$ are given by:

$$\varepsilon = \varepsilon(u, v) = \mathbf{X} - \mathbf{X}(u, v) \to \min, \mathcal{J} = \left[\mathbf{X}(u, v)_u \;\; \mathbf{X}(u, v)_v\right].$$

The terms of $\mathbf{X}(u, v), \mathbf{X}(u, v)_v, \mathbf{X}(u, v)_u$ are given by (3.8) (in which one has to replace z by the entries of $\mathbf{X}$ and select the suitable Sibson-triangle) and its derivatives. While for parameterization of the 2.5D surface, the first two rows $\mathcal{J}$ are made up by the identity matrix and the third row is z_u, z_v, it is a full 3×2 matrix at all following iterations (see Sec. 5.2.3). This parametrization process is schematically visualized on the left of Fig. 5.3.

5.2.3 Functional and algorithm for 3D L_1 splines

After parameter values (u_m, v_m) have been assigned to each point (x_m, y_m, z_m) as indicated in the previous section, we compute a 3D L_1 spline by minimizing the functional

$$(1-\lambda)\sum_{m=1}^{M}|w_m|z(u_m,v_m)-z_m|+\lambda\int(|z_{uu}|+2\,|z_{uv}|+|z_{vv}|)\mathrm{d}u\,\mathrm{d}v+ \\ \epsilon\sum_{nodes}(|z_u|+|z_v|)+\sum_{1}^{12}[\text{analogous expressions}], \tag{5.3}$$

where by "analogous expressions" we mean replacing z in (5.3) by the 12 functions x, y, $x\pm y$, $x\pm z$, $y\pm z$ and $x\pm y\pm z$, respectively. The functional (5.3) is more robust (at the cost of computing time!) with respect to outliers than three *uncorrelated* functionals as in (5.1) for $x(u,v), y(u,v)$ and $z(u,v)$. Functional ((5.3)) is minimized by the primal-affine algorithm described in Sec. 5.2.1 (with details suitably adjusted). The complete process consists of starting from a 2.5D L_1 spline and then iterating the two steps of parameterization and 3D spline generation several times.

The smoothness parameter λ in (5.3) does not need to be the same as in the (5.1). The automatic choice of suitable λ is not a trivial problem. Neither theoretical nor heuristic guidance is currently available. Like in (2.9) of the image-based part of this dissertation, changing λ by small values (in our case ± 0.05) does not result in large changes in the L_1 splines. Usually, it is recommended that λ be bounded away from zero in the non-parametric spline since we must make sure that the correct topological relations are not affected by outliers. For other iterations, smaller values of λ can be used.

5.2.4 Tessellation of the spline surface

As a result from the previous sections, we have an explicit representation of the object surface $\mathbf{X}(u,v)$ and also of its partial derivatives. Our task now is to create a triangular mesh that best fits the spline surface. This triangular mesh will be, at a later stage, the main input of the texturing procedure: its task will be to texture each triangle using one of the available reference views.

Surface meshing

There are two possibilities for meshing the surface obtained using the procedures of Sec. 5.2.3. The authors of [24] applied the Delaunay triangulation of the (u,v)-values of the points $\hat{\mathbf{X}}_m$ (the points on the surface closest to the data points $\mathbf{X}_m$) of the last of the iteratively calculated spline surface. Points within a rectangle $R=[u_i;\,u_{i+1}]\times[v_j;\,v_{j+1}]$ are compressed into *multipoints* $\mathbf{X}_r$ that coincide with the center of R. Another possibility is to use canonical triangulation of spline nodes in the (u,v) domain (rectangles cut by one of the diagonals, as proposed in 5.1.1). Since the number of spline nodes in each direction is about 30-50, we are able to model our objects by means of several thousands of triangles. Although this second approach results in a higher number of triangles, we use it in our further considerations because it represents the spline surface at its finest resolution and the high number of triangles can be reduced by efficient mesh-manipulation methods described in Sec. 2.4.2.

In our implementation, an optional step after tessellation is mesh manipulation by an edge-flipping method. From the initial triangulation, the (u,v)-values of 3D points $\mathbf{X}$ and the values of their normal vectors $\mathbf{n_X}=(\mathbf{X}_u\times\mathbf{X}_v)/\|\mathbf{X}_u\times\mathbf{X}_v\|$, we wish to obtain a new mesh that is more consistent with $\mathbf{n_X}$, as indicated in Fig. 5.3, right. To do this, one starts with considering for a triangle T with vertices $\mathbf{ABC}$ the terms (actually, three scalar products)

$$\mathbf{n}_1(T)=(\mathbf{n}_{T,0})^T\cdot[\mathbf{n_A}\ \mathbf{n_B}\ \mathbf{n_C}]\,, \tag{5.4}$$

where $\mathbf{n}_{T,0}$ is the normal vector of the triangle given by

$$\mathbf{n}_{T,0} = \frac{(\mathbf{A}-\mathbf{B})\times(\mathbf{A}-\mathbf{C})}{\|(\mathbf{A}-\mathbf{B})\times(\mathbf{A}-\mathbf{C})\|}.$$

If the normal vector of T is nearly parallel to the normal vector at one of its vertices, the corresponding entry of the vector $\mathbf{n}_1(T)$ in (5.4) is close to ± 1. Therefore, we choose, among a large number of possible energy functions for a triangle T, the very simple term $E(T) = -\|\mathbf{n}_1(T)\|_\infty$ and wish to minimize the total energy $E(\mathcal{T}) = \sum_{T\in\mathcal{T}} E(T)$ over triangulations $\mathcal{T}$.

The next step of our minimization algorithm consists of obtaining all interior edges of $\mathcal{T}$. Each of them is associated with a *quadrilateral*, so the energy value $E(Q)$ of every quadrilateral Q is computed. The energy of Q is given by the sum of the energies of both triangles composing Q. The energy values are now stored in non-decreasing order.

The activity status of all quadrilaterals is now set to be 1. The iteration loop runs over all swappable quadrilaterals of the list, where a quadrilateral Q is declared swappable if its activity status is 1, all its angles do not exceed π and the angle between the *oriented* normal vectors of the two triangles from which Q is made up is below a fixed value $(\pi/2-\varepsilon)$. If $E(Q) > E(Q')$ (where Q' is a swapped quadrilateral), the triangles composing Q are replaced by those composing Q', incidence and energy information of all quadrilaterals around Q is recalculated and their activity status is set to be 1. Finally, the activity status of Q' is set to be 0.

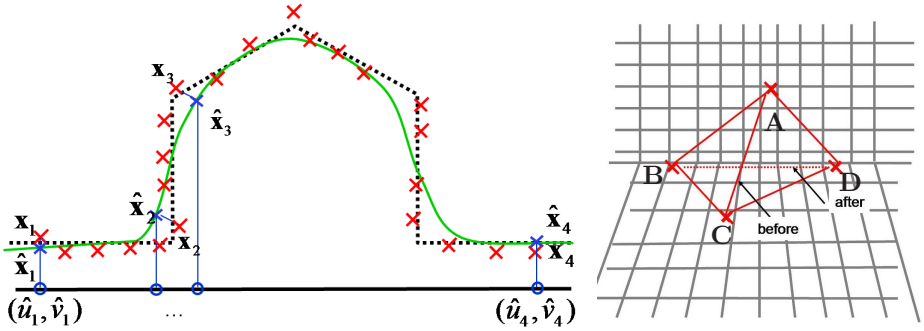

Figure 5.3: Left: Parameterization of the approximating spline surface (see Sec. 5.2.2). The 2.5D spline surface is depicted by the green curve, the point cloud is depicted by red crosses and the correct surface is indicated by a black dotted line. Points are projected onto the surface (depicted in selected cases by blue crosses) and the first two coordinates are chosen as independent parameters (blue circles). The approach will preserve topological relations of points when the inclination angle of the z-axis against the vertical direction of building walls is small and the input surface is good enough. Right: Visualization of the edge-flipping process. Two triangles sharing a common edge and not reflecting the values of the normal vectors of their vertices (given by derivatives) are flipped along this edge.

In our applications, it was convenient not to include quadrilaterals Q into the list when $E(Q)$ was below 0.0001 and so the number of iterations was always below 500. It is also important to point out that the final triangulation depends on the order of swapping and so there is generally no guaranty that, at the end of the process, the energy takes on the global minimum value $\arg\min_{\mathcal{T}} E(\mathcal{T})$ over all possible triangulations $\mathcal{T}$. However, since the energy of every swap reduces the total energy, it will be always lower than the energy in the beginning and therefore the algorithm terminates (in a local minimum of the total energy function) after a finite number of iterations, namely, when there are no longer any swappable quadrilaterals in the list. Further reduction of total energy can be achieved by considering

more sophisticated methods like *simulated annealing*, see, for instance, [118], but also here no statement can be made about conditions under which a *global* minimum of energy can be achieved in a reasonable time. Furthermore, simulated annealing is very sensitive to the choice of relaxation parameters and, as stated in [118], quantitative improvements of the *geometric* cost function are not as significant as those of the local result.

5.3 Implementation details of other procedures for surface reconstruction

In the next three short sections, we give brief descriptions of implementation details of several approaches that will be used to provide comparison with results obtained by the L_1-splines-based procedure.

5.3.1 Alpha-shapes

The main properties of Alpha-shapes (α-shapes, [43]) were discussed in Sec. 3.2.1. Because of its indisputable advantages (no need for 3D parameterization, regularized triangle sizes etc.), the α-shapes-based procedure will be our default TIN-based method for shape reconstruction. To compute an α-shape, one needs the Delaunay tetrahedrization of the input point cloud, after which for each face, the maximum and minimum value of α for which T belongs to the α-shape can be obtained. These values are stored in a $2 \times N$ array where N is the number of triangles. Then it is a trivial task to select triangles belonging to the α-shape from this array.

The value of α should be slightly larger than the average triangle edge size in meshes obtained by a local method. After the α-shape has been obtained, the vertices and mesh can be manipulated in order to detect large planar regions and to reduce the number of triangles. For the comparison of computational results, the Steps 1-4 of the procedure mentioned at the beginning of Sec. 5.2 are replaced by triangulation with α-shapes.

5.3.2 Iso-surface extraction

Similar to the previous section, we wish to understand the advantages and disadvantages of iso-surface extraction with respect to our applications. The most important parameters for the iso-surface extraction algorithm of [70] described in Sec. 3.2.2 are ρ (sampling density) and r (resolution). If ρ is too large, completely wrong results for the signed distance function can be obtained, as depicted in Fig. 5.4, top left. However, if ρ is too small, two many values remain undefined (Fig. 5.4, top right). A resolution grid that is too fine usually leads not only to an unnecessarily large number of triangles with coordinates of vertices contaminated by noise, but also to increased computing time, since, at least at present, $g_x \cdot g_y \cdot g_z \cdot M$ distance evaluations (where g_x, g_y, g_z are the numbers of nodes in a grid in the x, y and z directions, respectively, and M the cardinality of the input point cloud) for determination of closest points in (3.3) are required for every grid point. Grids that are too coarse usually ignore some fine details. For the data set *Gottesaue*, depicted in Fig. 5.4, bottom (intermediate result), we set $g_x = g_y = g_z = 2^6 = 64$.

Computing depth maps and rendering local tessellations according to Sec. 5.1 allows the assumption of a constant point density at least in large portions of the surface. We can compute the neighbors of a sample point using the well-known *approximate nearest neighbors* (ANN) method, [104]. The matrix of distances between the point set and its neighbors is obtained as well and a number proportional to the median of these distance values is set to be ρ. Now, if a sample point $\mathbf{X}$ projected by a reference camera, in which it is visible, lies in

a triangle consistent with the surface, we assign to the normal vector at $\mathbf{X}$ the normal vector of the triangle. Otherwise, the calculation of the normal vector is carried out by fitting a plane with RANSAC from the neighbors of $\mathbf{X}$. The most difficult part of the algorithm, namely determining the orientation of the normal vector, can be significantly simplified in our applications, because one can take the vector from $\mathbf{X}$ toward the camera as an initial orientation of the normal vector of $\mathbf{X} \in \mathcal{X}$. Multiplication by -1 proceeds merely in the regions of sharp curvature changes (it can not be completely skipped!) and is completed after several iterations. Finally, meshing is provided by the marching cubes algorithm [91].

In the post-processing step, another problem, namely, ghost triangles near the medial axis, can be partially solved by selecting a rather small value for resolution and then deleting all vertices lying in the cube where either the maximum of negative values at the vertices or the minimum of positive vertices is bounded away from zero. Finally, neighborhood relations of vertices sharing a triangle edge are established and we delete all triangles with too few neighbors.

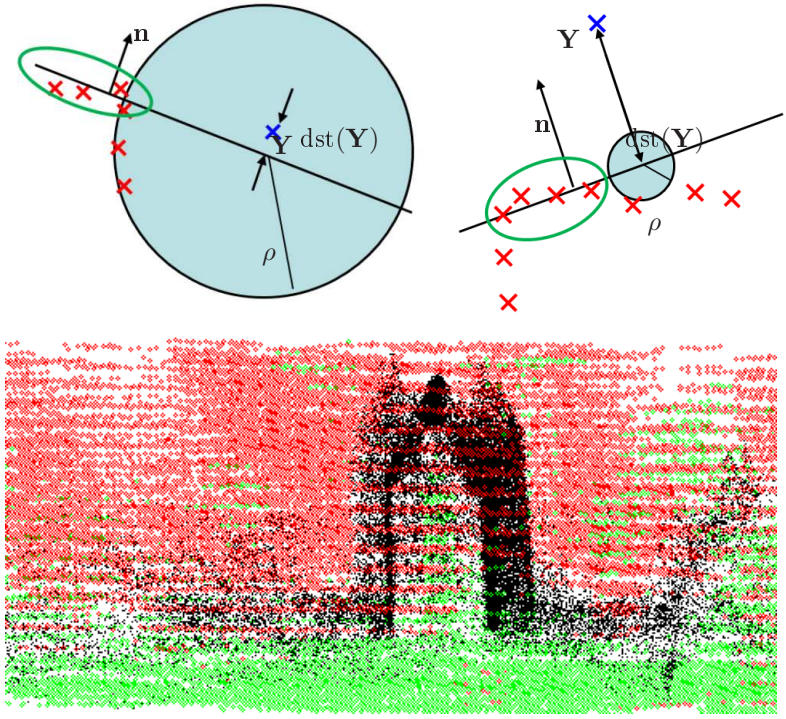

Figure 5.4: Top: Problematic of parameter choice for iso-surface extraction. Top left: Too large ρ and a too small value is assigned to $\mathrm{dst}(\mathbf{Y})$, namely the distance to the regression plane. Top right: Since no points of the sample lie in a circle of radius ρ (which was chosen to be too small), the value of signed distance function at $\mathbf{Y}$ remains undefined. A meaningful value would be assigned if ρ were slightly larger. The regression plane is always denoted by the thick black line, its normal vector by the arrow on the left, the input point set by red crosses and the points included into the consensus set for plane fitting by green ellipses. Bottom: An intermediate result of signed distance function extraction for the data set *Gottesaue*. The original point cloud is indicated in black, $\mathbf{Y}$ with positive values of the signed distance function in green, and those with negative values in red. One can see several wrong assignments which are mainly situated near regions with sharp gradient changes (e. g., towers), points of medial axis and outliers in the data. The result of the complete procedure for this data set is depicted in Fig. 6.33, p. 113, middle left.

5.3.3 Conventional (L_2) splines and gridfit

The procedure for conventional, or L_2 splines of Sec. 3.2.4 is the same as that stated for L_1 splines in Sec. 5.2 except that the absolute values in the minimization principles of (5.1) and (5.3) are replaced by squares. A conventional spline is easily obtained, since it is (the spline corresponding to) the value of $\mathbf{b}$ after the first iteration of Alg. 8.5. The tessellation procedure remains the same. Comparison of procedures based on conventional splines with our default procedure based on L_1 splines is of interest because conventional splines are commonly used in geometric modeling and because all of the differences in the results can be directly attributed to the differences in the functionals by which these splines are calculated.

Computational results generated by the Gridfit routine ([38], see also Sec. 3.2.4) for 2.5D surfaces in particular and for different grid sizes, regularization kinds, and smoothness terms, help understand to what extend C^0-*surfaces* can perform successful reconstruction from photogrammetrically generated point clouds. Comparison of surfaces generated by Gridfit with L_1-splines provides an additional component of comparison that assists in understanding the context.

5.4 Texturing

To texture the 3D surface obtained by a global algorithm, we must find for every triangle T of the mesh a (reference) camera k that completely observes it under a reasonable angle. "Reasonable angle" means that the cosine of the angle α between the triangle normal $\mathbf{n}_{0,T}$ and the ray connecting its center of gravity (denoted by $\mathbf{G}(T)$) with the location of reference camera ($\mathbf{C}_k$) must be bounded away from zero. The choice of such a camera is not a trivial task because there is a lot of available information (the distance $\mathbf{G}(T)\mathbf{C}_k$, which should not become too large, depth information for points within T in $\mathcal{I}_k$, and many others). So we first extract by means of depth maps information about which vertex is seen in which reference image. This set will be denoted by $v(\mathbf{X})$ for the given vertex $\mathbf{X}$. Then, the sets $\cup_{i=1}^3 v(\mathbf{X}_i)$ and $\cap_{i=1}^3 v(\mathbf{X}_i)$ are evaluated for the three vertices of T. If the first set is non-empty, we take one view from the intersection set for texturing. Otherwise it is clear that the triangle can theoretically be textured using any image of the second set. We therefore start with removing the views that cannot texture T either because at least one of it vertices is not visible in the image or because of coarse aberration from the indicated depth information, in other words

$$\min\left(\min_i\left(\left|\mathcal{D}_k(P_k\mathbf{X}_i)-d(\mathbf{X}_i)\right|\right),\left|\mathcal{D}_k(\mathbf{G}(T))-d(\mathbf{G}(T))\right|\right) < 2\varepsilon\cdot\mathcal{D}_k(\mathbf{G}(T)),$$

where $d(\mathbf{X})$ is the depth of the point $\mathbf{X}$ according to (4.1), and on the right, ε is the same as in Alg. 8.4 and the factor 2 considers the fact that the positions of mesh vertices are slightly changed by a global method. In [24], the reference image with the smallest value of $c_1(k,T) = |\mathbf{G}(T)\mathbf{C}_k|(1-\cos\alpha)$ was chosen from the remaining set of reference images. If the uncertainties in camera parameters are not negligible, the approach is modified by choosing the minimum value of $c_1(k,T) - Ac_2(k,T)$ where A is a large positive constant and the value of $c_2(k,T)$ is set to 1 if a triangle sharing an edge with T is chosen by the reference image k and 0 otherwise. This not only allows selecting cameras with low values of α and small values of $|\mathbf{G}(T)\mathbf{C}_k|$ for texturing T, but also making small errors of point projections less visible (since triangles are textured cluster-wise from reference images). The last strategy achieves its best impact as an iterative procedure where triangles already textured are propagated along their edges. Finally, triangles that cannot be assigned to any camera are textured by a neutral color and their transparencies are set to 0.5.

Chapter 6

Evaluation of algorithms

After presenting reconstruction algorithms in Chapters 4 and 5, evaluation of results obtained by these algorithms will be described in this chapter. To emphasize the generic character of our approach, video sequences of quite various types and quality will be described in Sec. 6.1. For each frame of the video sequence, we are given, as stated in Chapter 1, the corresponding camera matrix. As additional input, a sparse set of 3D-points is given together with a visibility information (which point is seen in which camera). Evaluation of sparse tracking algorithms, which represent Step 2.1 of our reconstruction pipeline of p. 15, takes place in Sec. 6.2. Qualitative and quantitative evaluation of dense image-based methods (Step 2.2) is provided in Sec. 6.3. Evaluation of the methods for shape reconstruction described in Sec. 6.4 (Steps 3.1 and 3.2) is divided into two parts: in Sec. 6.4, screen-shots of meshes and textured model representations are presented; a separate section (Sec. 6.5) is dedicated to quantitative evaluation. In order to visualize different steps of our algorithm from input images over depth maps and dense points clouds to textured model instances, qualitative results for two additional video sequences are presented in Sec. 6.6; for these sequences, only main challenges will be mentioned, but a detailed performance analysis will not be performed. Information about computing time is given in the concluding Sec. 6.7.

6.1 Data sets

The first data set that we discuss in this section is the well-known *Tsukuba* data set [115]. Several images and the disparity map between two of these images ($\mathcal{I}_{3,3}, \mathcal{I}_{3,4}$) are provided for verification and evaluation of the results. Although we do not consider this data set as characteristic for our applications and hence do not perform shape reconstruction in this case, we decided to demonstrate the performance of the image-based part of the algorithm for a data set with available ground truth. Since the surface has many self-occlusions, the grading of the *geometric complexity* of the scene is declared as high in Table 6.1 (where relevant properties of all data sets mentioned in this Chapter are summarized). For point tracking, we use either five images ($\mathcal{I}_{2,2}, \mathcal{I}_{3,2}, \mathcal{I}_{3,3}$ (reference image) $\mathcal{I}_{3,4}$ and $\mathcal{I}_{4,4}$ – in order to mimic a flying sensor) or nine images with ($2 \leq r, c \leq 4$) and again $\mathcal{I}_{3,3}$ is chosen to be the reference image. For dense estimation, the number of images was chosen to be five.

In the next data set, *Turntable houses*, only the moving parts of the images need be reconstructed. Since the (unmoved) background (see Figs. 6.1 and 6.5) does not satisfy the collineation constraint, it does not make much sense to perform a dense reconstruction of this data set, but it is still interesting to observe the results of sparse tracking for different methods and parameter sets for this labor data set. The extraction of camera trajectory and sparse point cloud was carried out by the structure-from-motion approach of [22, 23]

followed by a bundle block geometric error minimization. The total number of cameras was 81 and the number of points 8159. The shape reconstruction methods are applied to this point cloud. Several video frames of the data set as well as the result of Step 1 of our reconstruction pipeline is visualized in Fig. 6.1, bottom.

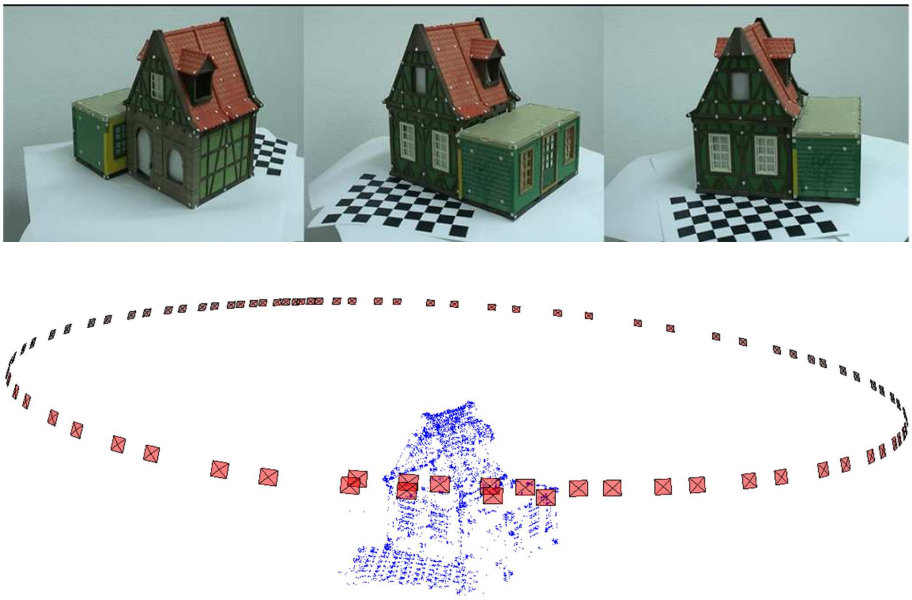

Figure 6.1: Top: Three views from the original sequence *Turntable houses*. Bottom: The complete camera trajectory and the point cloud as a result of a structure-from-motion algorithm are the input of our reconstruction pipeline.

Our next sequence, *Gottesaue*, shows a real building Gottesaue Palace in Karlsruhe, Germany. The results of the reconstruction presented in [24, 25] were derived from 339 images and 39059 points obtained using the methods of [22] without bundle adjustment (which was not possible to perform reliably for such a large number of cameras). The results depicted in Fig. 6.6, bottom, were produced by generating four workspaces from subsequences showing different but overlapping parts of the building. Each subsequence was self-calibrated and reconstructed by [22] with bundle adjustment in a Euclidean space and then transformed into the same coordinate system. Nevertheless, because of the flight in turbulent conditions (with a consequence of a high level of noise and outliers, partly produced by drift effects of the camera trajectory) and the challenging geometry (huge depth ranges, fine details in the structure of the palace and its surrounding terrain, abundance of non-fronto-parallel planes), the radiometric and geometric complexity of this sequence are classified to be high and very high, respectively, in Table 6.1. The total numbers of camera matrices and points are 310 and 39165, respectively; several frames of the video sequence together with the results of sparse reconstruction are illustrated in Fig. 6.2.

We also present an infrared video sequence of a skyscraper in the city Frankfurt (Oder) in the eastern part of Germany. This video was also recorded by an airborne sensor (in a helicopter) and reconstructed by a SLAM-method [9] after self-calibration of a short subsequence was performed. The whole sequence has 418 images and 3109 points (see Fig. 6.3). Its particular complexity consists of dead pixels and many textureless areas (radiometry) as well as slanted surface of huge depth ranges in the background (geometry). Contrary to the

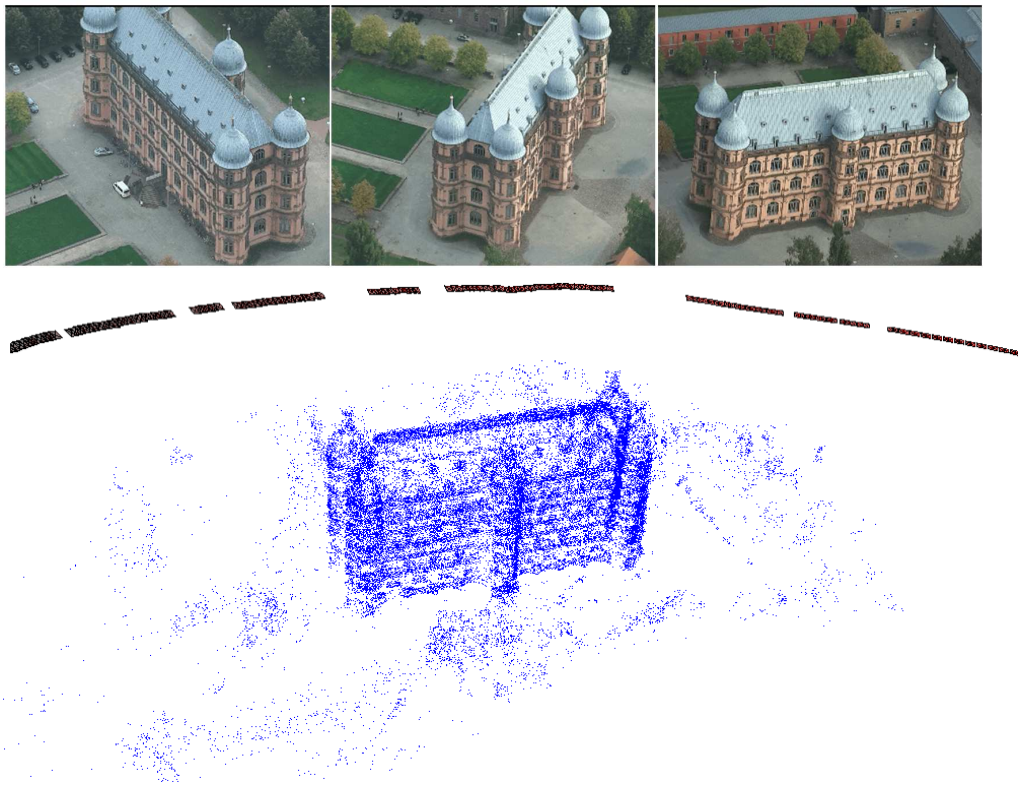

Figure 6.2: Top: Three views from the original sequence *Gottesaue*. Bottom: Part of the camera trajectory and the point cloud as a result of a structure-from-motion algorithm due to [22] are the input of our reconstruction pipeline.

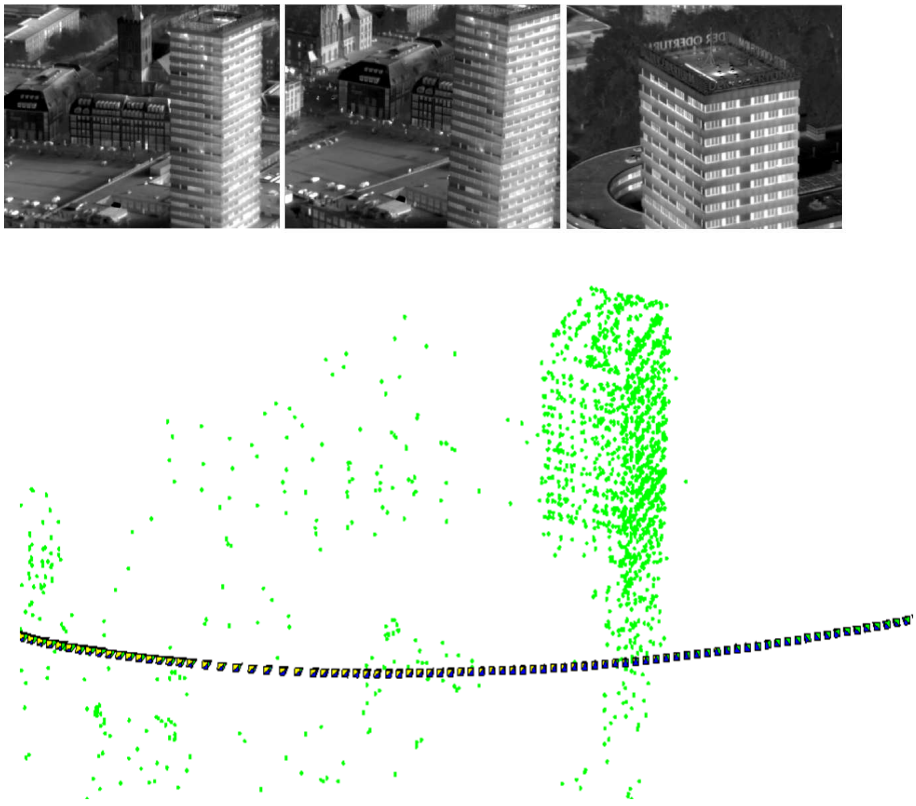

Figure 6.3: Top: Three views from the original sequence *Infrared*. Bottom: Part of the camera trajectory and the point cloud as a result of a SLAM algorithm due to [9] are the input of our reconstruction pipeline.

points in the background, 3D points at the walls of the tower can be computed with a higher accuracy. For this reason and also because of the abundance of planar regions (similar to the situation in the sequence *Turntable houses*), we decided to show the result of the LIFT algorithm in Sec. 6.4.1 for these two data sets.

The video sequence *Ettlingen church* is used for quantitative evaluation of several shape reconstruction algorithms in Sec. 6.5 because a laser point cloud representing the surface is available. Because of many fine details, the complexity of scene is high. Finally, to demonstrate the reliability of our reconstruction pipeline for different situations, we present in Sec. 6.6 qualitative results for two additional data sets: *Wangen* and *Speyer*, but, since the quantitative analysis of these two sequences does not represent a significant difference from other data sets and, therefore, was not carried out, we do not consider them in Table 6.1 below. Note that in Table 6.1, an important measure for geometric complexity of the input data is given by the ratio *field of view/spatial resolution ranges* that reflects the ranges for the quotient baseline/depth.

Table 6.1: Summary of data sets available for this work. It is also mentioned which experiments (denoted by Sec. 6.2-6.5) were carried out for which data set. "dl" means daylight video with three spectral channels, "ir" infrared video, and "i" denotes image sequence. The complexity of radiometric or geometric configuration of the scene is denoted by "!", if the scene is very complex, a "!!" is put. If the reason why a certain experiment was *not* carried out with a certain data set is not given, it was omitted because of redundancy. See text for further details.

Data set	*Tsukuba*	*Turntable h.*	*Gottesaue*	*Infrared*	*Ettlingen h.*
dl/ir	dli	dl	dl	ir	dli
sensor platf.	fixed	hand-held	Cessna	helicopter	hand-held
image	384 × 288	720 × 566	720 × 566	640 × 480	650×475
num. of frames / 3D-Points	9/-	81/8159	310/39165	418/3109	5/869
dist. cam/pt	10 to 39	7.5 to 10	17 to 25	10 to 17	14 to 17
focal (pix)	300	$1.07{\cdot}10^3$	$3.39{\cdot}10^3$	$4.67{\cdot}10^3$	$1.38{\cdot}10^3$
fow / spat. res. ranges	1 to 3.77	0.70 to 0.96	0.21 to 0.30	0.12 to 0.21	0.5 to 0.59
complexity rad./geom.	0/!	0/0	!/!!	!/!	0/!
test-runs	6.2, 6.3	6.2, 6.4	6.2, 6.3, 6.4	6.2, 6.3, 6.4	6.5

6.2 Sparse tracking and triangulation

For the benchmark data set *Tsukuba*, we first convert the data into the format described at the beginning of Chapter 4. Since the cameras have the same calibration and rotation matrices, we need only modify the camera centers. They lie in the same plane and in a equally spaced rectangular grid. We choose a reasonable calibration matrix to guarantee numerical stability of the calculations, rotation matrices are set to be identity matrices, and the translation vector corresponding to image $\mathcal{I}_{r,c}$ is $[1.5(c-3)\ 1.5(r-3)\ 0]^T$. The evaluation is carried out by projecting a 3D point into the images $\mathcal{I}_{3,3}, \mathcal{I}_{3,4}$ and computing the minimum absolute difference between $x_{3,3} - x_{3,4}$ and the true disparity values d_{gt} at rounded $\mathbf{x}_{3,3}$ and its 8 neighbors (in order to avoid rounding errors). In other words, we

have the set of incorrectly tracked pixels:

$$\mathcal{B} = \left\{ \mathbf{x} \left| \min_{\mathbf{v}} |d_{gt}(\mathbf{x}_{3,3} + \mathbf{v}) - (x_{3,3} - x_{3,4})| \geq 1 \right. \right\}, \tag{6.1}$$

where $\mathbf{v} = [v_x \; v_y]$, $-1 \leq v_x, v_y \leq 1$ and the error quantity $\min(\cdot)$ in (6.1) is denoted by ε. We provide in Table 6.2 the results of the state-of-the-art implementation of KLT-tracking, as well as the epipolar and simultaneous tracking described in Sec. 4.4, applied to 1238 characteristic points obtained as described in Sec. 4.2.

Table 6.2: The numbers of points tracked correctly, incorrectly and lost for different methods, different numbers of cameras and different window size (win) of the sequence *Tsukuba*. opt.r was set to zero everywhere. For the standard KLT-method, image pyramids at the third level were necessary to produce these results. The total number of points was 1238.

meth.	KLT, pyr = 5, cam = 3				KLT, pyr = 5, cam = 5				KLT, pyr = 5,cam = 9			
win	5	7	9	11	5	7	9	11	5	7	9	11
total	936	1005	1030	1039	650	741	787	798	475	572	608	619
cor.	859	916	941	954	649	732	777	789	474	570	606	618
incor.	77	89	89	85	1	9	10	9	1	2	2	1
lost	302	233	208	199	588	497	451	440	763	666	630	619
meth.	KLT-epi, cam = 3				KLT-epi, cam = 5				KLT-epi, cam = 9			
win	3	5	7	9	3	5	7	9	3	5	7	9
total	1061	1038	1018	1003	993	991	949	929	987	974	942	910
cor.	918	938	933	918	978	981	938	917	981	965	936	900
incor.	143	100	85	85	15	10	11	12	6	9	6	10
lost	177	200	220	235	245	247	289	309	251	264	296	328
meth.	simultan, cam = 3				simultan, cam = 5				simultan, cam = 9			
win	3	5	7	9	3	5	7	9	3	5	7	9
total	1125	1156	1171	1181	883	970	975	982	613	733	754	766
cor.	1026	1086	1086	1076	872	962	971	964	612	731	749	761
incor.	99	70	85	105	11	8	4	18	1	2	5	5
lost	113	82	67	57	355	268	263	256	625	505	484	472

For the data set *Turntable houses*, the considered subsequence consists of seven images $\mathcal{I}_1, ..., \mathcal{I}_7$ and the triangulation results are shown for 900 points detected in the reference image $\mathcal{I}_4$. Since it is quite difficult to obtain a data set with reliable ground truth and since a comparison with results obtained by different methods shows similar tendencies as in the case of the benchmark sequence, we compare the results of the epipolar and simultaneous tracking algorithms for all non-benchmark sequences with the standard KLT-tracking algorithm. We use 1 pixel as threshold for reprojection errors for triangulation for which the number of outliers in the benchmark data set is extremely low. After the 3D points are normalized to have average standard deviation of x, y, and z-coordinates of 1, a point tracked by the epipolar and simultaneous tracking algorithms is declared as tracked correctly if the Euclidean distance between the corresponding 3D point and its counterpart obtained by the KLT-tracking algorithm is below 0.1. Table 6.3 shows how many points were lost, tracked correctly (cor.) and tracked incorrectly (incor.).

For the sequence *Gottesaue*, the number of points with a high response of the operator (4.8) of Sec. 4.2 is 1517. Again, seven images are used for triangulation. Table 6.4 shows the sensitivity of the standard KLT-method for a video sequence taken from a small plane in extremely bumpy and turbulent conditions while Table 6.5 shows triangulation results obtained by epipolar and simultaneous tracking. Finally, for the sequence *Infrared* (and its short subsequence of seven images), we are interested in keeping the number of outliers small

Table 6.3: Results of tracking characteristic points for the data set *Turntable houses* with seven images, variable window size (win) and rectification option (opt.r). The total number of points was 900. The standard KLT-tracking with the window size 11 and the number of image pyramid levels 5 yielded 180 points.

meth.	KLT-epi, opt.r = 0, init = 1					KLT-epi, opt.r = 1, init = 1				
win	3	7	11	15	19	3	7	11	15	19
total	279	360	380	414	425	231	284	323	356	377
cor.	92	138	144	153	152	67	99	116	126	136
incor.	0	0	1	2	3	1	0	1	2	2
lost	88	42	35	25	25	112	81	63	52	42

meth.	KLT-epi, opt.r = 0, init = 0					KLT-epi, opt.r = 1, init = 0				
win	3	7	11	15	19	3	7	11	15	19
total	188	303	348	377	393	138	263	316	339	361
cor.	44	101	127	139	148	26	69	100	120	128
incor.	1	3	3	3	2	2	3	4	4	4
lost	135	76	50	38	30	152	108	76	56	48

meth.	simultaneous, opt.r = 0, init = 1					simultaneous, opt.r = 1, init = 1				
win	3	7	11	15	19	3	7	11	15	19
total	525	617	671	684	691	896	896	898	898	898
cor.	124	131	141	139	142	162	164	164	165	165
incor.	1	0	1	2	2	18	16	16	15	15
lost	55	49	38	39	36	0	0	0	0	0

meth.	simultaneous, opt.r = 0, init = 0					simultaneous, opt.r = 1, init = 0				
win	3	7	11	15	19	3	7	11	15	19
total	563	574	583	575	564	889	872	840	818	792
cor.	114	128	143	145	145	130	147	160	160	163
incor.	3	3	1	3	1	49	31	16	16	13
lost	63	49	36	32	34	1	2	4	4	4

and, for this purpose, varied the norm of the value of maximum total error $\varepsilon_{\max}$; a point is lost if at the end of the optimization process of Sec. 4.4.2, $\|\mathbf{c}\|$ from (4.11) exceeds $\varepsilon_{\max}$. Table 6.6 (from [28]) shows the results for 1170 characteristic points.

Table 6.4: Results of tracking characteristic points by the standard KLT-tracking with and without initialization for the data set *Gottesaue*, seven images and variable window size (win) and the number of image pyramid levels (pyr).

meth.	KLT-epi, pyr = 5, init = 0						KLT-epi, pyr = 1, init = 0					
win	5	7	11	15	19	23	5	7	11	15	19	23
total	64	134	313	473	532	542	1	2	2	10	18	20
meth.	KLT, pyr = 5, init = 1						KLT, pyr = 1, init = 1					
win	5	7	11	15	19	23	5	7	11	15	19	23
total	96	212	369	479	532	542	99	200	349	430	480	489

Table 6.5: Results of tracking characteristic points for the data set *Gottesaue* with seven images and variable window size (win), rectification and initialization options. The maximal error per pixel and interaction was 30.

meth.	KLT-epi, opt.r = 0, init = 1					KLT-epi, opt.r = 1, init = 1				
win	7	11	15	19	23	7	11	15	19	23
total	765	829	833	852	849	767	868	867	872	861
cor.	375	424	433	428	415	369	429	442	443	427
incor.	12	10	5	7	17	18	12	3	2	11
lost	92	45	41	44	47	92	38	34	34	41
meth.	KLT-epi, opt.r = 0, init = 0					KLT-epi, opt.r = 1, init = 0				
win	7	11	15	19	23	7	11	15	19	23
total	999	1029	1045	1049	1046	946	977	975	947	930
cor.	400	417	421	410	405	399	427	441	432	429
incor.	25	12	11	22	22	22	14	2	6	12
lost	54	50	47	47	52	58	38	36	41	38
meth.	sim. $\varepsilon_{\max} = 50$, opt.r = 0, init = 1					sim. $\varepsilon_{\max} = 50$, opt.r = 1, init = 1				
win	7	11	15	19	23	7	11	15	19	23
total	866	881	890	856	848	1219	1217	1191	1129	1087
cor.	339	357	356	347	345	387	396	394	395	380
incor.	23	13	14	13	10	72	62	60	55	66
lost	117	109	109	119	124	20	21	25	29	33

We can see from Tables 6.2-6.6 that both policies (epipolar and simultaneous tracking) yield more reliably reconstructed points than the original version of KLT-tracking without considering camera matrices (the total number is always higher). For the video sequence *Gottesaue*, recorded in turbulent conditions, standard KLT-tracking fails to obtain a large set of correspondences if the number of image pyramid levels is below 5 (Table 6.4). As soon as the initialization of depths provided by triangular interpolation as described in Sec. 4.3 is carried out, the total number of reliably triangulated points depends mainly on the window size and not so much on the number of pyramid levels. For the epipolar and simultaneous tracking algorithm, initialization is not crucial. The results are similar to those in Tables 6.3 and 6.5. Increasing the window size usually contributes to a larger number of triangulated points, because the risk of ending up in a local minimum of the cost function declines; unfortunately, the computing time depends quadratically on the window

size. Increasing the number of cameras always contributes to better reconstruction, as one can observe in Table 6.2. The parameter $\varepsilon_{\max}$ practically does not influence the results of the epipolar tracking algorithm. In simultaneous tracking, it clearly contributes to a larger number of tracked points (and, clearly, outliers between them). The next question concerns the rectification option opt.r: for the data sets *Gottesaue* and *Infrared*, one can significantly reduce the number of outliers for growing window size in epipolar tracking by using opt.r. The explanation is the following: while, for smaller windows, the interpolation errors in values of derivatives computed for rectified images deteriorate the results, the real invariance against rotation begins to show its effects for larger windows. We do not discuss the rectification option for *Tsukuba*, since it is already rectified nor for the sequence *Turntable houses* because here too many points lie on unmoved parts of the scene and invariance against rotation cannot be achieved for them. From Table 6.6, where efforts have been made to reduce the number of outliers, it becomes clear that the number of outliers for epipolar tracking is usually slightly smaller than for simultaneous tracking. Probably, the main reason lies in gross errors in single images. For simultaneous tracking, the only possibility to sort out points is to decrease $\varepsilon_{\max}$, in other words, the effect of gross errors can be distributed across all images preventing the point from being discarded during tracking. Also, the interpolation errors for (optional) image rectification and gradient computation as well as camera uncertainties cannot be corrected geometrically (i. e. by reprojection errors). Since in pairwise tracking gross errors in single images are detected and eliminated right away, we will use epipolar tracking as our default option.

We are also interested in the locations of the lost points and incorrectly tracked points in the images. Figures 6.4-6.7, on the left, show the already available features, depicted by orange points, and, on the right, the newly tracked features (yellow), the lost features (cyan circles) and the features tracked incorrectly (red diamonds). As could be expected, most of the lost points lie near occlusions; this is not really surprising, because only one part of the template window is seen in the new image and the other part changes from image to image. This problem can be partly solved by considering cost functions other than the c in (4.11) or norms other than L_2 for weighting the entries within windows, but we let that be a topic for future work. The few outliers lie in the weakly textured regions; here the cost function does not have a clear minimum and so the result is not reliable. One can apply heuristics as described in [29] and in Sec. 4.4.1 in order to eliminate outliers, but we do not consider these options here.

Table 6.6: Results of tracking characteristic points for the data set *Infrared* with seven images, variable window size (win) and rectification option (opt.r). The total number of points was 900 and the standard KLT-tracking with window size 11 and image pyramid levels 5 yielded 583 points. See also [28].

meth.	KLT-epi, opt.r = 0				KLT-epi, opt.r = 1			
win	7	11	15	19	7	11	15	19
total	764	807	821	813	616	709	757	770
cor.	487	530	545	538	416	474	593	510
incor.	0	0	1	10	0	0	1	4
lost	96	53	37	35	167	109	79	69
meth.	simultaneous, opt.r = 0				simultaneous, opt.r = 1			
win	7	11	15	19	7	11	15	19
total	985	995	995	971	957	987	966	942
cor.	571	575	576	564	563	570	557	550
incor.	2	3	3	11	4	6	12	14
lost	10	5	4	8	16	7	14	19

Conclusion

In the current version of our implementation, we use epipolar tracking as a default option. The reason is that the number of outliers is usually lower than in the case of simultaneous tracking and camera uncertainties are better taken into account during the final triangulation step. In the future work, we will restructure the simultaneous tracking algorithm: first by filtering out, by means of radiometric differences, the images where occlusions are probable and second by taking camera uncertainties into account.

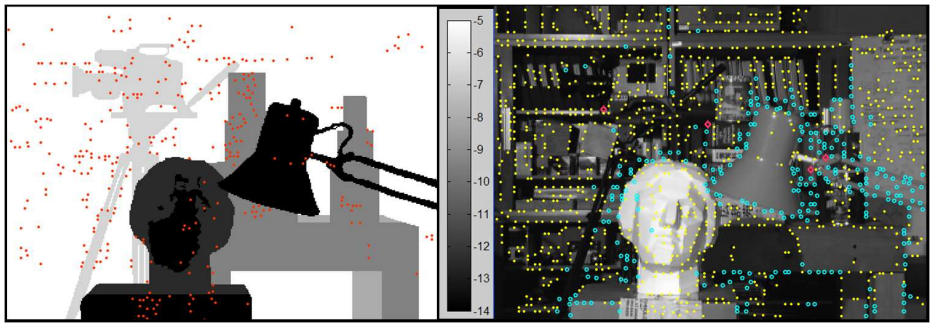

Figure 6.4: Left: The ground truth result of the benchmark data set *Tsukuba* needed for Sec. 6.3 with the original point cloud colored in orange. Middle: Disparity scale bar. On the right, the reference image with results of epipolar tracking. Points with disparity values correctly assigned by epipolar tracking are depicted by yellow dots, the lost points by cyan circles and outliers by red diamonds. See also [28].

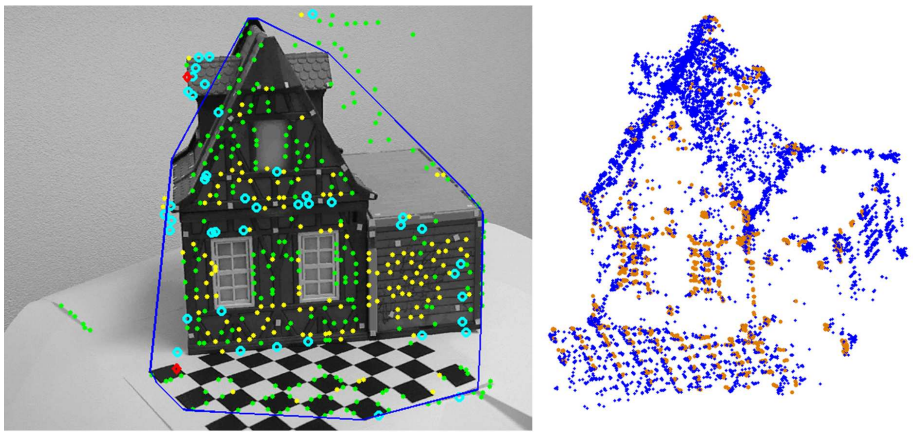

Figure 6.5: Left: The reference image of a subsequence of the data set *Turntable houses* with the results of epipolar tracking. Points with disparity values correctly assigned by epipolar tracking are depicted by yellow dots, the lost points by cyan circles and outliers by red diamonds. Points lost in the standard KLT tracking algorithm are depicted by green dots. Right: A view of the 3D-point cloud with already available points marked in orange.

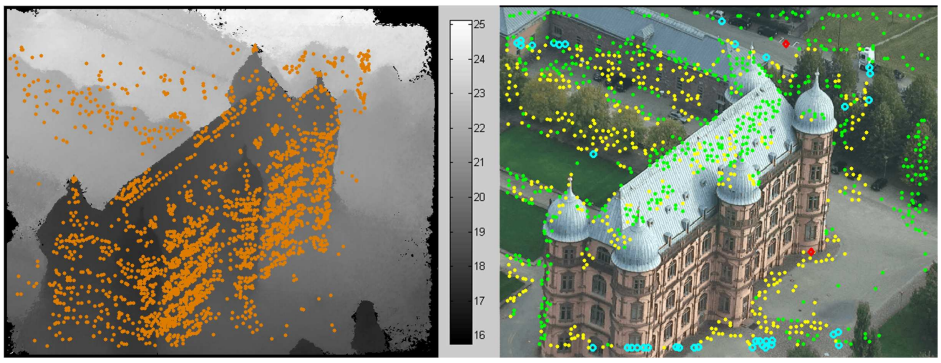

Figure 6.6: Left: the median-based depth map will be our ground truth of a subsequence of the data set *Gottesaue* in Sec. 6.3. The original point cloud is colored in orange. Middle: Depth scale bar. On the right, the reference image with results of epipolar tracking. Points with disparity values correctly assigned by epipolar tracking are depicted by yellow dots, the lost points by cyan circles and outliers by red diamonds. Points lost in the standard KLT-tracking algorithm are depicted by green dots.

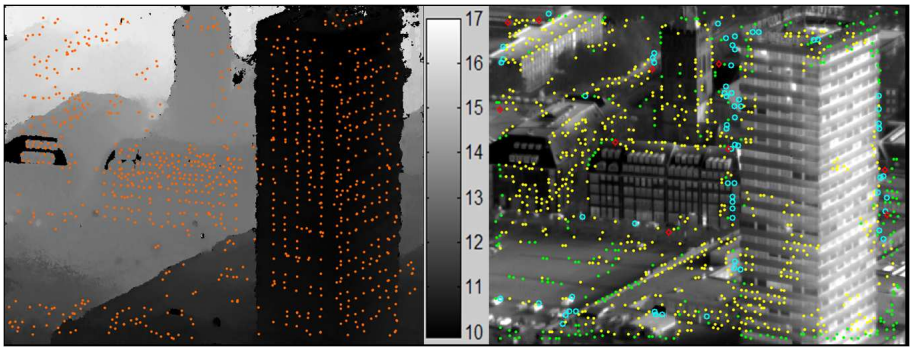

Figure 6.7: Left: The median-based depth map will be our ground truth for a subsequence of the data set *Infrared* in Sec. 6.3. The original point cloud is in orange. Middle: Depth scale bar. On the right, the reference image with results of epipolar tracking. Points with disparity values correctly assigned by epipolar tracking are depicted by yellow dots, the lost points by cyan circles and outliers by red diamonds. Points lost in the standard KLT-tracking algorithm are marked by green dots.

6.3 Dense reconstruction

This section will illustrate dense reconstruction of selected subsequences of video data with which we deal. We will structure this section in a manner similar to what we did in Chapter 4, first handling the binocular case (Sec. 6.3.1) and then multi-view reconstruction (Sec. 6.3.2). A subject of particular interest will be the automatic choice of the smoothness parameters λ_1, λ_2, covered in Sec. 6.3.3.

6.3.1 Binocular case

For the benchmark data set with the ground truth depth map shown in Fig. 6.4, the evaluation is carried out analogously to the previous section and we followed the choice of the authors of [115] to measure the number of incorretly tracked pixels, which we denote by $N_{\mathcal{B}} = \sum_{\mathcal{B}} 1$, as a function of different parameters. Alternatively, one can compute the average sum of relative depth deviations, denoted by $\varepsilon_{\mathcal{B}} = \sum_{\mathcal{B}} \varepsilon$, with $\mathcal{B}$ and ε defined in Eq. (6.1). For the data sets *Infrared* and *Gottesaue*, we chose the ground truth to be the median depth map using the methods of Sec. 4.5.2. This method is very robust – the reason that justifies us to take it as a ground truth – but also very time-consuming since semi-global optimization must be performed altogether $2K$ times (with cross-check as in (3.2), and $K+1$ number of images). We show in each of Figs. 6.8, 6.9, and 6.10, a typical result of the disparity estimation computed for the sequences *Tsukuba*, *Gottesaue*, and *Infrared*, respectively, with a local method supported by triangular meshes. One can see the two typical sources of errors: either too much noise makes it impossible to assign a triangle as consistent with the surface or a triangle is spuriously declared as surface-consistent. For the binocular case, this is especially visible in Fig. 6.9, where the stripes on the roof – which go perpendicular to the epipolar lines – provoke too many mismatches that cannot be corrected by the evaluation on triangles. As we will see later, this situation will be fairly seldom for the local algorithm applied to multi-view configurations because the pixels in somewhat textured area will be helped out of local minima by redundant views.

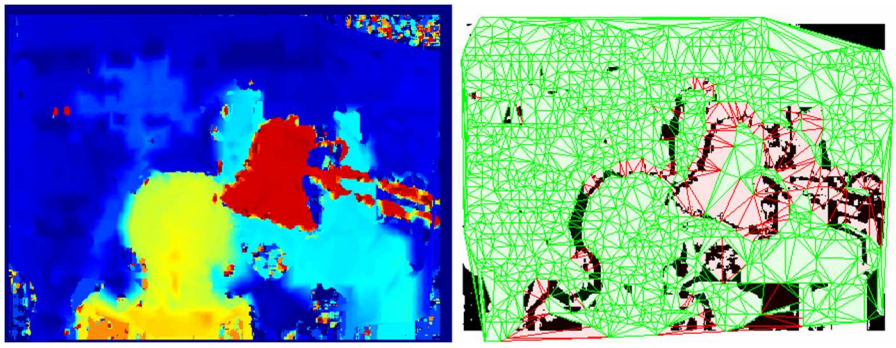

Figure 6.8: Top left: Illustration of the disparity map computed by the local algorithm from images $\mathcal{I}_{3,3}, \mathcal{I}_{3,4}$ of the sequence *Tsukuba*. Top right: Evaluation of the result on the left with incorrect matches depicted in black. The triangles consistent with the surface are marked in green, those inconsistent with the surface in red. The ground truth result is depicted in Fig. 6.4, p. 90, left.

In the next step, we turn our attention to global and semi-global methods. Figures 6.11, 6.12 and 6.13 illustrate typical results for the sequences *Tsukuba*, *Gottesaue*, and *Infrared*, respectively, of the graph-cuts-algorithm implementation of [81] (top) and the semi-global optimization due to Hirschmüller as in [67] (bottom). In the graph cuts algorithm, the

data-cost function was given by the truncated SSD, as in Eq. (2.4), $p = 2$, the smoothness function was given by (2.16), where

$$\lambda(i, i') = \lambda_1 U(u \geq 8) + 3\lambda_1 U(u < 8), u = \min\left(|\mathcal{I}(\mathbf{x}) - \mathcal{I}(\mathbf{y})|, |\mathcal{I}'(\mathbf{x}') - \mathcal{I}'(\mathbf{y}')|\right),$$

and points $\mathbf{x} = (x, y)$ with the property $d(x, y) = d(x + 1, y) + 1$ were marked as occluded. In the implementation of the semi-global algorithm for the binocular case, mutual information was our the data-function; also the cross-check test according to (3.2) followed by the evaluation on triangles by the methods of Sec. 4.5.1 was carried out.

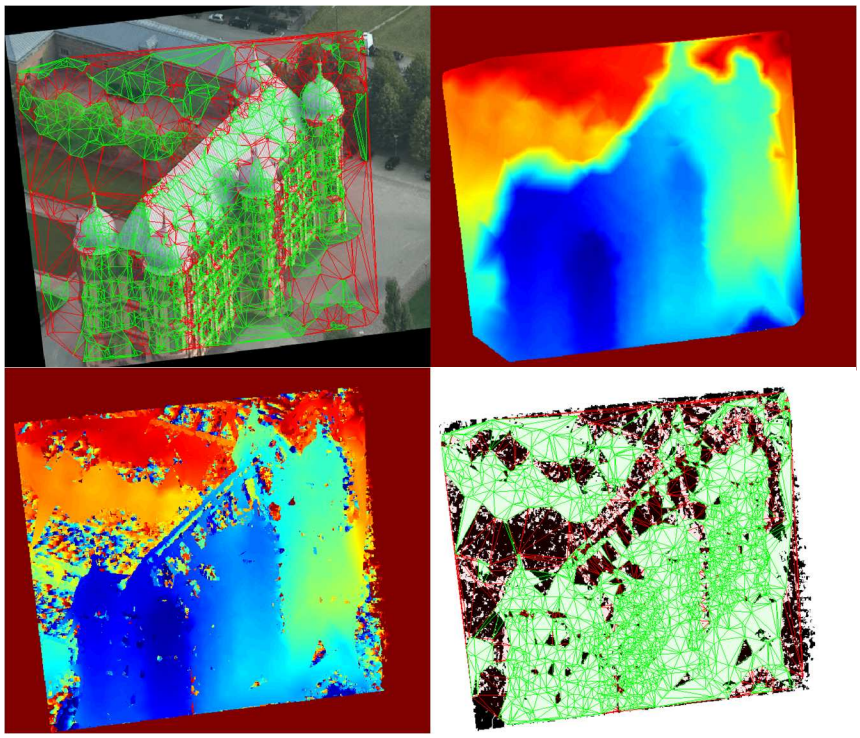

Figure 6.9: Top left: Part of the rectified reference image from the sequence *Gottesaue*. Triangles declared as consistent with the surface by the local algorithm are colored in green while inconsistent triangles are colored in red. Right: Disparity map $\mathcal{D}_\mathcal{T}$ produced by the triangular interpolation described in Sec. 4.3.1. Bottom left: a typical result $\mathcal{D}_{loc}$ of the local depth computation. Bottom right: evaluation of $\mathcal{D}_{loc}$ on the left with incorrect matches depicted in black and triangles consistent and inconsistent with the surface in green and red, respectively.

The next several figures show quantitative evaluations of the binocular dense reconstruction. The global results, demonstrated for the three sequences *Tsukuba*, *Gottesaue*, and *Infrared*, in Figs, 6.14, 6.15, and 6.16, respectively, are important for understanding, among other things, the performance of the graph cuts algorithm (always top row) in comparison with the performance of semi-global matching (bottom row). The local results will be covered in Sec. 6.3.2 because of a strong analogy with the multi-view case.

One can see that the graph-cuts algorithm, despite its positive properties to perform well near occlusions and in regions of repetitive patterns of texture, is barely suitable for computing disparity maps for the sequences *Gottesaue* and *Infrared*. In the latter sequence, application of the graph-cuts algorithm even deteriorates the results of the local algorithm supported by triangular meshes, while, in the first, it improves them only slightly. The

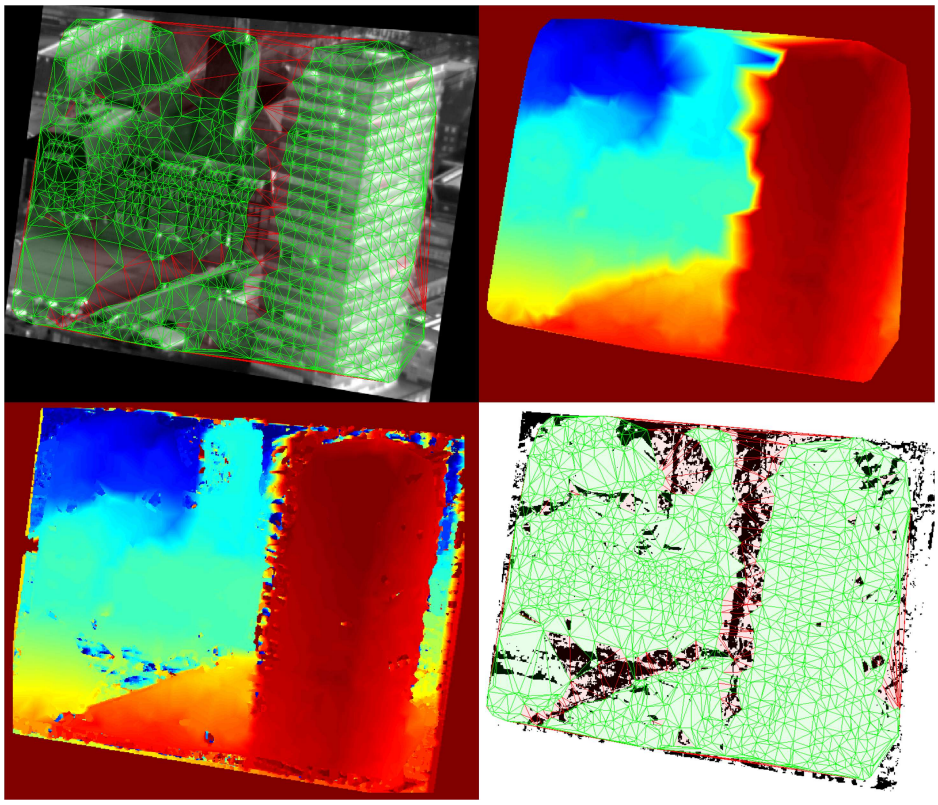

Figure 6.10: Top left: Part of the rectified reference image from the sequence *Infrared*. Triangles declared consistent with the surface by the local algorithm are colored in green while inconsistent triangles are colored in red. Right: Disparity map $\mathcal{D}_{\mathcal{T}}$ produced by the triangular interpolation described in Sec. 4.3.1. Bottom left: a typical result $\mathcal{D}_{loc}$ of the local disparity computation. Bottom right: evaluation of $\mathcal{D}_{loc}$ on the left with incorrect matches depicted in black and triangles consistent and inconsistent with the surface in green and red, respectively.

idea behind the graph-based algorithm based on alpha-expansions is to overwrite a set of pixels of a given initial disparity map $\mathcal{D}$ by a scalar value α. In other words, if we have a pixel with disparity label α in a textured region, this value will be propagated to neighboring untextured regions until no improvements take place. Hence a risk to fall into a local minimum is very high. The susceptibility of the algorithm towards fronto-parallel planes additionally aggravate this problem; and evaluation of triangles cannot actually solve it because the percentage of pixels that the algorithm recognizes to be consistent with the surface is rather low (see Fig. 6.12). As a result, the disparity values are likely to be grouped into segments whose borders are often drawn somewhere within textureless regions. The semi-global method can reduce the number of pixels with wrongly assigned disparities, especially if evaluation of triangles takes place, but for the remaining pixels (which are usually situated near occlusions, the values of the disparities are forced to be near to those of neighboring pixels or are interpolated linearly. Therefore the value of $\varepsilon_{\mathcal{B}}$ increases while $N_{\mathcal{B}}$ falls and we can state that occasional over-smoothing edges represents the main drawback of the semi-global method.

With respect to the choice of the smoothness parameter for the graph-based method, good experiences were made with the heuristic described in [79]. One can see a clear minimum in the number of pixels with incorrectly assigned disparity values in Fig. 6.14 which

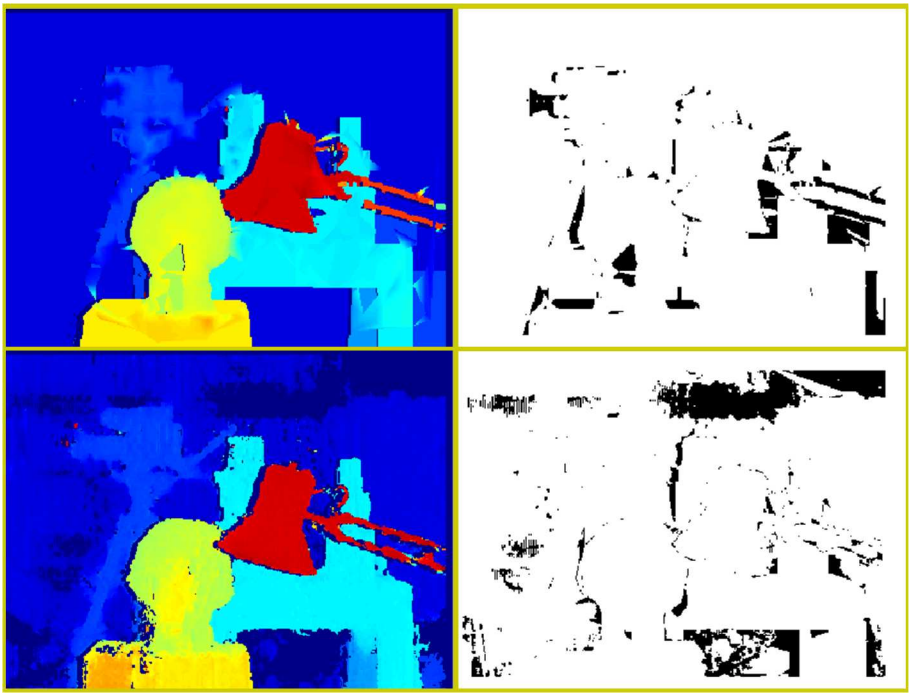

Figure 6.11: Top left: Illustration of the disparity map computed by the graph cuts algorithm [81] for the sequence *Tsukuba*. Top right: Evaluation of the result on the left with incorrect matches depicted in black. Bottom: Result and evaluation of the semi-global algorithm.

results from the automatically selected value of λ_1. In the case of data sets with less self-occlusions (for instance, Nadir flights over urban terrains), λ_1 can be chosen slightly larger than the automatically computed value. In the case of the semi-global optimization, we have two smoothness parameters. Both in binocular and multi-camera configurations, visually good results were obtained if the strategy to choose a moderate value of λ_1 (to admit slanted surfaces) and $\lambda_2 = 2\lambda_1$ was followed. We refer to Sec. 6.3.3, where the question of automatic choice of λ_1 for dynamic programming and semi-global optimization methods will be covered in a more detailed way.

Our next issue concerns reduction of the computing time by initialization of the graph-cuts algorithm. As one can see from Figs. 6.12 and 6.13, quantitative results of a *global* algorithm do not depend significantly on the initialization, so we are concerned here about the number of iterations in the process of computing the disparity map. Since we have here a random process, we carried out the energy minimization several times and computed the average number of iterations. The test data set was *Gottesaue* because the number of pixels in the images was larger than in other data sets and so the randomization effects of order of disparity values for alpha-expansions could be reduced. The correlation between the energy ratios at the beginning and at the end of the algorithm is indicated in Table 6.7. We see that a good initialization is equivalent to a low energy at the beginning of the graph-cuts algorithms and so, in the majority of cases and especially for larger values of smoothness parameter λ, less iterations are needed to reach a (local) minimum of the energy functional.

For the semi-global method, computation of *Mutual information* matching table from the triangular mesh and initialization with this result helps to produce comparable results as in the case of image pyramids as one can see from the blue and cyan curves in Fig. 6.15. This kind of initialization can thus be preferred to the computation of image pyramids proposed in [67].

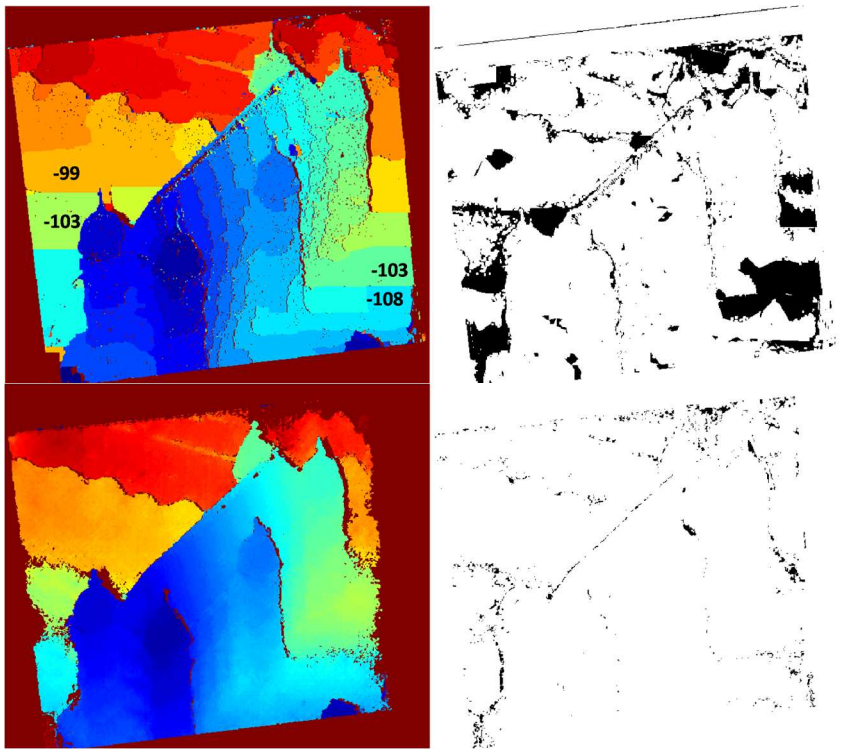

Figure 6.12: Top left: Illustration of the disparity map computed by graph cuts algorithm from two frames of the sequence *Gottesaue*. The discretization artifacts are very visible in the final result because no subpixel matching is performed. Top right: Evaluation of the result on the left with incorrect matches depicted in black. Typical problems emerging in this algorithm are shown by marking some disparity labels; of course no jumps in the disparity exist in the reality (see Fig. 6.6 above). Bottom, left and right: Result and evaluation of the semi-global algorithm.

Table 6.7: Correlation between the energy ratios at the beginning and the end of the graph cuts algorithm and the computing time, which is directly proportional to the number of iterations. Sequence *Gottesaue*, different smoothness parameters λ.

no init						
λ	100	200	300	400	500	600
E_0/E	0					
av. iter	12.7	10.3	10.05	9.15	8.05	6.85
	init $\mathcal{D}_{loc}$					
E_0/E	−0.43	0.12	0.29	0.36	0.44	0.48
av. iter	14.45	11.85	8.95	8.55	7.35	6.75
	init $\mathcal{D}_{\mathcal{T}}$					
E_0/E	−0.12	0.31	0.44	0.50	0.56	0.59
av. iter	11.6	8.25	8.65	8.5	6.9	6.05

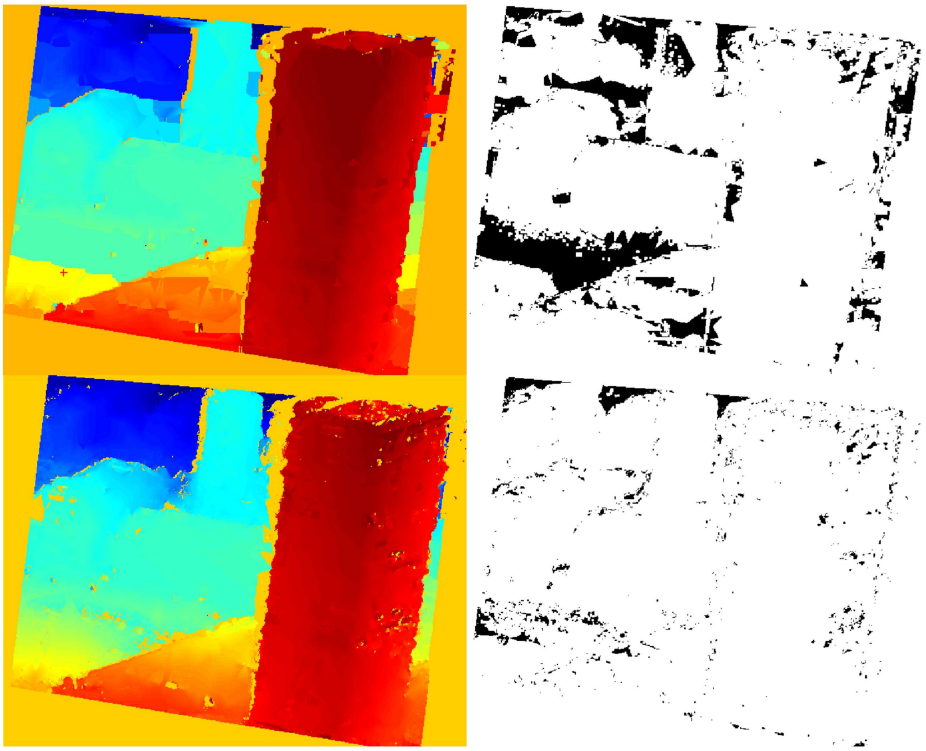

Figure 6.13: Top left: Illustration of the disparity map computed by graph cuts algorithm from two frames of the sequence *Infrared*. Top right: Evaluation of the result on the left with incorrect matches depicted in black. Bottom: result and evaluation of the semi-global algorithm.

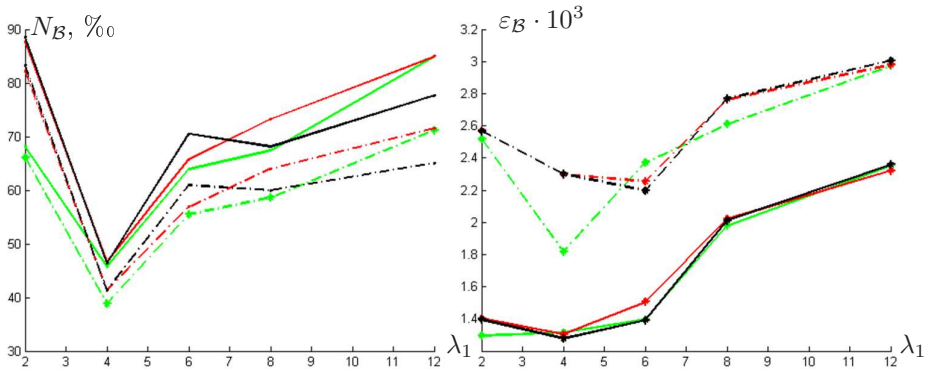

Figure 6.14: Results of disparity estimation for the sequence *Tsukuba* with the graph-cuts methods. Left: The ‰-value $N_{\mathcal{B}}$ of pixels with incorrectly assigned disparity values as a function of smoothness parameters λ_1 and the triangulation-based parameter γ. The choice for $\gamma = 0.75$ is always marked by solid lines and $\gamma = 1$ by dotted lines. The black, green and red curves represent results initialized with the local disparity map $\mathcal{D}_{loc}$, initialized with $\mathcal{D}_{\mathcal{T}}$ and without initialization, respectively. On the right, average error per pixel $\varepsilon_{\mathcal{B}}$ for all configurations described above. Quantitative analysis of this data set with the semi-global method will be performed for a multi-view configurations in the next section.

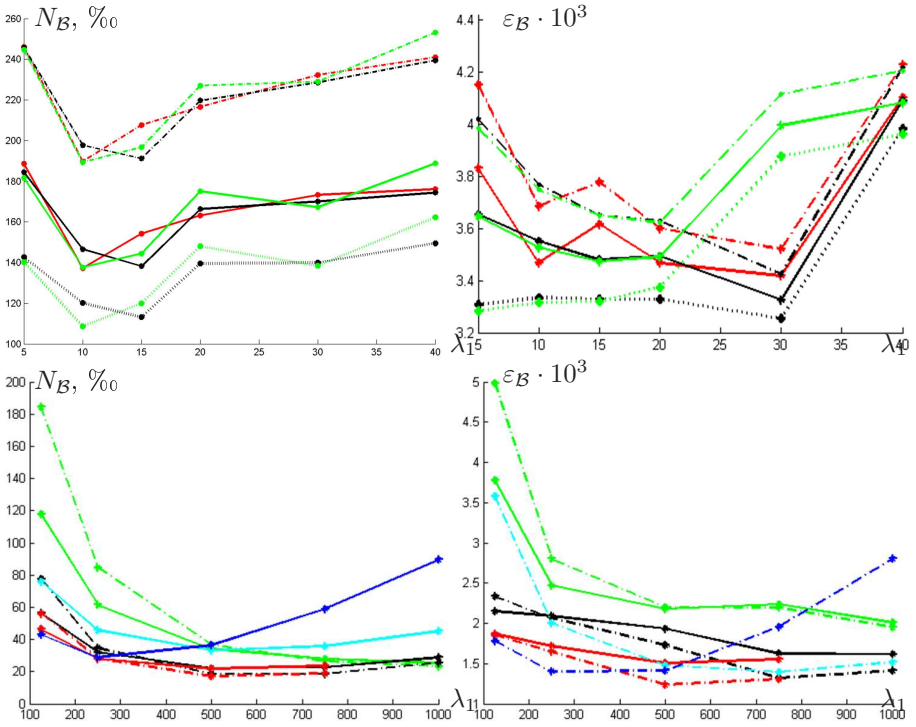

Figure 6.15: Results of disparity estimation for the sequence *Gottesaue* with non-local methods. Top left: Graph-guts algorithm: the ‰-value ($N_{\mathcal{B}}$) of pixels with incorrectly assigned disparity values as a function of λ_1. The black, green and red curves represent results initialized with the local disparity map $\mathcal{D}_{loc}$, initialized with $\mathcal{D}_{\mathcal{T}}$ and without initialization, respectively. The dashed, solid and dotted curves represent choices $\gamma = 0.5, \gamma = 0.75$ and $\gamma = 1.0$, respectively. Bottom left: Results for the semi-global algorithm. The ‰-value of $N_{\mathcal{B}}$ as a function of λ_1, where $\gamma = 0.75$ is always marked by solid lines and $\gamma = 1$ by dotted lines. Blue and cyan curves denote the results from the initialization as in [29] while all other curves use image pyramids and mutual information as the cost function. Green and cyan curves stem from the choice $\lambda_2 = \lambda_1$, black and blue curves stem from the choice $\lambda_2 = 2\lambda_1$ and the red curve from the choice $\lambda_2 = \min(4\lambda_1, 2047)$ (see explanation of Eq. (4.25)). On the right, average error $\varepsilon_{\mathcal{B}}$ per pixel for all configurations described above.

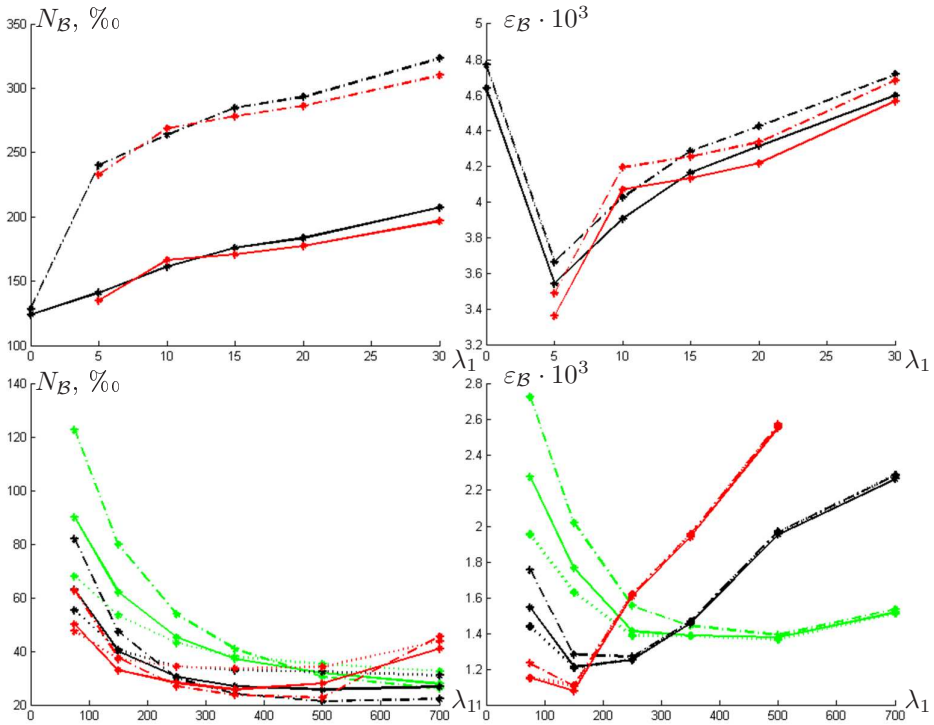

Figure 6.16: Results of disparity estimation for the sequence *Infrared* with non-local methods. Top left: Graph-cuts algorithm The ‰-value of $N_{\mathcal{B}}$ as a function of λ_1, where $\gamma = 0.67$ is marked by solid lines and $\gamma = 1$ by dashed lines. The black curves denote the results with initialization and red curves without. Bottom left: Results for the semi-global algorithm. Green curves stem from the choice $\lambda_2 = \lambda_1$, black curves stem from the choice $\lambda_2 = 2\lambda_1$ and the red curve from the choice $\lambda_2 = \min(4\lambda_1, 2047)$. The dotted, solid and dashed curves represent the choices $\gamma = 0.5, 0.75$ and 1.1, respectively. On the right, average error per pixel $\varepsilon_{\mathcal{B}}$ for all configurations described above.

6.3.2 Multi-view configurations

In order to demonstrate that the matching ambiguities in regions of repetitive patterns of texture and near occlusions can be resolved by using redundant views, we now replace the binocular configuration of the previous section by the multi-view configuration made up by five images in data set *Tsukuba* and seven images in both data sets *Gottesaue* and *Infrared*. The ground truth result remains the same as in the last section, but we changed slightly the evaluation criterion for data sets *Gottesaue* and *Infrared* in order to take into account the rather vast depth ranges which vary from several dozens to at least several hundreds of meters. We say that a pixel $\mathbf{x}$ is assigned to $\mathcal{B}$ if the deviation of $d_{\mathbf{x}}$ from the ground truth $d_{gt}(\mathbf{x})$ value is more than 5%, in other words:

$$\varepsilon = |d_{gt}(\mathbf{x}) - d(\mathbf{x})|/|d_{gt}(\mathbf{x})| > 0.05,$$

and the definitions for $N_{\mathcal{B}}$, $\varepsilon_{\mathcal{B}}$ remain the same.

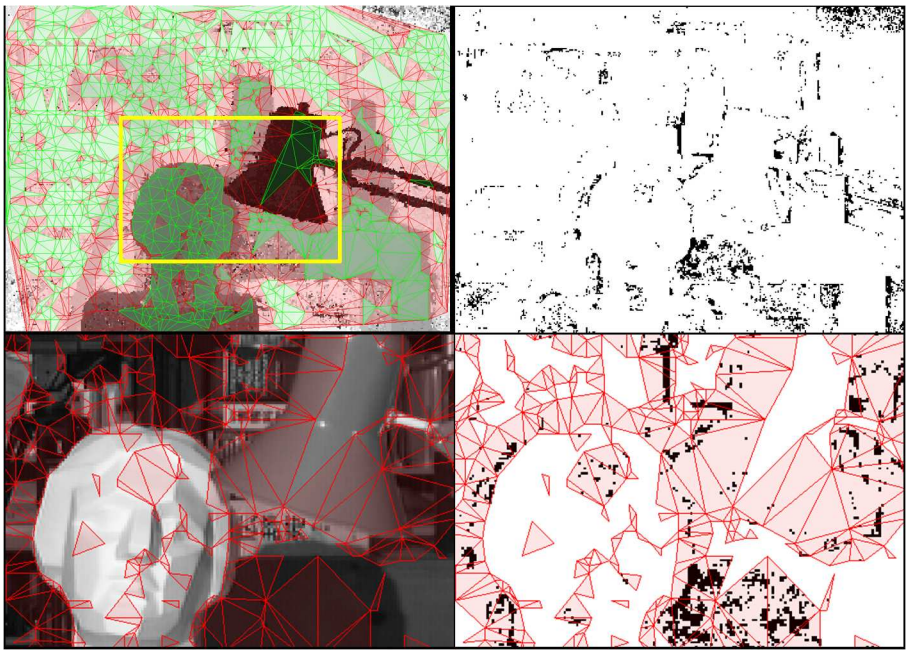

Figure 6.17: Top Left: Triangular mesh and the result of the local disparity map of the data set *Tsukuba* from five images and the mesh rendered from the enriched point set where the triangles consistent and inconsistent with the surface are marked in green and red, respectively. Top right: Evaluation of the result on the left by means of the ground truth disparity map depicted in Fig. 6.4, all matches where the difference exceeds one pixel are depicted in black. Bottom left: Part of the reference image (denoted by yellow rectangle above) where triangles inconsistent with the surface are given red color. Bottom right: evaluation of this part, almost all wrong matches lie inside of red triangles.

Extended tests were carried out for 9 local parameters (number of cameras K, window size, cost function which we denote here by $\varepsilon_{\max}$, rectification option opt.r and interpolation option opt.i, the parameter ε_y responsible for compensating errors due to uncertainties in camera positions as well as triangulation-based smoothness terms A, σ, γ) and two global parameters λ_1, λ_2 for semi-global optimization. Many of these parameters were already object of related research (see [115] and references therein), therefore we will not vary here the

value of every parameter by letting fixed all others (and this for each data set), but restrict ourselves to describing in the graphics below the influence of the most important ones, especially those related to triangular meshes and global methods. For the other parameters, we give only resuming observations.

We show in Figs. 6.17, 6.18, and 6.19 typical results of the local approach with considering the local smoothness term E_T from Eq. (4.21) for the data sets *Tsukuba, Gottesaue*, and *Infrared*, respectively. The result of applying the local triangulation-based smoothness terms from the *enriched* point set (as the result of Sec. 6.2) is shown together with the triangulated point set, triangles consistent and inconsistent with the surface (colored in green and red, respectively), and binarized absolute differences from the ground truth. For the non-local optimization algorithms of dynamic programming and multi-view semi-global optimization, we show typical results of the multi-view dense reconstruction for the three data sets in Figs. 6.20, 6.21 and 6.22, respectively.

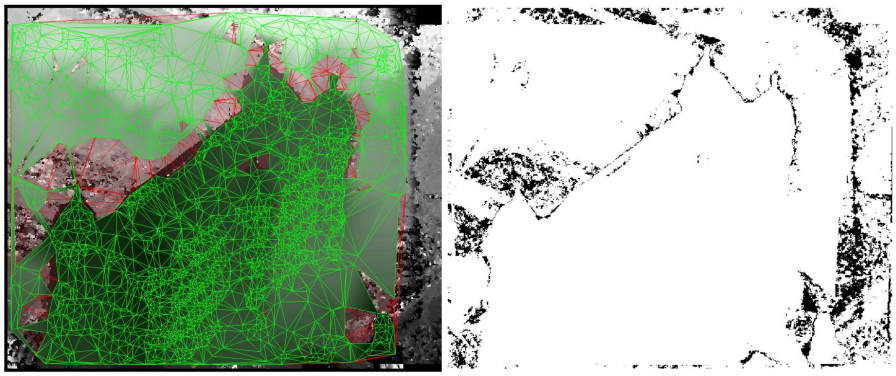

Figure 6.18: Left: Triangular mesh and the local result of the depth map of the data set *Gottesaue* from seven images and the mesh rendered from the enriched point set where the triangles consistent and inconsistent with the surface are marked in green and red, respectively. Right: Evaluation of the result on the left with incorrect matches depicted in black.

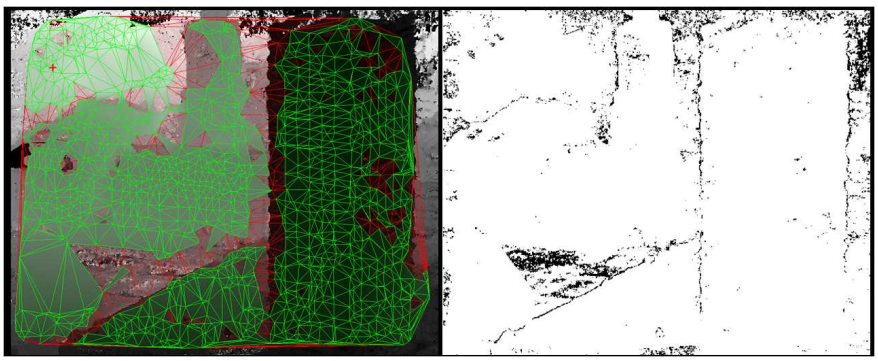

Figure 6.19: Left: Triangular mesh and the local result of the depth map of the data set *Infrared* from seven images and the mesh rendered from the enriched point set where the triangles consistent and inconsistent with the surface are marked in green and red, respectively. Right: Evaluation of the result on the left with incorrect matches depicted in black.

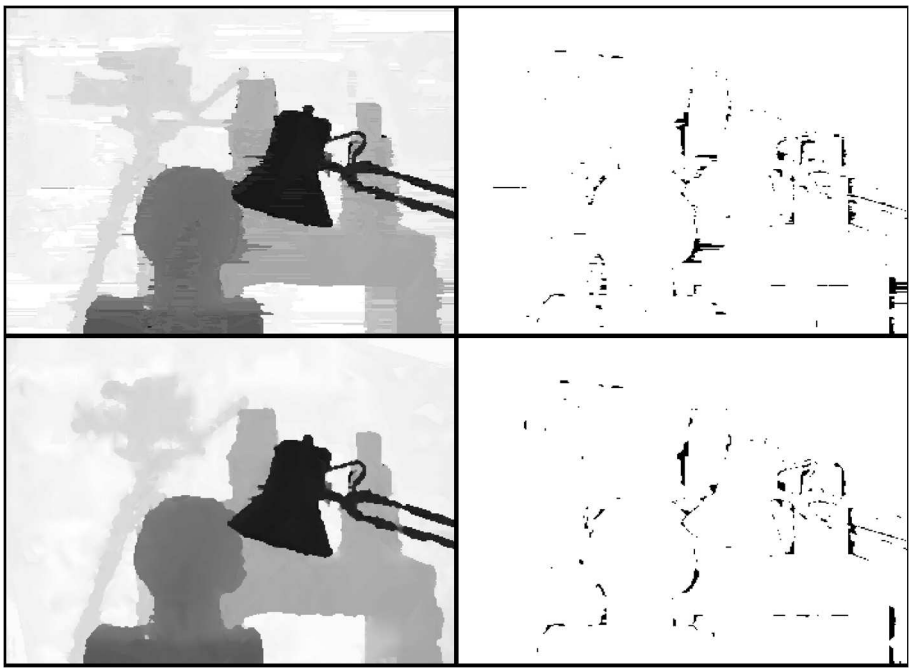

Figure 6.20: Top left: The result of energy minimization with dynamic programming for the data set *Tsukuba*, five images, window size = 3. Top right: Evaluation of the result on the left with incorrect matches depicted in black. Bottom left and right: Result and evaluation of the semi-global algorithm.

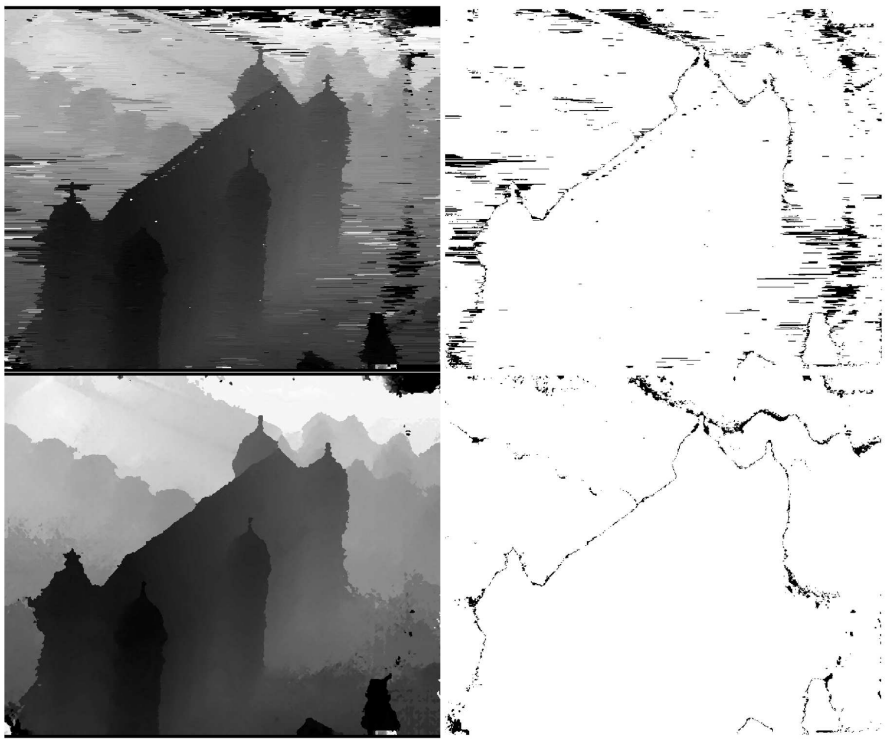

Figure 6.21: Top left: Result of energy minimization with dynamic programming for the data set *Gottesaue*, seven images. Top right: Evaluation of this result with incorrect matches depicted in black. Bottom left and right: Result and evaluation of the semi-global algorithm.

In Figs. 6.23, 6.24, and 6.25, dependence of the results on the window size, cost function, A, σ and γ for data sets *Tsukuba, Gottesaue*, and *Infrared*, respectively, is represented. The red and green curves stand for the truncated SAD from Eq. (2.4) with $\varepsilon_{\max} = 15$ and 40, respectively. The blue curves stand for the NCC (2.6) and the black curves for the simplification (2.7). A smaller percentage of incorrectly reconstructed pixel makes clear that for a video sequence, it makes more sense to use (truncated) SAD as a cost function. A possible explanation lies in the parameters a and b of (2.5). These additional degrees of freedom allow a more flexible distribution of gray values within windows, but their values must also satisfy (at least a piecewise-)smoothness condition because the reflection coefficient of the material surface is made of as well as the angle between normal vector of a point and a camera plane are constant in the whole regions. As a consequence, Eq. (2.6) is implicitly over-parametrized and therefore blue and black curves lie above the red and green ones. In other experiments, which go beyond the scope of this work, we were able to ascertain a slight improvement of the results after activating opt.r or opt.i (the bilinear interpolation instead of rounding) while increasing ε_y (see Eq. (4.10), p. 53) does not influence much the results. Finally, augmenting the number of cameras K and the window size *win* is helpful to reduce $N_{\mathcal{B}}$ and $\varepsilon_{\mathcal{B}}$ although the computing times clearly increase.

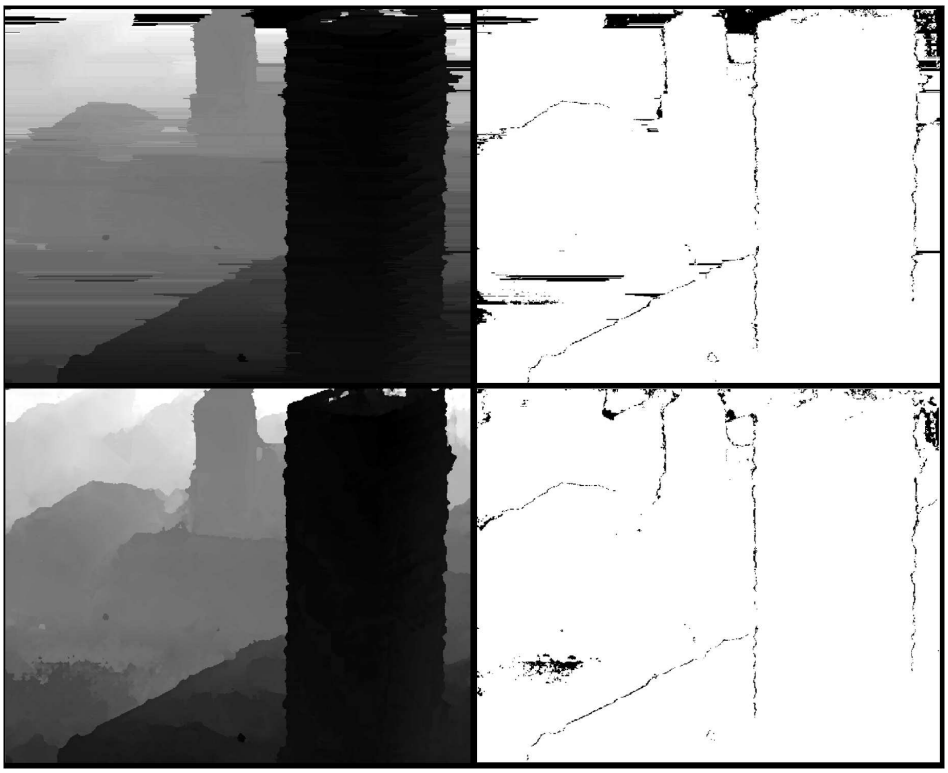

Figure 6.22: Top left: The result of energy minimization with dynamic programming for the data set *Infrared*, seven images, window size = 3. Top right: Evaluation of the result on the left with incorrect matches depicted in black. Bottom left and right: Result and evaluation of the semi-global algorithm.

We go on by investigating the influence of the triangulation-based smoothness terms whose presence usually not only increases the accuracy but also smoothers the effects of too small K, or opt.r = 0. As one can see from Figs. 6.9 and 6.18, there are almost no mismatches in triangles consistent with the surface. If $\gamma(T) < 1$, then all pixels within T

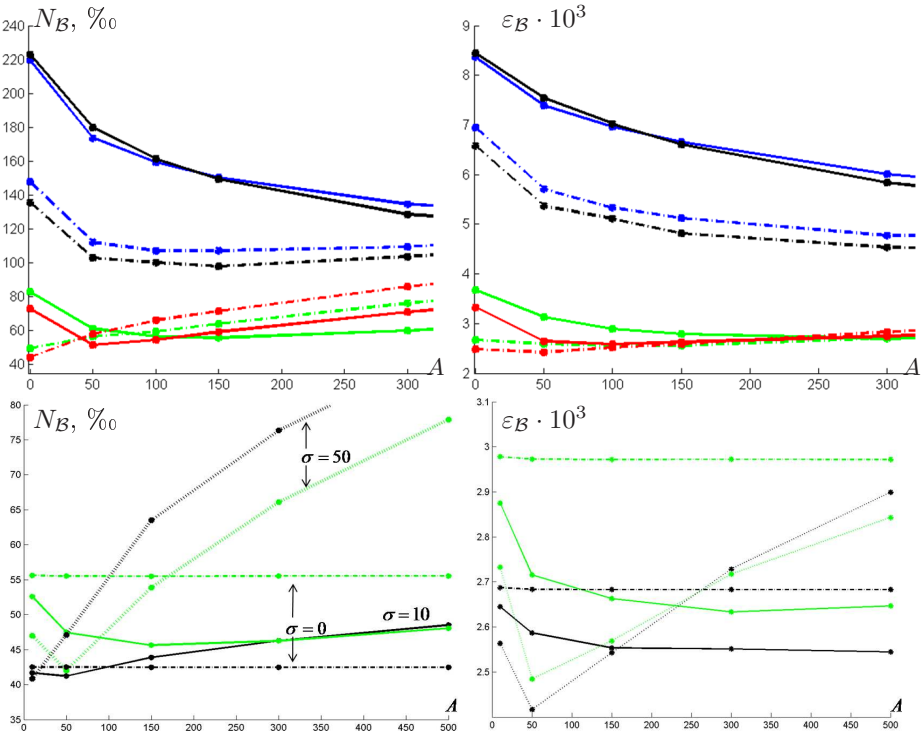

Figure 6.23: Top left: The ‰-value ($N_{\mathcal{B}}$) of pixels with incorrectly assigned depth values as a function of A, cost function and γ, data set *Tsukuba*, window size = 5, opt.r = 0 and $\sigma = 50$. The dashed curves correspond to the value $\gamma = 0.67$, solid curves for $\gamma = 1$. The behavior for different cost function: red and green curves for (2.4) with $\varepsilon_{\max} = 15$ and 40, respectively, blue curves for (2.6) and the black curves for (2.7). Bottom row: Variation of σ and γ. Black curves correspond to $\gamma = 0.75$ and green curves to $\gamma = 0.95$, the dashed, solid and dotted lines correspond to different choices of sigma ($\sigma = 0, 10, 50$, respectively). On the right, average error $\varepsilon_{\mathcal{B}}$ per pixel for all configurations described above.

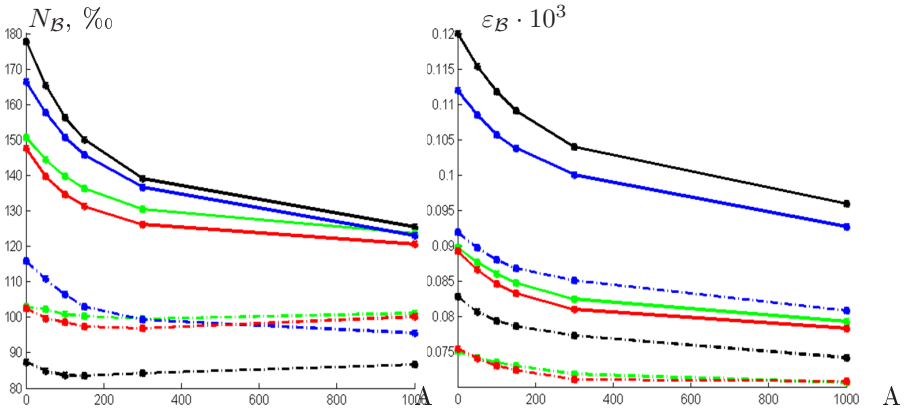

Figure 6.24: Top row, left: The ‰-value of $N_{\mathcal{B}}$ as a function of A, cost function and γ, data set *Gottesaue*, window size = 5, opt.r = 0, number of images = 5 (images 1, 2, 4, 6, 7 used) and $\sigma = 50$. The dashed curves correspond to the value $\gamma = 0.67$, solid curves for $\gamma = 1$. The behavior for different cost function: red and green curves for truncated SAD from Eq. (2.4) with $\varepsilon_{\max} = 15$ and 40, respectively, blue curves for NCC from Eq. (2.6) and the black curves for (2.7). On the right: Average error $\varepsilon_{\mathcal{B}}$ per pixel for all configurations described above.

are assigned depth values d_T; this assumption is reasonable, because for large point clouds nearly homogeneously distributed in the image, the number of triangles compatible with the surface will normally be quite high. One can see the noisy distribution of depth values within red triangles and the smooth (and correct) depth values by green triangles in Figs. 6.17 and 6.19. Declaring a triangle consistent with the surface can be further eased by adding a triangulation-based smoothness term E_T term of the form (4.14) or (4.22); this approach proves to be very efficient at a pixel $\mathbf{x}$ in a low textured area (see Fig. 6.23 and Fig. 6.25) where the cost function is likely to yield quite similar results for several depth labels. In this case, a support for the plausible value $d_{T,\mathbf{x}}$ can help to assign correct depth values with subpixel accuracy. Of course, if T is inconsistent with the surface (i. e. when one or two of its vertices lie on an occlusion edge), then T will be mapped in a wrong way; therefore the terms $N_{\mathcal{B}}, \varepsilon_{\mathcal{B}}$ become larger if σ and A are unreasonably high. The results of triangular interpolation become indeed worse for very high σ and A, as one can see, for example, from the dotted lines in Fig. 6.23 where *too many* triangles were declared consistent with the surface.

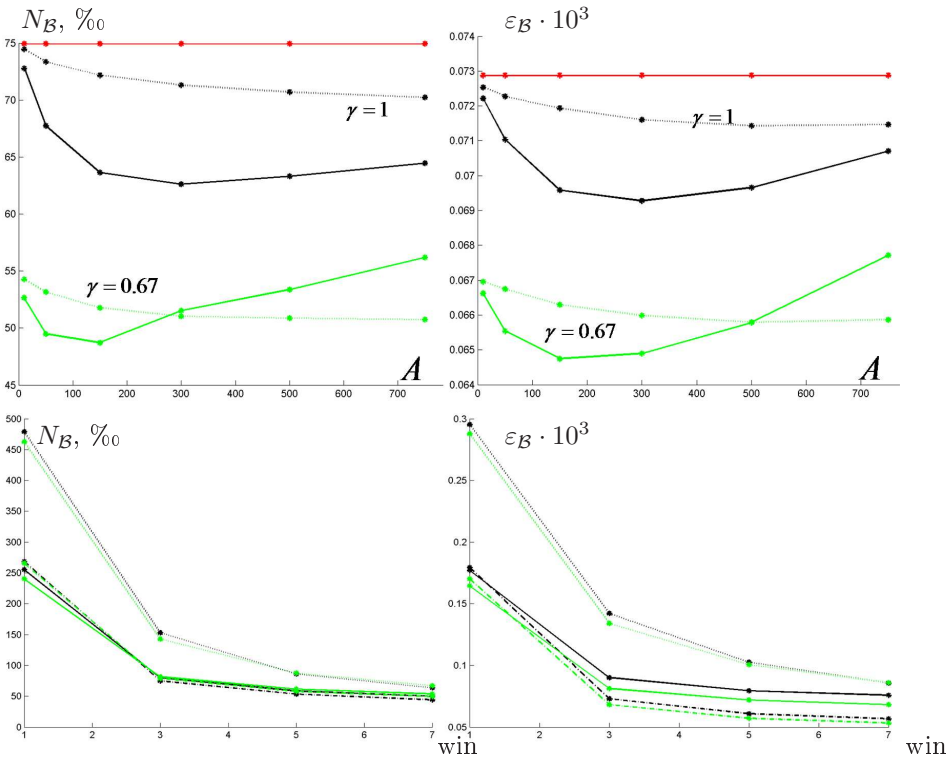

Figure 6.25: Left: The ‰-value of $N_{\mathcal{B}}$ as a function of A, σ and γ, data set *Infrared*, window size = 3, opt.r = 0, number of images = 7. The green curves correspond to the value $\gamma = 0.67$, black curves for $\gamma = 1$ and the red horizontal line shows the result without consideration of triangulation-based smoothing. The behavior for different σ-values ($\sigma = 10$: dashed line or $\sigma = 50$: solid line) is illustrated as well. Bottom left: $N_{\mathcal{B}}$ (in ‰) as a function of the window size. The different curves are shown for opt.r = 0 (green line) and opt.r = 1 (black line) as well as different choices of images: for the dashed line, all 7 images were considered, for the solid line, images 1, 2, 4, 6, 7 were used and for the dotted line, only images 1, 4, 7. The reference image was always image 4 and the number of levels for disparity computation was the same for each experiment. On the right, top and bottom: Average error $\varepsilon_{\mathcal{B}}$ per pixel for all configurations described above.

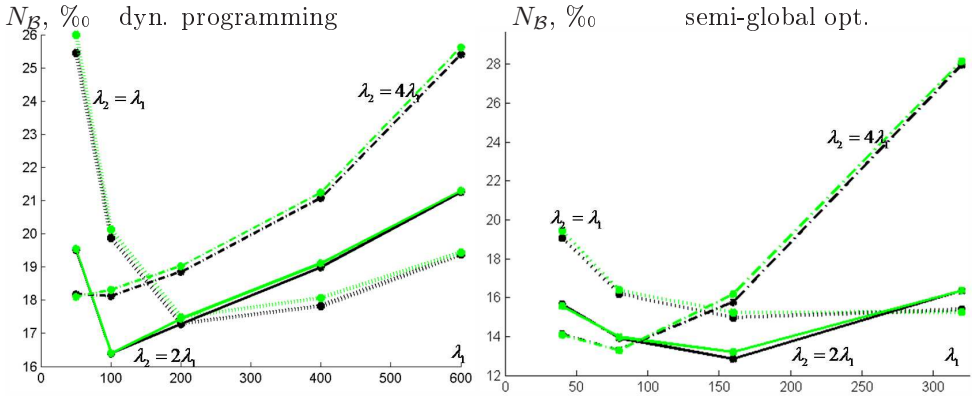

Figure 6.26: Left: The ‰-value of $N_{\mathcal{B}}$ as a function of λ_1 for dynamic programming in the data set *Tsukluba*. Right: Results of semi-global matching. The dotted, solid and dashed curves correspond to different choices (1, 2, 4, respectively) for the ratio λ_2/λ_1. Furthermore, $\gamma = 0.67$ for black curves and $\gamma = 0.95$ for green curves.

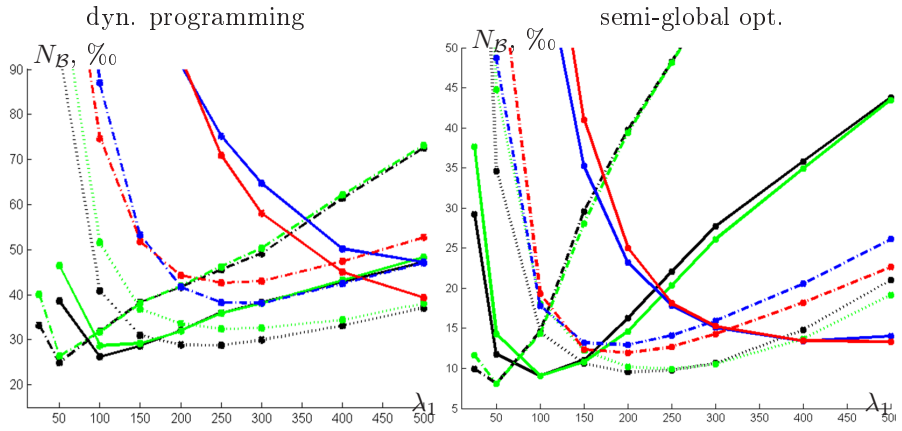

Figure 6.27: Left: The ‰-value of $N_{\mathcal{B}}$ as a function of λ_1 for dynamic programming in the data set *Gottesaue*. Right: Results of semi-global matching. The dotted, solid and dashed curves correspond to different choices (1, 2, 4, respectively) for the ratio λ_2/λ_1. Furthermore, $\gamma = 0.5$ for black curves and $\gamma = 0.95$ for green curves with the cost function initialized by a truncated SAD (see Eq. (2.4), $\varepsilon_{\max} = 40$), blue curves stand for NCC in (2.6) and red curves for (2.7). For blue and green curves, the value of γ was always 0.5.

Since the improvements of our local algorithm (for a fixed point set) are limited by the number of triangles consistent with the surface and it depends on the complexity of the scene how far we can go with increasing σ and A and also decreasing γ, a further optimization can be achieved by applying non-local algorithms. In Figs. 6.26, 6.27, 6.28, values of $N_{\mathcal{B}}$ for dynamic programming and the semi-global algorithm are presented (since the results for $\varepsilon_{\mathcal{B}}$ show a similar behavior) for the data sets *Tsukuba*, *Gottesaue*, and *Infrared*, respectively. For all data sets, we varied the values of λ_1 and ratios λ_2/λ_1. On the other hand, we varied γ in Fig. 6.26, γ and the cost function in Fig. 6.27 and the number of cameras in Fig. 6.28. The results confirm that anything what improves the performance of a local algorithm, will also do of a global one. We decided to use for all data sets window size 3 (because considering only pixels themselves without neighbors results in a rapid increase of the number of mismatches

and larger windows make increase computing time without very significant improvements of the results), opt.r was set to zero (because image transformations take extra computing time) and the number of images was five for *Tsukuba* data set and seven for other data sets.

We can see from illustrations and graphics that dynamic programming can eliminate most outliers within epipolar lines, but since epipolar lines are usually differently over-smoothed, there are visually unpleasant streaking artifacts in the result. Applying the semi-global algorithm with 16 smoothing directions allows eliminating these artifacts and so the number of mismatches (which are mostly made up by points near occlusion edges and far away from the camera positions) usually tends against zero (compare Figs. 6.20-6.22 for visualization, Figs. 6.26-6.28 for quantitative evaluation). The ‰-values for $N_{\mathcal{B}}$ decrease from around 45 (local method) to 20 (dynamic programming) and to 15 (semi-global matching) for both data sets. For the data set *Tsukuba*, the lowest values of $N_{\mathcal{B}}$ are around 1.3% and 2.9% (with and without correction for rounding errors, respectively). This means that our method is one of the best among those mentioned by [115] and so a multi-view configuration supported by a dynamic or, even more, a semi-global algorithm outperforms most of the two-camera algorithms.

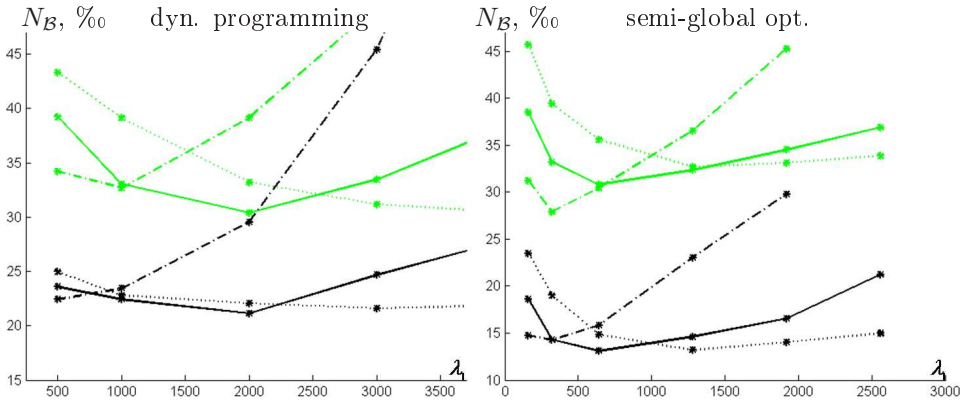

Figure 6.28: Left: The ‰-value of $N_{\mathcal{B}}$ as a function of λ_1 in the data set *Infrared* for dynamic programming. On the right, results for semi-global matching. The black curves result from considering all 7 images, the green curves from considering only images 1, 4, 7. The dotted, solid and dashed curves are for ratios $\lambda_2/\lambda_1 = 1, 2, 4$, respectively.

6.3.3 Automatic choice of smoothness parameters

Our next issue will be the automatic choice of smoothness constants λ_1 and λ_2. For the data sets *Tsukuba* and *Infrared*, we write down the best ranges of λ_1 (with respect to $N_{\mathcal{B}}$ and $\varepsilon_{\mathcal{B}}$ and ratios λ_2/λ_1. We can clearly see from Eq. (4.27) that the automatic choice of smoothness parameter must depend on the cost/aggregation function c. As a consequence, Table 6.8 shows the results for four typical cost functions: NCC from (2.7), Sec. 2.2.2 (p. 21), Mutual Information (MI) from (2.8), Sec. 2.2.3 (p. 22)[1], as well as the truncated SAD from (2.4) with two different values of $\varepsilon_{\max} = 15$ and 40. Here $\mathcal{C}_1^\gamma$ and $\mathcal{C}_2^\gamma$ are the values of confidence terms in equations Eqs. (4.26) and (4.27) of Sec. 4.5.4, respectively, which correspond to the

[1] In the experiments to this chapter, it was important to check the consistences of best choices for smoothness parameters and quantile values of the confidence maps for *all* available cost functions; therefore Mutual Information was included into computations and, since, at the time of evaluation, computation of this cost function was only possible in the case of a rectified stereo pair, the number of images was restricted to be two.

quantile γ and the superscripts $\cdot^S$ and $\cdot^D$ denote parameters corresponding to semi-global optimization and dynamic programming respectively.

Table 6.8: Correlation between quantile values for confidence terms $\mathcal{C}_1^\gamma$ and $\mathcal{C}_2^\gamma$ and smoothness parameters λ_1 and λ_2 which yielded best results for the evaluation pipeline described above. The number of cameras was two, the size of the correlation window win $= 5$, triangulation-based constants $A = 50, \sigma = 50$, opt.r was set to 1 and the cost values for assigned (non-occluded) values of $c(\mathbf{x}, j)$ were scaled between 0 and 1, in other words, multiplication by 2048 required in the considerations of p. 65 was not carried out. Similar results were obtained also for other sequences and other parameter settings.

data set	Seq. *Tsukuba*				Seq. *Infrared*			
method	NCC	MI	SAD $\varepsilon_{\max} = 15$	SAD $\varepsilon_{\max} = 40$	NCC	MI	SAD $\varepsilon_{\max} = 15$	SAD $\varepsilon_{\max} = 40$
$\mathcal{C}_2^{0.7}$	0.36	0.15	0.38	0.30	0.40	0.22	0.50	0.41
$\mathcal{C}_2^{0.9}$	0.45	0.24	0.50	0.45	0.48	0.34	0.59	0.53
$\mathcal{C}_1^{0.7}$	0.26	0.051	0.20	0.12	0.094	0.023	0.13	0.063
$\mathcal{C}_1^{0.9}$	0.18	0.13	0.37	0.27	0.20	0.068	0.28	0.14
λ_1^S	0.18-0.68	0.13-0.29	0.37-0.98	0.27-0.78	0.20-0.78	0.068-0.29	0.28-0.98	0.14-0.78
λ_2^S/λ_1^S	2-4	1	2	4	2-4	1-2	2-4	2
λ_1^D	0.49-0.78	0.29-0.59	0.49-0.78	0.59-0.78	0.49-0.68	0.20-0.39	0.29-0.98	0.39-0.59
λ_2^D/λ_1^D	1-2	1-2	2	2	2-4	2	2-4	2

From Table 6.8, one can clearly see that the quantile values of $\mathcal{C}_1^\gamma$ and $\mathcal{C}_2^\gamma$ show similar tendencies as λ_1 for both algorithms described above. If one of quantile values becomes larger, a right-shift of the range suitable for λ_1 can also be expected. Conversely, for smaller quantile values, also smaller λ_1 can be chosen. Generally, a value around $1.5 \cdot \mathcal{C}_2^{0.9}$ and $2.5 \cdot \mathcal{C}_1^{0.9}$ is a suitable choice for λ_1 and, according to our earlier considerations, the default value for λ_2/λ_1 is 2.

Conclusion

Summarizing the content of this section, we can state that dense depth maps extraction represents a very useful module for our pipeline first because it contributes to homogenization of the point clouds (better input for Steps 3.1 and 3.2 of our reconstruction pipeline) and second because it enhances the visibility information in the texturing portion of Step 3.2. We have seen that the local methods supported by triangular meshes can reduce the number of wrong matches within triangles consistent with the surface. We can even claim that the bigger the number of points consistent with the surface is, the more similar the results of local optimization with a triangulation-based smoothness term in a binocular configuration are to those in a multi-view configuration. In the general case, multi-view configurations provide a better resolution of depth and allow treating occlusions and the regions of repetitive texture in a robust way. In order to save computing time, we prefer the *simultaneous* method supported by triangular meshes to the *median-based* method and, especially with respect of treating regions with homogeneous texture and slanted surfaces, we recommend using the semi-global global algorithms as non-local optimization method because of its clear advantages to algorithms of dynamic programming and graph-cuts.

6.4 Shape reconstruction methods – qualitative results

In this section, results for textured reconstruction from our main data sets are presented and discussed. Section 6.4.1 shows reconstruction results for the LIFT procedure; these results can be obtained if Step 3.2 of the reconstruction pipeline Alg. 1.1, p. 15 is completely omitted. Results of our main procedure for surface reconstruction by L_1 splines are presented in Sec. 6.4.2 and those of other procedures in Sec. 6.4.3.

6.4.1 Results for the LIFT-algorithm

The results for the *Local Incremental Fusion of Tessellations* algorithm, LIFT, supported by dominant-planes extraction from local tessellations (as described in Sec. 5.1.2) are presented in Figs. 6.29 and 6.30 for the video sequences *Turntable Houses* and *Infrared*, respectively. In the data set *Infrared*, points far away from the skyscraper were deleted because long skinny triangles deteriorated the visual quality of the results. Although there seem to be little sense (from the point of view of photogrammetry) to reconstruct pieces of surfaces situated several hundreds of meters from the camera locations while the length of the baseline measures only several meters, it will be, nevertheless, interesting to see in the next sections how the point-based methods are able to reconstruct this kind of surface (even when interrupted by occlusions, as in the example of the video sequence *Infrared*).

Figure 6.29: Two screen shots from the textured model of the sequence *Turntable Houses* reconstructed by the LIFT-procedure. Note the small number of undetected triangles inconsistent with the surface. Several video frames and a view of the reconstructed point cloud and the camera trajectory are given in Fig. 6.1, p. 82.

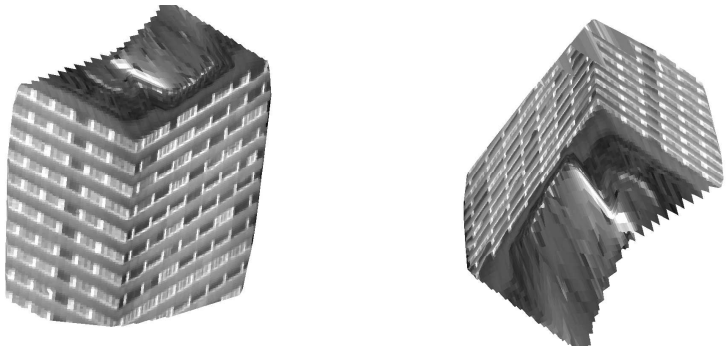

Figure 6.30: Results of reconstruction from the sequence *Infrared* with the LIFT algorithm, two screen shots from the textured model. Video frames as well as a part of the camera trajectory are given in Fig. 6.3, p. 84.

6.4.2 L_1-splines-based results

For the domain on which the nonparametric and parametric L_1-splines of Step 1 and Step 3, respectively, of the procedure described in Sec. 5.2 are calculated, we used an equally-spaced rectangular grid extending from $\min_m(X_m)$ to $\max_m(X_m)$ and from $\min_m(Y_m)$ to $\max_m(Y_m)$ in the horizontal and vertical directions, respectively. For the data sets *Turntable Houses* and *Infrared*, the number of grid cells was 30×30. We used the original point cloud depicted in Figs. 6.1 (bottom) and 6.3 (bottom) as input for the algorithm; several reference images (some of them are shown in Figs. 6.1 and 6.3, top) were used for texturing. Note the abruptly changing nature of the point cloud, with adjacent sparse and dense regions (hundreds data points describing the oblique roof near almost no points on the flat roof of Fig. 6.1) and the changes of depth (Fig. 6.3). The weights w_m were chosen equal to 1 divided by the number of points $\mathbf{X}_m$ in each triangle of the Sibson element, the smoothness parameter λ was set to be 0.7 for the nonparametric spline and the first parametric spline, and 0.8 for the second and third parametric splines (in accordance with our considerations in Sec. 5.2.3). Two views of the final mesh and three views of textured images are given in Fig. 6.31 for the data set *Turntable Houses*. A colormap view of the final mesh and a view of the result of the textured reconstruction are given in Fig. 6.32 for the data set *Infrared*. Note the topological connectivity of meshes in Fig. 6.31 in comparison to Fig. 6.29 and the ability of L_1-splines-based surfaces to obtain good reconstruction in sparsely covered areas. These areas can be observed by slightly lighter pieces that mark the texture of triangles not completely seen by any of the reference cameras behind the tower in Fig. 6.32, bottom. We also refer the reader to [24] where the process of *surface evolution* – e. g. using the nonparametric spline that results from Step 1 of the L_1-splines-based procedure – is illustrated.

6.4.3 Reconstruction results by other global methods for shape reconstruction

Comparison with the alpha-shapes procedure and iso-surface extraction

It was shown in [25] that, for non-regularized point clouds with many outliers, α-shapes are not able to provide significantly better reconstruction than the local methods of Sec. 5.1. The reconstruction results are somewhat better if the input of the algorithm is given by the regularized (for instance, RTDQT) nodes of Sec. 5.1.1. The number of holes is thereby reduced, but the problems of a noisy point cloud and an unnecessarily high number of triangles remain. One can now use commercially available software packages mentioned in [126] to perform interactively operations of mesh compression and hole filling, but an automatic approach is hardly possible here. The result of the α-shapes procedure with texturing as in Sec. 5.2.4 is visualized for data set *Gottesaue* in Fig. 6.33, top left. As mentioned in Sec. 3.2.2, the most challenging step of the algorithm based on iso-surface extraction lies in the retrieval of the normal vector field in the areas of sharp gradient change. In the middle left portion of Fig. 6.33, severe artifacts are clearly visible in the areas of the gabled roof and the towers. The visually best results of all of the methods implemented here were obtained by applying the procedure based on L_1 splines, depicted in Fig. 6.33, middle right and bottom. We can see that the L_1-splines-based surface is less affected by noise and outliers in the point cloud, as one can see in the area in front of the building; it is homeomorphic to a plane (has genus zero) and also the changes of gradient are reliably treated. Unfortunately, the problem of parameterization is not completely solved here because the surface remains a 2.5D manifold $z(x, y)$, not a parametrized 3D manifold $(x(u, v), y(u, v), z(u, v))$. We also refer the reader to [25], where, for further comparison, the qualitative results of the procedure based on conventional splines are shown and present in the next subsection a comparison in performance of two procedures in three exemplary regions of the surface.

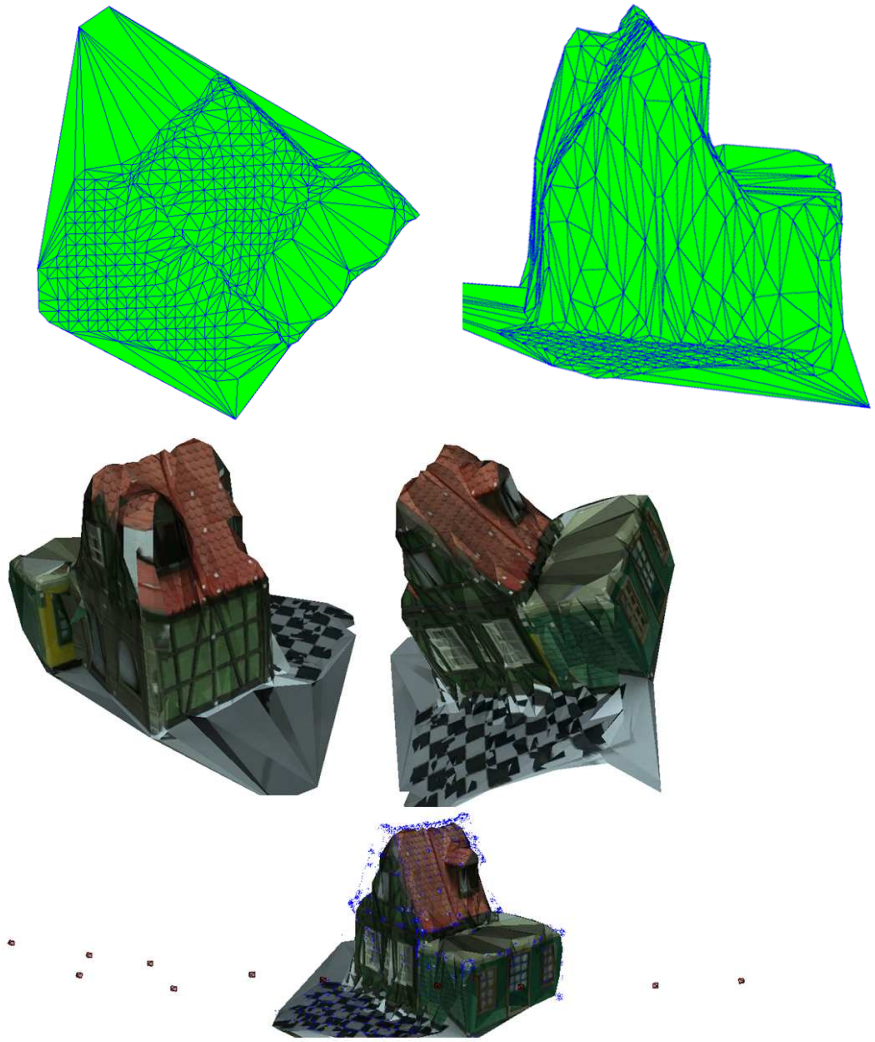

Figure 6.31: Reconstruction results of the data set *Turntable Houses* produced by the L_1-spline-based procedure of Sec. 5.2. Top: Two views of the triangular mesh. Middle and bottom: Three views of the textured reconstruction. The bottom view contains the 3D points (depicted in blue) as well as a part of camera trajectory. See also [24].

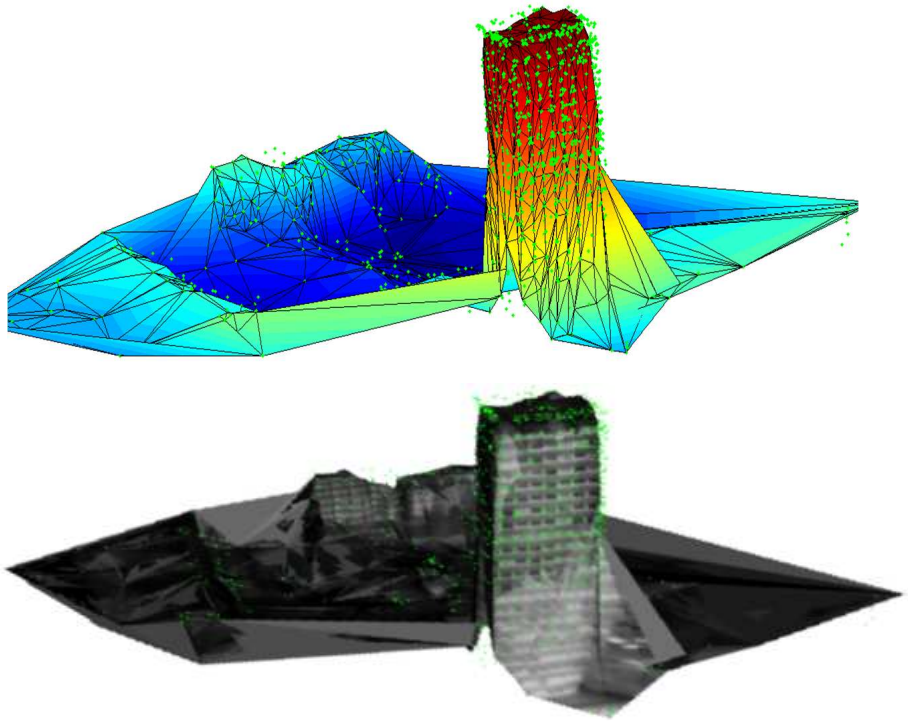

Figure 6.32: Reconstruction results of data set *Infrared* produced by the L_1-spline-based procedure of Sec. 5.2. Top: A colormap view of the triangular mesh. Bottom: A view of the textured reconstruction with the input point cloud depicted in green.

Comparison with the conventional-splines-based procedure

We are now interested in the locations and distributions of the errors in the surfaces reconstructed from the sequence *Gottesaue* by means of L_1 splines and conventional splines. Other methods are left out of consideration here since they produce topologically inconsistent meshes. Figure 6.34, left, shows a reference frame of this sequence. In this frame, we manually selected three portions of the surface corresponding roughly to the ground, wall and roof. Then the residual errors of the points near the three regression planes were computed. The three histograms depicted in the right hand side portion of Fig. 6.34 illustrate the error distribution of points to the ground plane $\pi_1 : z - z_0 = 0$ by the red histogram, of points to the wall plane $\pi_2 : x - x_0 = 0$ by the blue histogram and of points to the roof plane $\pi_3 : ax + by + cz + d = 0$ by the green histogram. We rotated the point cloud as described at the beginning of Sec. 5.2.1, oriented the ground plot of the palace to be nearly parallel to the coordinate axes and, finally, chose a translation vector and a scaling factor to put the input point cloud into the bounding box $[-4; 4] \times [-4; 4] \times [-1.5; 1.5]$. In Fig. 6.34, right, one sees that all histograms nearly correspond to Gaussian distributions, possibly contaminated by several outliers. The error distribution of the points near the ground plane is less favorable (due to the low quality of points in the textureless areas, further distance from the camera and the drift errors) than that of the points near the wall and roof. Also, since the parameters a, b, c, d of the roof plane were computed automatically, the error distribution of points on the roof is the best. We illustrate by means of the histograms of Fig. 6.35, left and right, the error distributions of surface points sampled from triangles constructed by the conventional-splines-based and the L_1-splines-based approaches, respectively. In Tab. 6.9, we report the numbers of triangles that participate in the evaluation and the measures of the error function that result from the sum of zero-mean absolute differences of the z-coordinates (in the case of π_1 and π_3) and the x-coordinates (in the case of π_2) between the

plane and the corresponding spline. For instance, in the case of π_3, this measure is

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N} |z(u,v) - \bar{u}|, u = z(u,v) + (ax + by + d)/c, \tag{6.2}$$

where N is the number of points in the triangles to be evaluated. We can see that the error of the non-parametric L_1-splines-based surface is always lower than that of the conventional-splines-based surface and that, due to the parametrization problem, the performance is worse in the area of the wall than in the ground plane and roof plane. Fully 3D parametric splines as in Sec. 5.2.2 and Sec. 5.2.3 allow reducing the error for the wall from 0.041 (which corresponds, after consideration of the real building size, to approximately 0.33m) to 0.018 (some 0.14m), but the value for the roof plane increases (for a reason that is not yet clear[2]) from 0.014 to 0.022.

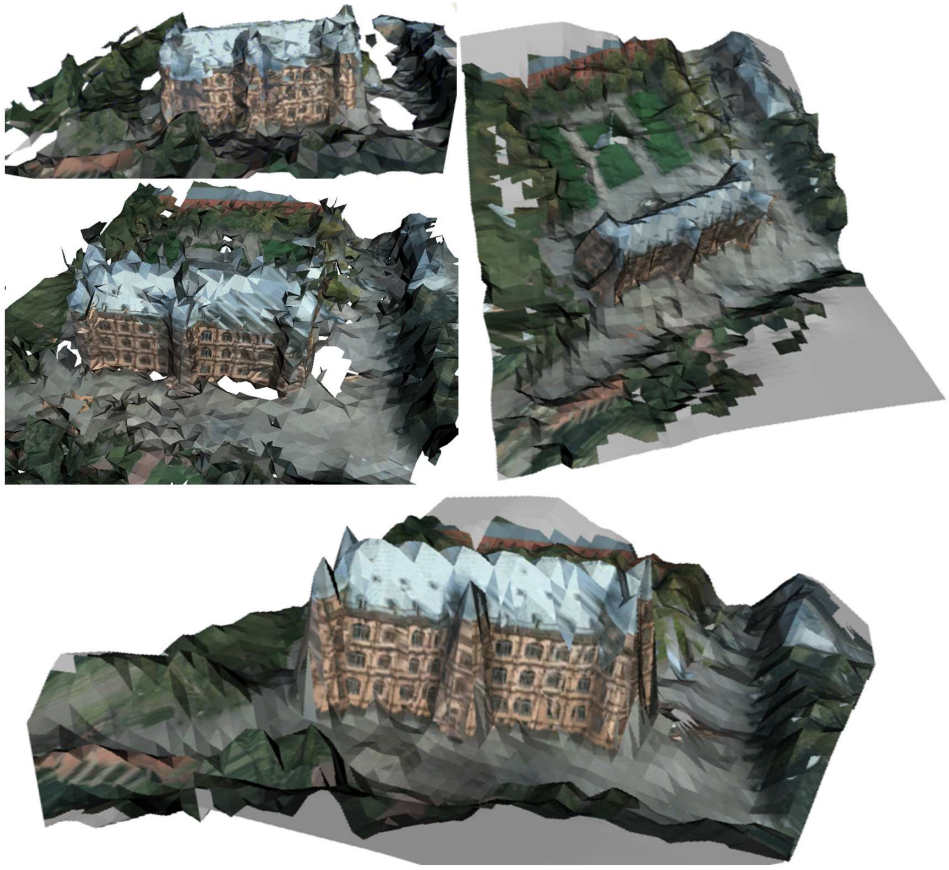

Figure 6.33: Reconstruction results from the sequence *Gottesaue*, top left: α-shapes procedure, middle left: iso-surface extraction, bottom: L_1-splines-based procedure. All three figures represent the frontal view of the building. Top right: another view of the reconstruction by the L_1-splines-based procedure.

Results for conventional cubic splines were shown here to demonstrate the susceptibility of these splines to Gibbs artifacts in areas of fast gradient change, noise and outliers.

[2]One possible interpretation is suggested by the small dormers: these are textured regions in images and therefore contain many data points. They also lie in a vertical plane and so they are inconsistent with π_3 while likely to be reconstructed by the parametric L_1-spline

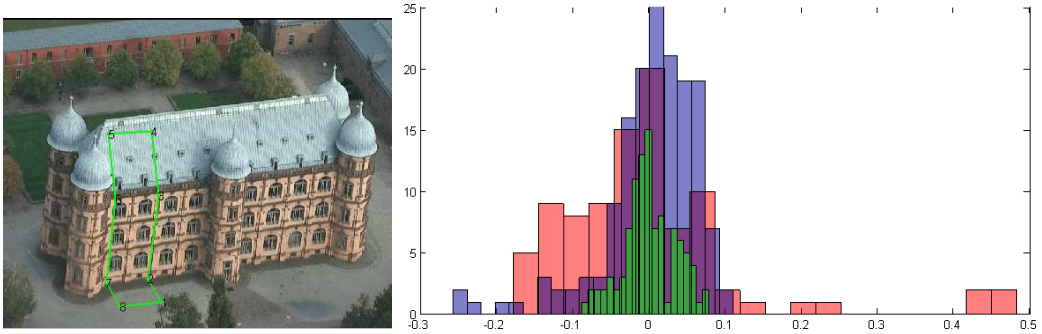

Figure 6.34: Left: A reference frame from the sequence *Gottesaue* and, marked by the green curve, the part of a surface to be evaluated. The three portions of the surface belong to the ground plane, the wall and the roof. Right: Error distributions of sample points near the ground plane (red histogram), wall (blue histogram) and roof (green histogram).

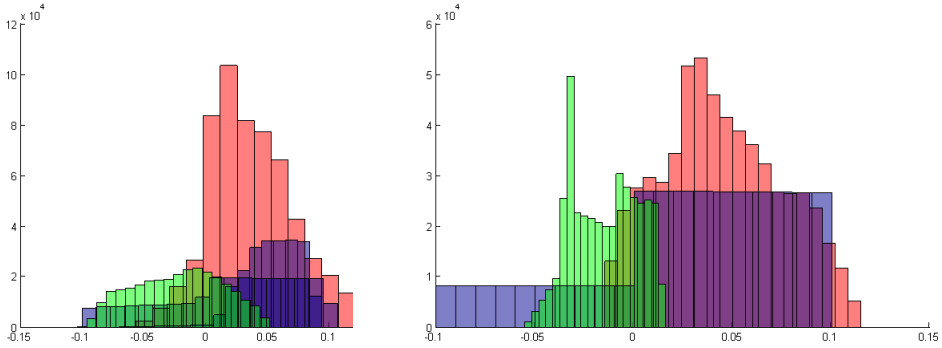

Figure 6.35: Distribution of the (non-zero-mean) deviations u from (6.2) for the procedure based on conventional splines (left) and L_1-splines (right). Red histograms stand for the reconstruction results of the ground plane, blue from the walls and green for the roof.

Table 6.9: Zero mean average deviations ε from (6.2) of the spline-based surfaces from three selected planes. Sequence *Gottesaue*, 40×40 tensor-product grid, $\lambda = 0.3$. The first number in parentheses denotes the deviations in meter while the second is the number of evaluated triangles.

	deviations ε (in m) (number of triangle)		
Method	Ground plane	Wall	Roof
L_2- splines	0.030 (0.24)(36)	0.045 (0.36)(21)	0.030 (0.24)(18)
L_1- splines (non-param)	0.025 (0.2)(21)	0.041 (0.33)(24)	0.014 (0.11)(24)
L_1- splines (param)	0.025 (0.2)(24)	0.018 (0.14)(11)	0.022 (0.18)(14)

Although these results were the only results for conventional splines presented here, the conclusions of this work and of [25] about conventional cubic splines can be generalized to other types of conventional splines mentioned in Sec. 3.2.4.

6.5 Shape reconstruction methods – quantitative evaluation

While the last section presented screen shots of the reconstruction results, the task of this section is to perform a quantitative evaluation of several procedures for shape reconstruction. The evaluation of methods for shape reconstruction is carried out in two separate sections for two main reasons. First, finding ground truth and an appropriate measure for the comparison of ground truth with triangular meshes for buildings with many complicated structures are not trivial problems. Although the comparison measure should ideally contain penalty terms for both geometry and texture, we concentrate here only on the geometry of the reconstruction and adopt the well known *Hausdorff Distance*. We motivate in Sec. 6.5.1 our choice of the Hausdorff distance as a metric for the quality of reconstruction while, in Sec. 6.5.2, we describe several technical details of the computation of this distance. While being applied on point clouds obtained from our reconstruction pipeline, any distance measure is biased not only by the quality of the input data set but also by the reconstruction result of Step 2, which makes it almost indispensable to consider a synthetic data set (not contaminated by systematic errors, such as camera drift), as we do in Sec. 6.5.3, before evaluation of a real data set can be performed in Sec. 6.5.4.

6.5.1 Hausdorff distance as a measure for completeness and correctness

A crucial issue when making comparisons is the metric (measure of similarity) in which the comparisons are made. Conventional metrics such as the average error and generalizations thereof, such as the L_p norms [40], measure similarity in ways inconsistent with human perception. For many commonplace situations, for example, thin walls in urban terrain, these metrics indicate that two sets are nearly the same when observers judge them to be dissimilar, and, conversely, for other situations, they indicate that two sets are very different while the user assesses them to be very similar.

Given a ground truth model $\mathcal{Y}$ and a reconstruction result denoted by $\mathcal{X}$, our goal is to evaluate $\mathcal{X}$ in terms of completeness (i. e. how much of $\mathcal{Y}$ is modeled by $\mathcal{X}$) and correctness (how closely $\mathcal{X}$ models $\mathcal{Y}$). These two anchors for evaluation of any algorithm were used by e. g. Heipke et. al. in [64] and, specially for geometric reconstruction, by Seitz et. al. [120]. The latter paper motivated us to use the *Hausdorff distance* as the quantitative measure to compare different procedures for geometric surface reconstruction. Other applications of the *Hausdorff metric* are to measure similarity of objects in computer vision [58] and to match objects with templates for identification in geometric modeling and tracking [109].

We denote the distance from a point $\mathbf{X}$ to mesh $\mathcal{Y}$ and the distance from mesh $\mathcal{X}$ to mesh $\mathcal{Y}$ by $\mathrm{dst}(\mathbf{X},\mathcal{Y}) = \inf_{\mathcal{Y}} d(\mathbf{X},\mathbf{Y})$ and $\mathrm{dst}(\mathcal{X},\mathcal{Y}) = \sup_{\mathcal{X}} \mathrm{dst}(\mathbf{X},\mathcal{Y})$, respectively. For our purposes, $d(\mathbf{X},\mathbf{Y})$ is the Euclidean distance between $\mathbf{X}$ and $\mathbf{Y}$ and in all definitions above, " inf " can be replaced by " min " and " sup " by " max ", because we always deal with *compact* surfaces. The Hausdorff metric for the "distance" from one set of points $\mathcal{X}$ (could be disjoint points or a continuous surface) to another set of points $\mathcal{Y}$ is

$$d_H(\mathcal{X},\mathcal{Y}) = \max\left\{\mathrm{dst}(\mathcal{X},\mathcal{Y}), \mathrm{dst}(\mathcal{Y},\mathcal{X})\right\}. \tag{6.3}$$

One can see from Fig. 6.36, left, that $\mathrm{dst}(\mathcal{X},\mathcal{Y})$ describes the correctness and $\mathrm{dst}(\mathcal{Y},\mathcal{X})$ the completeness of the reconstruction to be evaluated. The Hausdorff metric is sensitive to outliers, a property that makes it a suitable tool for evaluating surface reconstruction methods for practical applicability such as automatic navigation. Note that, in our case, the outliers to be punished are not the *input sample points* lying far from the surface but those triangles of the resulting mesh that contain points far from the surface. There also exist generalizations of the Hausdorff distance that play down the effect of outliers, for example, the generalization of [5], where an error integral over a discretized volumetric domain $\mathcal{D}$

$$d_{p,c,\mathcal{D}}(\mathcal{X},\mathcal{Y}) = \left(\int_{\mathbf{V}\in\mathcal{D}} \min\left(|\mathrm{dst}(\mathbf{V},\mathcal{X}) - \mathrm{dst}(\mathbf{V},\mathcal{Y})|^p, c \right) \right)^{1/p}$$

with two scalars $c > 0, p \geq 1$ is considered. However, in our work, the original Hausdorff distance of Eq. (6.3) (which comes out of the last equation in case $p, c \to \infty$) is adopted to perform comparisons for a simple object.

6.5.2 Details of the implementation

Care must be taken with the implementation details of the computation of the Hausdorff distance in order to prevent the algorithm from becoming quadratically expensive in terms of the sampled points, which is, of course, the worst-case scenario of (6.3). Since we work with triangular meshes ($\mathcal{Y} = (\mathbf{Y}, \mathcal{T})$), we observe that the distance $\mathrm{dst}(\mathbf{X},\mathcal{Y})$ is either a shortest distance from $\mathbf{X}$ to a vertex of the point set $\mathbf{Y}$ or the shortest length of the perpendicular from $\mathbf{X}$ to one of the faces given that the base point $\mathbf{V}$ as in (3.3) lies within a triangle T. A rather efficient way to compute $\mathrm{dst}(\mathbf{X},\mathcal{Y})$ is thus as follows.

1. compute $d_1 = \min_{\mathbf{X}} \mathrm{dst}(\mathbf{X},\mathbf{Y})$,
2. by considering normals $\mathbf{n}_{\mathcal{T}}$ (of length 1) of all triangles in the mesh, compute simultaneously (with (3.3)) both the length of the perpendiculars $d_\perp(\mathbf{X},T)$ and the base points $\mathbf{V}$,
3. as a last step, perform for every triangle T, for which $d_\perp(\mathbf{X},T)$ lies below d_1, the test $\mathbf{V} \in T$ is performed. The minimum of these values is denoted by d_2. We have $\mathrm{dst}(\mathbf{X},\mathcal{Y}) = \min(d_1, d_2)$.

The third step is the most time-consuming. It could be carried out, for example, by checking whether the sum of the barycentric coordinates $\mathcal{U},\mathcal{V},\mathcal{W}$ of $\mathbf{V} \in T$ is equal to 1. However, two heuristics can be applied to avoid this calculation. The first heuristic is a trivial one that takes into account the coming calculation of $\mathrm{dst}(\mathcal{X},\mathcal{Y})$. If we see that d_1 or d_2 is already smaller than the value $\mathrm{dst}(\mathcal{X},\mathcal{Y})$, we interrupt the calculation. The second heuristic directly concerns step 3 previously mentioned. If we assume that $\mathbf{V} \in T$, then by the Pythagorean Theorem,

$$d_\perp(\mathbf{X},T)^2 = \mathrm{dst}(\mathbf{X},\mathbf{Y})^2 - \mathrm{dst}(\mathbf{Y},\mathbf{V})^2 > \max\left(\mathrm{dst}(\mathbf{X},\mathbf{Y}_T)^2\right) - \xi(T)^2, \tag{6.4}$$

where $\mathbf{Y}$ is a vertex of T and $\xi(T)$ is the maximal Euclidean distance between a point $\mathbf{V}$ within a triangle and the vertices of the triangle:

$$\xi(T) = \max_{\mathbf{V}\in T} \left(\min_{\mathbf{Y}_T} \left(d(\mathbf{V},\mathbf{Y}_T) \right) \right)$$

It can be proven that $\xi(T)$ is either the radius of circumference (if no angle of T exceeds $\pi/2$) or the distance from the vertex opposite to its longest side to the intersection point of the perpendicular bisector of the second-longest side of T with the longest side (otherwise),

as illustrated in Fig. 6.37. The proof of this statement is trivial in the first case; in the second case, one denotes the smallest angle of T by β and the median angle by α. Then the statement follows after analysis of the two subcases $\alpha \geq 2\beta$ and $\alpha < 2\beta$ (see Fig. 6.37). The computation of the two quantities in the rightmost part of (6.4) proceeds simultaneously and is therefore very fast. Verification of the *necessary* condition (6.4) allows rejecting triangles that do not satisfy $\mathbf{V} \in T$ without computing $\mathcal{U} + \mathcal{V} + \mathcal{W}$.

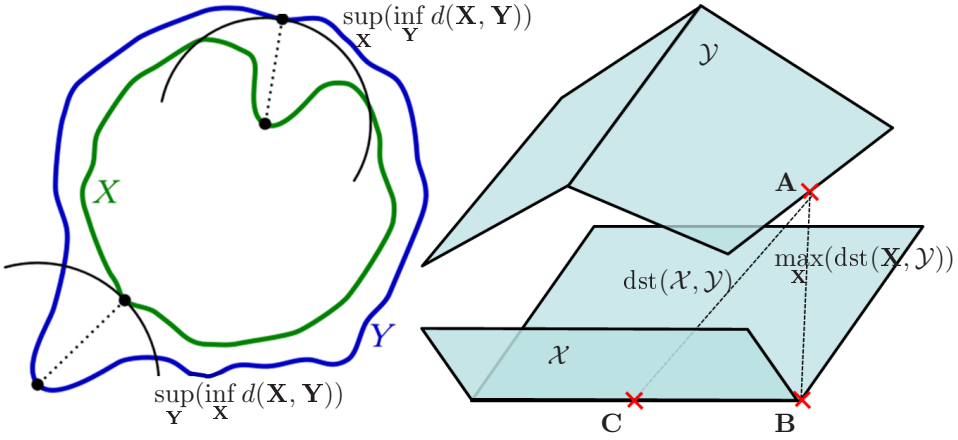

Figure 6.36: Left: The Hausdorff distance measures *completeness* and *correctness* of the reconstruction, and, as originally formulated, is sensitive to outliers (Source: Wikipedia). Right: A configuration of two points sets consisting each of two rectangles (or, equivalently, four triangles) for which both values $\mathrm{dst}(\mathcal{X}, \mathcal{Y})$ (corresponds to **AB**) and $\max_{\mathbf{X}}(\mathrm{dst}(\mathbf{X}, \mathcal{Y}))$ (that is larger or equal than **AC**) differ significantly.

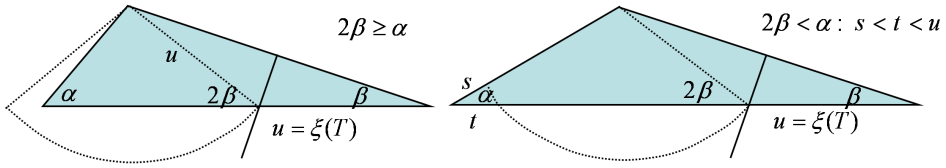

Figure 6.37: Computation of $\xi(T)$ in the case one of angles of T exceeds $\pi/2$.

One can now have an idea to evaluate $\mathrm{dst}(\mathcal{X}, \mathcal{Y})$ by evaluating *each* vertex $\mathbf{X} \in \mathcal{X}$ by the procedure described above and taking the maximum value $\max(\mathrm{dst}(\mathbf{X}, \mathcal{Y}))$. Unfortunately, even in the case of connected meshes, the extreme point is not necessarily a vertex, but can lie in the interior of an edge, as illustrated in Fig. 6.36, right, and, by a "suitable" (worst-case) choice of parameters, the difference $\mathrm{dst}(\mathcal{X}, \mathcal{Y}) - \max(\mathrm{dst}(\mathbf{X}, \mathcal{Y}))$ can be, theoretically, arbitrarily high. In the case of meshes topologically different from planes (e. g. with holes) which may be obtained from application of procedures based on α-shapes or iso-surface extraction, $\max(\mathrm{dst}(\mathbf{X}, \mathcal{Y}))$ is even a worse estimate of the one-sided distance. Therefore, for the general case, we implemented several features of the algorithm described in [56]: the points sampled from triangles in $\mathcal{X}$ and $\mathcal{Y}$ are stored in an octree array, whose finest resolution multiplied by $\sqrt[3]{2}$ is the discretization error. From the centers of the disjoint cells of the octree, the cells filled by points of the other set are identified and, if the computation to the submesh makes sense (i. e. the distance between cells is not too short), it is carried out by the methods used for computing $\mathrm{dst}(\mathbf{X}, \mathcal{Y})$. The option of fast computation of $\max_{\mathbf{X}}(\mathrm{dst}(\mathbf{X}, \mathcal{Y}))$, which, for non-pathologic cases such as that in Fig. 6.36, is a good approximation of $\mathrm{dst}(\mathcal{X}, \mathcal{Y})$, is adopted for tensor-product surfaces that produce meshes without holes.

6.5.3 Evaluation of several algorithms on a synthetic data set

The test object represented by the point cloud $\mathcal{X}$ must be simple enough that it can be correctly evaluated with the Hausdorff metric. On the other hand, it should possess all of the properties of a point cloud obtained by photogrammetric methods in urban terrain: gradient discontinuities (characteristic for man-made objects), high amplitude of Gaussian noise, several outliers and varying density of points. In [26], the point cloud $\mathcal{X}$ to be used in the comparisons represents a house with an overhanging roof (see Fig. 6.38). Computational experiments were carried out for levels 0.025 and 0.15 of Gaussian noise and for outlier percentages of 0%, 1% and 10% for x, y, and z coordinates of the point (in the case of iso-surface extraction, also for normal vectors). Here, outliers were randomly chosen points in the bounding box of the object. The density of points remained roughly unchanged in all experiments but was variable in different regions of the data set. For each level of noise and outliers, we carried out data set generation, reconstruction and evaluation 10 to 15 times and computed the average of the Hausdorff distances (6.3). Qualitative results from the L_1-splines-based procedure is shown in Figs. 6.39. As we see in the graphics that demonstrate the quantitative performance of different algorithms Fig. 6.40, our default procedure turns out to be the most robust with respect to the increasing outlier percentage. In order to reconstruct this clearly 3D point set $\mathcal{X}$ by tensor-product surfaces, we manually chose suitable spatial homographies for points on the ground, on the walls, on the roof and under the overhang that transform the points from different parts of the house into the (u, v)-plane and preserve topological relations between these points. The (u, v)-parametrization is shown in Fig. 6.38, top center. For the qualitative illustrations of other procedures, we refer to [26].

As one saw previously, the L_1-splines-based procedure shows the most stable results with respect to the percentage of outliers and noise, despite limitations due to the relatively small number of grid nodes and the rather inflexible structure of a tensor-product rectangular grid. In is also noticeable to observe the high Hausdorff-distance error of the iso-surface extraction generated by the method of [75] in the absence of noise which we believe happens because of degenerate configurations, for instance, planar structures.

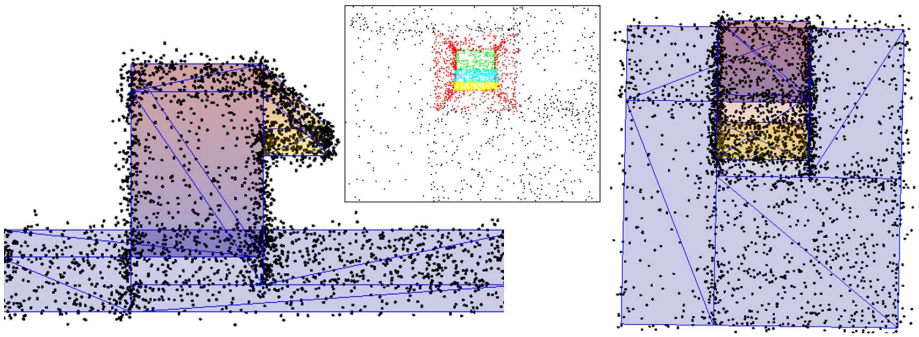

Figure 6.38: Model *Synthetic house with overhanging roof* and a point cloud without outliers, see also [26]. On the left: view from side, right: view from top, middle at top: parametrization in (u, v)-domain (points on the ground, on the walls, on the horizontal, upper and lower overhanging parts of the roof are marked in black, red, green, cyan and yellow, respectively).

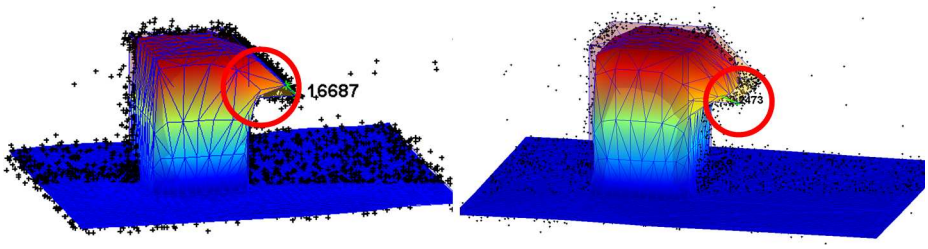

Figure 6.39: Modeling the data set *Synthetic house with overhanging roof* with L_1-splines (see also [26]). Outlier percentage is 0.01 everywhere. Equally spaced grid. Left: $\lambda = 0.3$, right: $\lambda = 0.5$. The endpoints $(\mathbf{X}, \mathbf{Y})$ producing the largest values of $d_h(\mathcal{X}, \mathcal{Y})$ are surrounded by a red circle and connected by a green line.

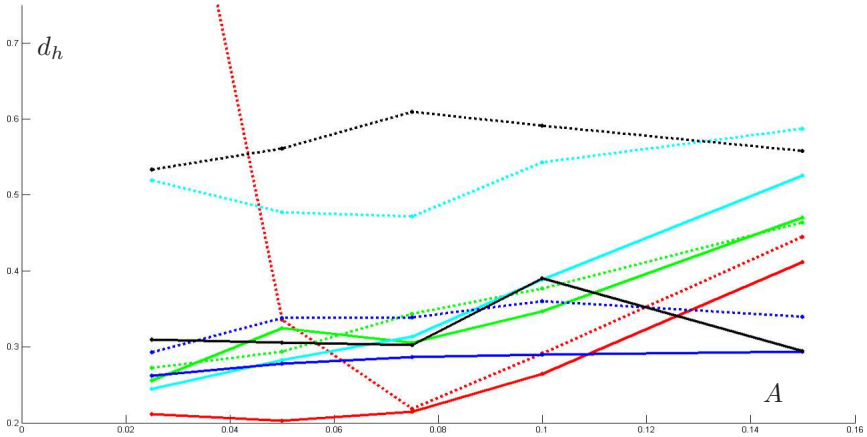

Figure 6.40: The average values of the Hausdorff distance d_h obtained for the data set *Synthetic house with overhanging roof* for Gaussian noise amplitudes (A) 0.025-0.15 by alpha-shapes (green/cyan line: a small/large value), iso-surface extraction (red), grid-fit (black) and L_1-splines (blue). Curves for data sets without outliers are shown by solid lines, for outlier percentage 0.01 by dotted lines.

6.5.4 Evaluation of a real data set

In this section, we again turn our attention to real data. Fragments of five high-resolution images of the sequence *Ettlingen church* present the entrance area of the Herz-Jesu church in Ettlingen, near Karlsruhe, Germany. The laser point set $\mathcal{Y}$, obtained from multiple scan positions by means of Zoller+Fröhlich IMAGER 5003 laser scanner and registered interactively, as a ground truth, several images and corresponding camera matrices are available [125] for evaluation of multi-view dense estimation and surface reconstruction algorithms. We selected and down-sampled five images of the sequence. Our reference image (presented in Fig. 6.41, right) is the third image of the subsequence. We mention here the two main problems that emerged during the evaluation process:

1. The laser point set contains several millions of points and is therefore not convenient for further processing (e. g., building meshes). For this reason, we did not perform meshing of the ground truth point cloud $\mathcal{Y}$, but generalized our calculations directly for the point set. For example, in order to calculate $\mathrm{dst}(\mathcal{X}, \mathcal{Y})$, the ANN algorithm due to [104] can be used. Here $\mathcal{X}, \mathcal{T}$ is again the mesh resulting from the reconstruction.

2. As one can see from Fig. 6.41, left, the laser point set $\mathcal{Y}$ is not complete (due to the unfavorable position of the scanner) and therefore cannot be considered as ideal ground truth. The error in *correctness* of our reconstruction results will be unnecessarily high if care is not taken to exclude the triangles lying in the regions where no ground truth is given. In the current implementation, we projected $\mathcal{Y}$ by the reference camera into the image and calculated the histogram that assigns the number of laser points to each triangle of the reconstruction. If we denote by $\mathcal{T}_0$ all triangles whose support set contains less than a fixed number of points in $\mathcal{Y}$, then we exclude the set of triangles

$$\mathcal{T}_1 = \{T \,|\, \text{one of vertices of } T \text{ is incident with a triangle of } \mathcal{T}_0\}$$

   from consideration. Of course, this approach will fail if some empty triangles are occluded from the reference image by regions sufficiently covered by laser scanner data, but this is not the case for our data set. By luck, also triangles near the image borders with spurious depth values at the vertices – mainly because these regions were not covered by a sufficient number of images, e. g., in the bottom left corner and on the right – belong to $\mathcal{T}_1$ and are left out of consideration.

Because of last two issues, we will treat separately the two values of the Hausdorff distance in Eq. (6.3), which, as we saw previously, denote the correctness and completeness of the reconstruction. We denote the two penalties for correctness and completeness by d_1 and d_2, respectively.

We begin with sparse reconstruction from a set of images and points tracked by the method of [94] and triangulated by means of the DLT algorithm [61]. In Fig. 6.41, left, these points are depicted in green while every 200^{th} laser point is shown in blue. We compute the Delaunay triangulation of these points, and, since the number of outliers is low and the surface we wish to describe is approximately 2.5D, the value of d_1 is low for this simple mesh. The value for d_2 is rather high because large portions of the reference image are not covered. When we computed the depth map as described in Sec. 4.5.3 (with parameters suitable for this data set (window $win = 2$, data cost function: NCC, triangulation-based parameters: $A = 50, \sigma = 100, \gamma = 0.75$, non-local optimization: semi-global algorithm) and the RTDQT-mesh (see Sec. 5.1.1) starting from the depth map, one sees in Fig. 6.42, top left and right, respectively, that the number of outliers (caused in this case by reflections in the windows) and, therefore, the value of d_1 increases. If one computes a 2.5D L_1 spline from these nodes, as described in Sec. 5.2.1, the value of d_1 becomes smaller while the value of d_2 also slightly decreases. In Fig. 6.42, in the bottom row, left and middle, meshes obtained by

the RTDQT and L_1-spline-based procedure, respectively, as well as pairs of points that are responsible for the maximum values of the correctness (d_1) and completeness (d_2) penalties are depicted. On the right of Fig. 6.42, bottom, we show two screen shots of the textured reconstruction. Quantitative results for the three procedures already mentioned here and two other tensor-product-based procedures, namely gridfid and conventional splines, are shown in Table 6.10.

Remark: The deviations of around one meter seem, clearly, very high for this simple image sequence. However the output of this section is always the *highest* deviation that can be indeed quite high. Computation of average deviations for 3D models would require modification of (6.3) that, unfortunately, is not available yet. Computation of average deviations for "2.5D models" is equivalent to comparison of depth maps and yields similar results as in Sec. 6.3.

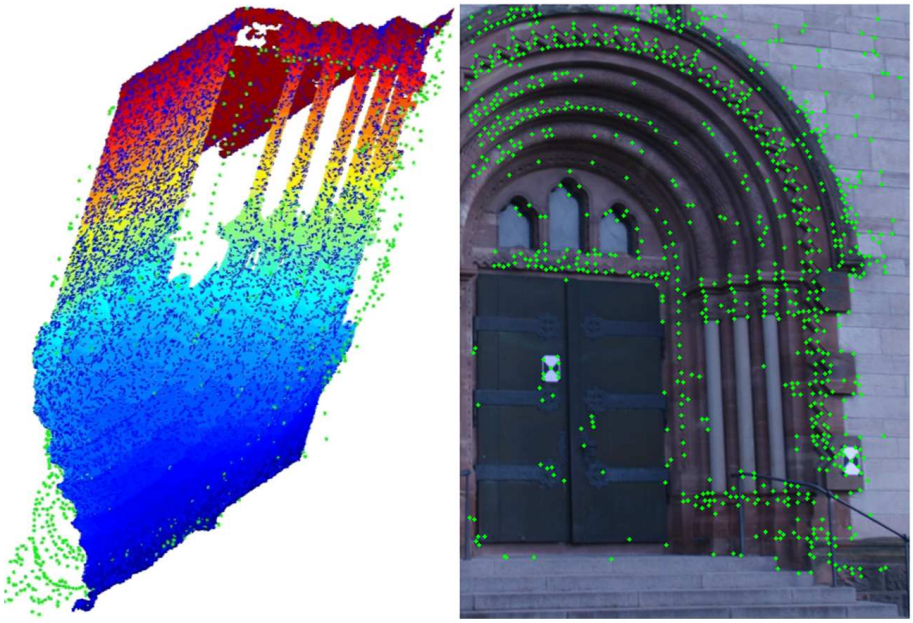

Figure 6.41: Left: The ground truth mesh with vertices given by laser points (in blue) from the image sequence *Ettlingen church*, view from behind. Note that orientation of the z-axis in the input data set is from top to bottom. The triangulated points are illustrated by green dots both in the 3D space (left) as well as in the image space of the reference image (right).

Table 6.10: Reconstruction results for the data set *Ettlingen church* produced by several methods. The grid size for all tensor-product-based methods was 50×50. The smoothness parameter λ was 0.1 for L_1 splines and conventional splines, and 0.8 for gridfit. The object bounding box measures were $[8.3; 10.8] \times [-9.5; -5.9] \times [-5.6; 0.8]$ m

.

method	Delaunay	RTDQT	L_1-splines	conv. splines	gridfid
d_1	0.216	0.754	0.186	0.718	0.298
d_2	1.00	0.610	0.486	0.478	0.598

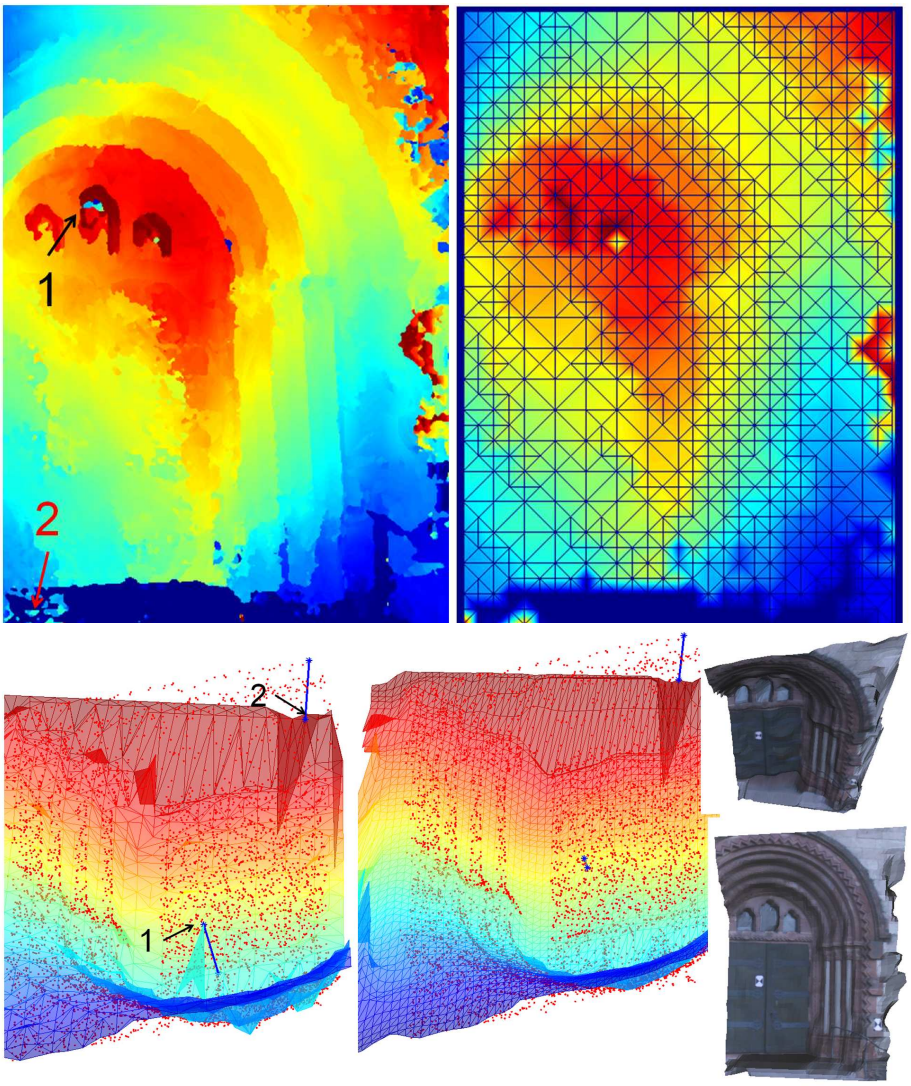

Figure 6.42: Evaluation of the data set *Ettlingen church*. Top left: depth map computed by means of the simultaneous algorithm of Sec. 4.5.3. Top right: RTDQT-mesh produced from the depth map. Bottom left and middle: Top view of the RTDQT-mesh and the mesh obtained from the L_1-splines-based procedure. The pairs of points in the ground truth and resulting meshes responsible for the highest values of the correctness (d_1) and completeness (d_2) penalty terms are depicted by blue stars and denoted, for further clarification, by 1 and 2, respectively. Bottom right: Visualization of the textured reconstruction provided by the L_1-splines-based procedure.

6.6 Computational results for the reconstruction pipeline for two more data sets

We decided to include in this work two more data sets that assist in (and are very suitable for) demonstrating the potential of our reconstruction pipeline and, in particular, that of the L_1-splines-based procedure. The village of *Wangen* in Switzerland represents a destroyed urban scenery (designated for training of police units, fire fighters and military forces) and was recorded by a quadrocopter of the type depicted in Fig. 1.1, b. It is clear that the model-based approaches are not expected to do a good job for this kind of scene. On the other hand, this scenario is exactly what the automatic navigation, disaster management, and defense missions in non-cooperative terrain are facing in a continuously increasing number of cases.

The sparse point cloud and the camera trajectory were reconstructed by means of our structure-from-motion algorithm [22]. Since the images are nearly 2.5D, it is, for qualitative illustration of the results, sufficient to compute RTDQT with filled holes from one reference frame and to model the distance of 3D points to the image plane of the reference frame using cubic splines (that is, using the 2.5D surface of Sec. 5.2.1 only and not the complete procedure). The reference image, corresponding depth map computed using median depth estimation, and several views from the point clouds triangulated by means of (4.2) and exported into an OpenGL-interface (which assigns to each 3D point its color) are depicted in Fig. 6.43. Furthermore, we illustrate in Fig. 6.44 compressed representations of the 3D point cloud produced by RTDQT-mesh (top left) and by the L_1-spline-based procedure (bottom left and right). The main observation that can be made here is that the the 2.5D L_1 spline can suppress the noise in the coordinates of the 3D points.

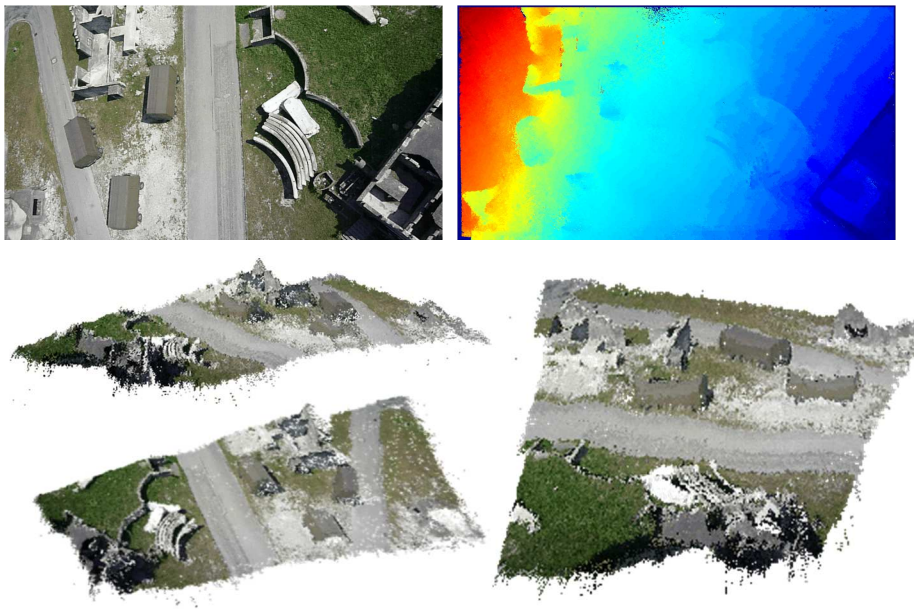

Figure 6.43: Top left: The reference image of the sequence *Wangen*. On the right, the depth map created as a median fusion of six depth maps as described in Sec. 4.5.2. Note that even by means of depth map, one can clearly see which part of the roof in the house at the bottom left still remains and which does not. (This is extremely difficult to realize when viewing the original image sequence!) Bottom: Three views of the dense point cloud (Fig. courtesy of Peter Wernerus).

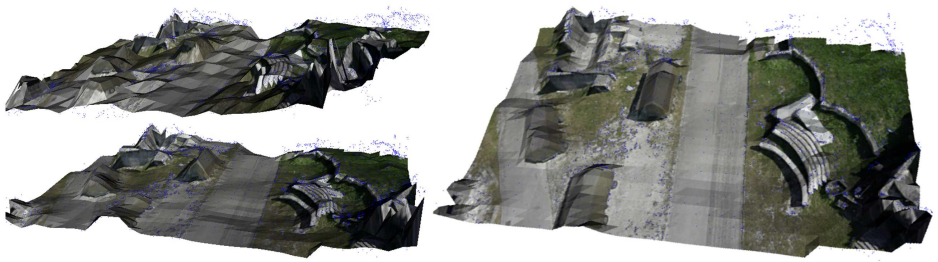

Figure 6.44: Top left: a view of the textured reconstruction from the sequence *Wangen* by the local algorithm of Sec.5.1.1 with pyramids up to level 4 and one reference image. Bottom: A similar view of the L_1-spline-based reconstruction. One can see how the 3D points not exactly computed by depth maps were replaced by spline vertices. Right: Another view of the L_1-splines-based reconstruction. The original point cloud is depicted in blue.

The next data set shows the cathedral of Speyer, a historical building in the southwest of Germany. The video sequence, from which 200 frames were automatically extracted and oriented by the procedures of [22], was recorded in late autumn by a hand-held camera mounted on a Cessna. In this case, the reconstruction is particularly difficult because of the leafless trees, which not only violate the assumption of a piecewise smooth surface needed for the image-based methods of Chapter 4 but also contribute to degeneracy of the surface, which is no longer a 2D manifold of genus zero (contrary to the assumptions of Chapter 5). Nevertheless, our methods showed their robustness and achieved reliable reconstruction in the large parts of the scene. Various steps of the reconstruction from reference frames to the views of the textured mesh are visualized in Figs. 6.45 and 6.46.

Conclusion

From the contents of Secs. 6.4-6.6, it becomes clear that the L_1-splines-based procedure is able to produce topologically consistent surfaces with reliable information even in areas not covered by the camera. Moreover, it can cope with a considerable percentage of outliers in the point clouds.

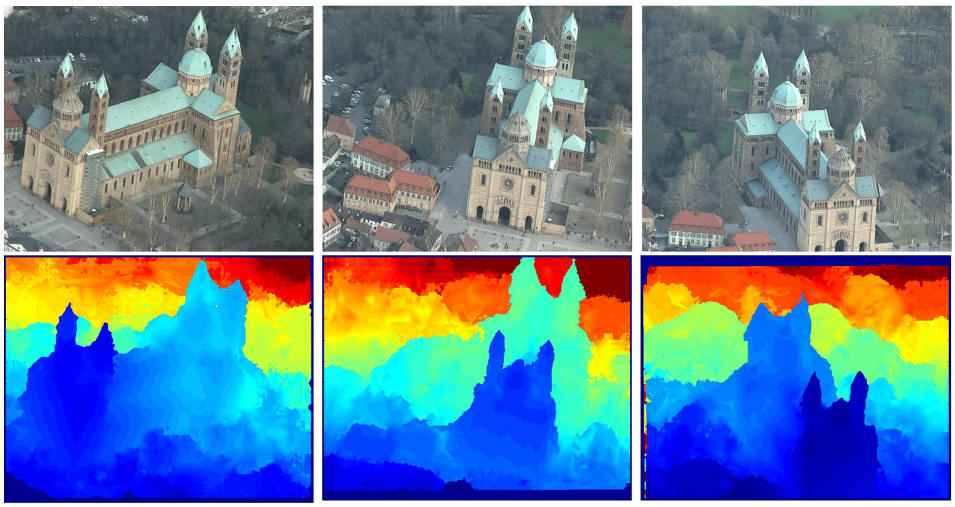

Figure 6.45: Three reference images from the sequence *Speyer* (top) and corresponding depth maps (bottom) created by the algorithm described in Sec. 4.5.3.

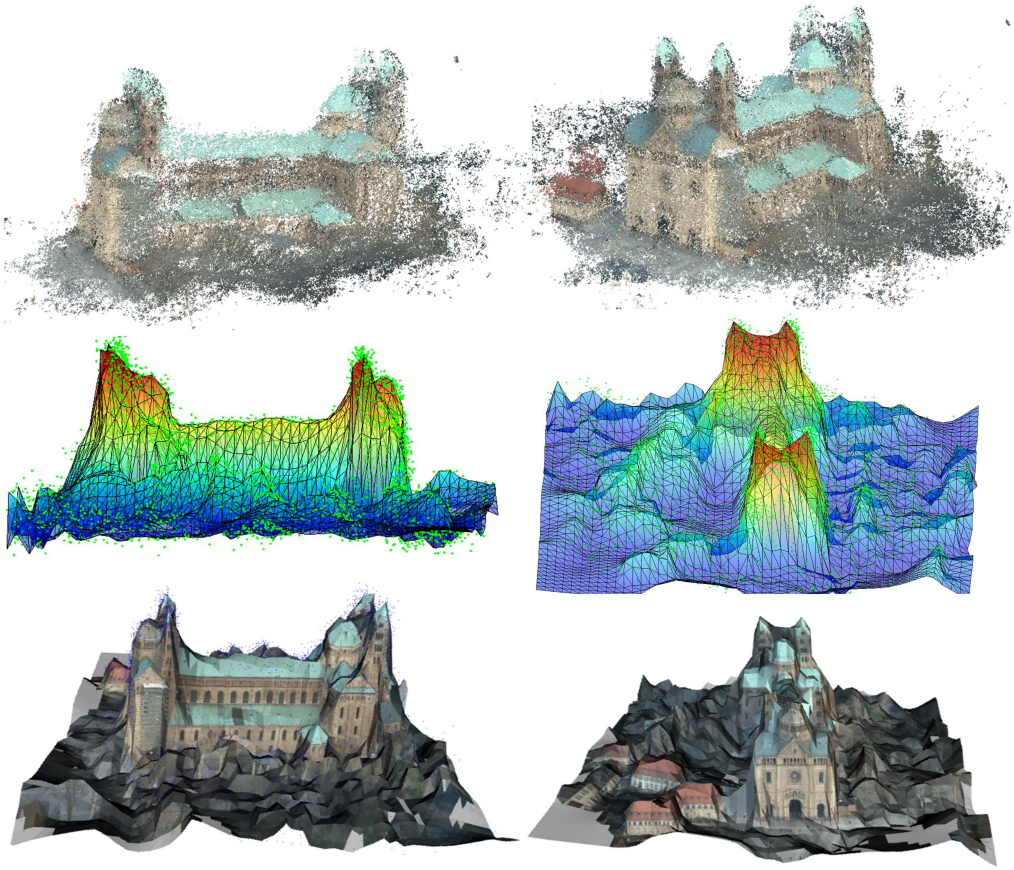

Figure 6.46: Reconstruction results for the sequence *Speyer*. Top row: Two views of the dense point cloud (Fig. courtesy of Peter Wernerus). Middle: Two views of the mesh resulting from the L_1-spline-based procedure with original point cloud depicted in green. Bottom: Two views of textured reconstruction.

6.7 Computing times

This section gives a coarse information about computing times for the main modules of the program coded on a standard laptop by the author of this work in a MATLAB GUI with several C(++)-files (mostly coded as mex-functions) for the most time-consuming procedures. Generally, there are two important properties of our algorithm that prevent the software from rapidly increasing the time for computation. The first is the subdivision in the image- and point-based steps and the second is its modular structure; the time-consuming modules of dense depth maps or L_1-splines can be omitted or replaced by the simple Delaunay triangulation or the (less time-consuming) procedure of α-shapes, respectively. The user can decide which modules should be activated.

According to the reconstruction pipeline Alg.1.1, there are four main modules: *Sparse tracking, dense reconstruction, local tessellations* and *global approach for shape reconstruction* (including texturing). In the following four paragraphs, we will report the computing times of these modules and their main subroutines. The computational "bottlenecks" of the respective modules will be described as well.

Sparse tracking includes MATLAB implementations of the epipolar and simultaneous tracking algorithms and a mex-function for the standard KLT algorithm. MATLAB files

need around 0.25 minutes for some 500 points and two pictures. The bottleneck is the choice of the relevant image fragment in (4.11), which takes place by means of bilinear interpolation and is therefore rather time-consuming in MATLAB. The mex-function of standard KLT-tracking with 5 pyramids requires less than 0.1 second for the same input data.

Dense reconstruction consists of two submodules coded by mex-functions: computation of the data term with triangulation-based smoothing and a smoothness function that is by default semi-global optimization. For 7 images with 384×288 pixels and 21 depth labels, both submodules need some 0.5 minutes. The current bottlenecks are the data exchange and the not very efficient computation of the aggregation function (4.20). Use of dynamic programming instead of semi-global optimization allows reducing the computing time by up to 3 seconds.

Local tessellations are computed directly from depths maps. Less than one minute is usually required in MATLAB in order to compute a LIFT interaction between two local tessellations (shapes). The computing time increases linearly with the number of shapes, and the whole procedure is then quadratic. Running the C-code for (optional) fitting of several dominant planes in relatively sparse point clouds requires some 1-2 seconds.

Global approach is the last step of our algorithm. The most time-consuming procedure is clearly the L_1-spline based minimization algorithm, which includes iterative solution of a linear equation system and has either $3(I+1)(J+1)$ or $9(I+1)(J+1)$ unknowns (the values of $z(x,y)$ or $\mathbf{X}(u,v)$ and their derivatives at Steps 1 and 3, respectively, of Sec. 5.2). So the computation of L_1 splines depends on the number of iterations (the inner iteration loop is needed for the primal-affine algorithm and the outer to compute the parametric spline in Sec. 5.2) and can take up to about 1 hour of time ($I = J = 40$, 1 outer iteration). Rendering of a 2.5D L_1 spline requires, however, only 1 minute. Improvements in the current (C-)code can be carried out. In addition, we mention in Sec. 7.2 several general ideas for future work that can reduce the computing time of the algorithm by orders of magnitude.

Other shape-reconstruction procedures are significantly faster. For example, the calculation of an α-shape for several thousands of 3D points requires only about 1 second. The most computationally expensive portion of this procedure is Delaunay tetrahedrization. Iso-surface extraction (implementation in C++ and MATLAB) requires 2 to 3 minutes because the normals of all points must be computed and oriented by identifying neighbors and RANSAC-based plane fitting.

The computing times for all other routines needed for our approach (detection of characteristic points, texturing, mesh manipulation, etc.) are not higher than a couple of seconds.

Chapter 7

Summary and outlook

The concept presented in this work has proved to provide good visual and quantitative reconstruction results for monocular, uncalibrated video sequences of a challenging quality from both infrared and daylight cameras. The procedure is subdivided into two major parts: image-based and point-based. This separation was retained throughout the whole process not only in order to save computation time but also in order to avoid getting stuck in a local minimum of some global minimization functional. We showed in the image-based portion how to obtain depth maps from short subsequences of images. In the point-based portion, also called shape reconstruction, these depth maps are integrated into a global triangular mesh and textured by the images.

The algorithm is nearly autonomous. The only user intervention may consist of selecting the method of surface reconstruction and specifying thresholds $\min_m(x_m)$, $\max_m(x_m)$, $\min_m(y_m)$ and $\max_m(y_m)$ to reconstruct the fragment of interest. The reconstruction pipeline is real-time oriented and only the last step – surface reconstruction – must wait until the whole point cloud is obtained. We start the detailed discussion of our conclusions in Sec. 7.1 by emphasizing the main features of the image-based methods. The methods for shape reconstruction are summarized in Sec. 7.2. For every contribution mentioned in this work, we discuss not only the main advantages and drawbacks, but also ideas recommended for future work which suppose improvements over the existing drawbacks.

7.1 Image-based methods

Using the algorithms presented in Chapter 4, we are able to compute correspondences for a sparse or dense point sets from several images, optionally pairwise rectified to epipolar geometry, using modular cost functions, with or without triangular meshes, with or without subpixel precision and with or without non-local refinement (for dense methods) by means of dynamic programming, semi-global optimization, or, in the binocular case, graph-based approach of alpha-expansions.

Sparse tracking and triangulation

We have seen from Tables 6.2-6.6 of Sec. 6.2 that consideration of multi-camera systems is a powerful tool in order to obtain both exact spatial coordinates from characteristic points in images and dense depth maps without too many additional heuristics. The precision of the results obtained by epipolar and simultaneous tracking policies is, in theory, approximately the same since, in the end, all cameras participate in the reconstruction. But in practice, simultaneous tracking suffers more from uncertainties in camera positions, from the not

always correct assumption of almost fronto-parallel object planes (which requires including orientation (the normal vector) of π from **Result 1** into the optimization pipeline and, in particular, Eq. (4.11), as it was done in [54] for sparse tracking, [18] for local methods (see Sec. 3.1.2) as well as [76] for global surface reconstruction methods followed by the level set procedure of Sec. 3.2.3) and from radiometric artifacts in the reference image. While the last problem can be solved by varying interacting pairs of images, both of the other problems can hardly be solved without introducing additional parameters and statistical tests as in [54]. Considering camera uncertainties as described in Sec. 6.2 would probably improve the situation because the error bounds for camera matrices are usually known from Step 1 of Alg. 1.1.

Depth map extraction

A new idea of applying triangulation-based smoothing was presented in the course of this work. It consisted of a smoothness term and an additional evaluation step that ascertains whether a triangle is consistent or inconsistent with the surface. This helps overcome the biases of the non-local methods toward fronto-parallel surfaces. Since triangulation-based terms are also a kind of smoothing, they usually seem – at first glance – not to bring very significant improvement of the graphics of Figs. 6.26-6.28 if they are followed by non-local methods with suitably chosen parameters, but these graphics do not reflect the fact that the depth values of points within triangles consistent with the surface are obtained with subpixel precision. An isolated outlier within the point set usually does not affect the performance of the algorithm because triangles incident with it are supposed to be filtered out as inconsistent with the surface. By considering further reference frames, as described in Sec. 5.1.2, it is also possible to correct gross errors for triangles spuriously added to the list of surface-consistent triangles. Other advantages of the triangulation-based approach – its ability to initialize depth maps, disentanglement from discretization heuristics, the perspective of optimization with global methods *only* in areas made up of triangles that are inconsistent with the surface – make us believe that the approach can still be improved. One can, for example, consider for equations (4.21) and (4.22) a term $A(T)$ instead of A, where $A(T)$ decreases as the variance of the depth at triangle vertices increases, and $\sigma(T)$ instead of σ, where $\sigma(T)$ is larger for triangles with homogeneous color distribution in order to improve the classification of triangles into consistent and inconsistent with the surface. Within one subsequence, our future work will also consist of pushing forward the histogram approach described in [29] for finding similar triangles and recalculating cost functions for triangles with flipped depth values. This approach must first be generalized for multi-camera configurations.

As for non-local methods, numerous tests were carried out with dynamic programming, semi-global optimization, and, in the binocular case, with the graph-cuts-based approach. Semi-global optimization with 16 optimization paths obtained clearly better results than dynamic programming (due to streaking artifacts) and the graph-cuts-based approach (due to its susceptibility to fronto-parallel planes) while the computing time turned out to be a clear advantage of dynamic programming. Overall, the implementation of the image-based part of our reconstruction pipeline is very favorable for future developments. New cost functions as well as other aggregation functions and non-local algorithms can easily be added as additional modules. Because of the efficient, abstract problem statement for dynamic programming and semi-global matching, other smoothness functions can also be integrated into the software if necessary. However, for multi-view dense reconstruction of our data sets, the smoothness term (4.23) contributed to better results than other terms mentioned in Sec. 2.3.

In the current version of the software, automatic choice of reference frames and other images of the subsequence is insufficiently covered. Motion blur and many other artifacts can

make the reference frame unsuitable for dense reconstruction. Other images can have parallaxes to the reference frame that are either too large (which leads to many disparity/depth levels and therefore high computing time) or too small, which has the consequence that the numerical stability for retrieving 3D structure is lost. Adopting some of the heuristics mentioned in [50] will help to overcome these drawbacks.

7.2 Shape reconstruction and visualization

Local methods for shape reconstruction

We start this section by summarizing our local method, the LIFT algorithm introduced in Sec. 5.1. This is a close-to-real-time incremental method for filtering triangles that not only does not require solving texturing problem (as in global methods, see Sec. 5.2.4) but also allows covering the object surface with multi-sensorial texture. An example of triangulation-based multi-sensorial surface representation is presented in [27], where the author works with disparities and **Result 2**, for which the 3D structure does not need to be explicitly computed. A textured 3D model representation from additional sources (e. g. combination infrared and daylight videos) is also possible. The simple concept of the LIFT algorithm allows improving the quality of the mesh by additional sources, such as dominant planes. The main conceptual drawback of the current implementation is that the algorithm is biased toward the old reconstruction: if a new triangle blocks an old one, it is deleted, although it is theoretically possible that the positions of the vertices of the old triangle are less accurate. The parameter ε in Alg. 8.4 is thus a user-specified threshold and the results are very sensitive to its choice. In order to solve these problems, it will be necessary to take the accuracy of the 3D points into account and to consider the global structure of the scenery, for instance, by maintaining and updating, after processing every reference frame, an octree structure.

Global methods for shape reconstruction

Among many procedures tested in the course of this work, the L_1-splines-based procedure performs the most robust reconstruction of the urban terrain despite highly varying density of points, high amplitude of Gaussian noise and outliers. The fact that the L_1-norm is coupled to the coordinate axis and is not affine invariant against rotations and affine transformations does not significantly affect the computational results. Making use of additional information, such as known footprints of buildings that might be obtained from photogrammetric or architectural databases, or developing approaches for removing outliers, would improve the performance of all procedures, including that of the L_1-spline-based procedure. Still, by not using the bells and whistles, one gets clear insight into the fundamental capabilities of the proposed method unaffected by other factors. The present work treats the case when the footprints of buildings and other model-based information (except the direction of the z-axis, described in Sec. 5.2.1) are not a priori known.

There were several limitations in the current implementation of the L_1-splines-based procedure, namely,

1. use of a static, coarse, equally spaced rectangular grid that does not adapt to the local density and characteristics of the point cloud,
2. non-adaptive balance parameters in functionals (5.1) and (5.3),
3. high computing time due to global calculation of the L_1 splines and
4. use of the parameterization of points described in Sec. 5.2.2 that is very sensitive to the quality of the initial triangulation and the correct choice of the z-axis.

The results that we have presented in this work prove the principle of comparability or superiority of our method in comparison with other procedures but, because of the limitations mentioned above, the procedure for this method is not yet fully flexible and not yet computing-time-optimized. By making further improvements in the implementation of the L_1-spline-based procedure, we expect to achieve further improved textured reconstructions. Specifically, in the future, we will investigate extending the procedure of Sec. 5.2 using

1. flexible triangular grids that adapt to the local density and characteristics of the point cloud. Possible directions of research on triangular grids include but are not limited to C^0 linear splines (for comparison with gridfit) and C^1 cubic L_1 splines. These splines consist of Clough-Tocher elements (separate cubic polynomials in three subtriangles of a mesh triangle) [73] and are analogous to C^1 cubic L_1 splines on rectangular grids, which consist of Sibson elements. The triangulation to be chosen will be data-dependent, with roughly the same number of data points assigned to each triangle in the parametric (u, v)-domain, and it will preserve topological relations.

2. locally adaptive balance parameters λ in functionals (5.1) and (5.3) (that will not over-smooth the edges describing the walls of buildings). Alternatively, since an automatic choice of λ is in general a non-trivial issue, use of L_1 spline fits [84], which do not involve any balance parameter, can be considered.

3. reduction of computing time by 1-4 orders of magnitude by local processing of the point cloud using domain decomposition, that is, by computing local models on overlapping local domains and assembling the local models to generate the global model (see [88]). This is feasible without detriment to accuracy because L_1 splines keep local perturbations in the data completely (not just mostly) local in the surface.

The parameterization of points is indeed a rather complicated issue for future work. From Fig. 5.3, left, one can see that the building walls will not become completely vertical even after a large number of iterations and that the approach can fail if the angle between the z-axis and the correct vertical direction is too large. (It could be asserted that an angle of 15 degrees is already critical for a data set similar to the synthetic one described in Sec. 6.5.3, but, in this case, the problem can be alleviated by rescaling the point cloud). We will search for a solution both by manipulating the point cloud by means of the approaches mentioned in Sec. 5.2.2 and by modifying approaches that are not based on systems of coordinates (such as level sets with consideration of image information) by our L_1-splines-based tools.

Two possibilities for meshing the surface after its generation were mentioned in Sec. 5.2.4: Delaunay-triangulation of multi-points and canonic triangulation of the spline nodes. Here, our future work will consist of further effort to manipulate the mesh with the goal of compressing the mesh without deteriorating its quality.

Due to the strict separation of image- and point-based methods in our reconstruction pipeline as well as the quite simple texturing step described in Sec. 5.2.4, our textured models have several disadvantages, such as differences in the luminance of neighboring triangles that have been textured from different images, occasional errors caused by choosing a wrong camera (if the visibility relations are not exact) and, finally, the fact that the cameras are not error-free and so the choice of image coordinates is not always exact. Improving the texturing portion of the reconstruction procedure can proceed by a combination of following ideas that will be part of our future work.

1. modification of the cost function and applying non-local labeling algorithms on triangular grids in the same way that the algorithm mentioned in Sec. 3.1 and Sec. 4.5.3 works on rectangular grids.

2. smoothing, as described e. g. in [45], the color distribution of the triangles by using linear combinations $I(T) = \sum_k t_k I_k(T)$ where $I(T)$ denotes intensity or color values of the triangle T in 3D space, the $I_k(T)$ denote intensity or color values of triangles in the images I_k in which T is visible, and the t_k are transparency values that satisfy the constraint $\sum t_k = 1$ and depend on the angles that the triangle normal builds with the camera rays toward the center of gravity of T. Of course, the problems of a rapidly increasing number of triangles as well as uncertainties in the positions of cameras must be taken into account.

3. simultaneous consideration of image- and object-based modeling as mentioned in the end of the previous paragraph.

Evaluation of algorithms for shape reconstruction

Our next group of observations concerns performance evaluation of shape reconstruction algorithms by means of the Hausdorff distance as described in Sec. 6.5. Experiments described in this section as well as in [26] make clear the correlation between lower Hausdorff distance and better reconstruction in the view of the user interested in practical applications. Three important directions of future work are

1. modifying the error function to make it less outlier-sensitive,

2. applying modifications of Eq. (6.3) that allow considering not only geometry, but also texture deviations of the reconstructed models and

3. comparing the procedures investigated in this work with a wider class of reconstruction procedures.

Conclusion

Despite several still existing problems – efforts to cope with them are currently being made – it is clear that the reconstruction procedure presented in this work can be used for obtaining excellent textured 3D models for buildings and surrounding terrain from monocular aerial and UAV-videos.

Bibliography

[1] *Adamson, A. and Alexa, M.:* Approximating bounded, non-orientable surfaces from points. In: Shape Modeling International, pp. 243–252, (2004)

[2] *Adler, I. and Monteiro, R.D.C.:* Limiting Behavior of the Affine Scaling Continuous Trajectories for Linear Programming. In: Mathematical Programming, 50, pp. 29–51, (1991)

[3] *Akkouche, S. and Galin, E.:* Adaptive Implicit Surface Polygonization Using Marching Triangles. In: Computer Graphic Forum, 20(2), pp. 67–80, (2001)

[4] *Amenta, N. and Bern, M.:* Surface Reconstruction by Voronoi Filtering. In: Discrete and Computational Geometry, 22, pp. 481–504, (1999)

[5] *Baddeley, A.J.:* An Error Metric for Binary Images. In: Förstner, W. and Ruwiedel, S. (eds), Robust Computer Vision, Karlsruhe, Germany, pp. 59–78, (1992)

[6] *Bajaj C., Bernardini, F., and Xu, G.:* Adaptive Reconstruction of Surfaces and Scalar Fields from Scattered Trivariate data. In: Computer Science Technical Report CSD-TR-95-028, Purdue University, (1995)

[7] *Baker, S., Szeliski, R., and Anandan, P.:* A Layered Approach to Stereo Reconstruction. In: Int. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 434–441, (1998)

[8] *Bay, H., Tuytelaars, T., and Van Gool, L.:* SURF: Speeded Up Robust Features. In: Proceedings of the 9th European Conference on Computer Vision (ECCV), (1), pp. 404–417, (2006)

[9] *Beder, C. and Steffen, R.:* Incremental Estimation Without Specifying A-priori Covariance Matrices for the Novel Parameters. In: VLMP Workshop on CVPR, Anchorage, USA, (2008)

[10] *Belhumeur, P.:* A Bayesian Approach to Binocular Stereopsis. In: International Journal of Computer Vision (IJCV), 19(3), pp. 237–260, (1996)

[11] *Bernardini, F. and Bajaj, C.:* Sampling and Reconstructing Manifolds using Alpha-Shapes. In: Proc. of the Ninth Canadian Conference on Computational Geometry, pp. 193–198, (1997)

[12] *Bernardini, F., Mittleman. J., Rushmeir, H., and Silva, C.:* The Ball-Pivoting Algorithm for Surface Reconstruction. In: IEEE Trans. Visual. Comput. Graphics , 5(8), pp. 349–359, (1999)

[13] *Birchfield, S. and Tomasi, C.:* A Pixel Dissimilarity Measure that is Insensitive to Image Sampling. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 20(4), pp. 401–406, (1998)

[14] *Bleyer, M. and Gelautz, M.:* A Layered Stereo Matching Algorithm using Image Segmentation and Global Visibility Constraints. In: ISPRS Journal of Photogrammetry and Remote Sensing, 59(3), pp. 128–150, (2005)

[15] *Bleyer, M. and Gelautz, M.:* Simple but Effective Tree Structures for Dynamic Programming-based Stereo Matching. In: Int. Conf. Computer Vision Theory and Applications (VISAPP), (2), pp. 415–422, (2008)

[16] *Boissonnat, J. D.:* Geometric Structures for Three-Dimensional Shape Representation. In: ACM Transactions on Graphics, 3(4), pp. 266–286, (1984)

[17] *Boissonnat, J. D., Cohen-Steiner, D., Mourrain, B., Rote, G., and Vegter, G.:* Meshing of Surfaces. In: Boissonnat, J. D. and Teillaud, M., (eds), Effective Computational Geometry for Curves and Surfaces, Springer-Verlag, Mathematics and Visualization, pp. 181–229, (2006)

[18] *Boykov, Y., Veksler, O., and Zabih, R.:* A Variable Window Approach to Early Vision. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 20(12), pp. 1283–1294, (1998)

[19] *Boykov, Y., Veksler, O., and Zabih, R.:* Fast Approximate Energy Minimization via Graph Cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 23(11), pp. 1222–1239, (2001)

[20] *Brovelli, M. A. and Cannata, M.:* Digital Terrain Model Reconstruction in Urban Areas from Airborne Laser Scanning Data: the Method and an Example for Pavia (Northern Italy). In: Computers & Geosciences, 30, pp. 325–331, (2004)

[21] *Brox, T., Bruhn, A., Papenberg, N., and Weickert, J.:* High Accuracy Optical Flow Estimation Based on a Theory for Warping. In: Pajdla, T. and Matas, J. (eds), Computer Vision (ECCV), Lecture Notes in Computer Science, 3024, Springer, Berlin, pp. 25–36, (2004)

[22] *Bulatov, D.:* Towards Euclidean Reconstruction from Video Sequences, In: Int. Conf. Computer Vision Theory and Applications (VISAPP), (2) pp. 476–483, (2008)

[23] *Bulatov, D.:* Metric Reconstruction from Video Sequences. Open German Russian Workshop (OGRW) in Ettlingen, Germany (2007) and in Pattern Recognition and Image Analysis (PRIA), pp. 300–308, (2008)

[24] *Bulatov, D. and Lavery, J.:* Reconstruction and texturing of 3D urban terrain from uncalibrated monocular images using L_1 Splines. Photogrammetric Engineering and Remote Sensing, (PE & RS), 75(10), pp. 439–450, (2010)

[25] *Bulatov, D. and Lavery, J.:* Comparison of Reconstruction and Texturing of 3D Urban Terrain by L_1 Splines, Conventional Splines and Alpha Shapes. In: Int. Conf. Computer Vision Theory and Applications (VISAPP) (2), pp. 403–409, (2009)

[26] *Bulatov, D. and Lavery, J.:* Comparison in the Hausdorff-Metric of Reconstruction of 3D Urban Terrain by Four Procedures. In: Int. Conf. Computer Vision Theory and Applications (VISAPP-GRAPP), pp. 125–129, (2010)

[27] *Bulatov, D.:* Triangulation-based Video Registration and Applications. In: International Archives of Photogrammetry and Remote Sensing, Proc. of the High-Resolution Earth Imaging for Geospatial Information, ISPRS-workshop in Hanover, Germany, (2009)

[28] *Bulatov, D., Wernerus, P., and Heipke, C.:* Multi-view Dense Matching Supported by Triangular Meshes, to appear in ISPRS Journal of Photogrammetry, (2011)

[29] *Bulatov, D., Wernerus, P., and Lang, S.:* A New Triangulation-Based Method for Disparity Estimation in Image Sequences, Scandinavian Conference on Image Analysis (SCIA), Oslo, Norway, pp. 279–290, (2009)

[30] *Canny, J. A.:* Computational Approach to Edge Detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 8(6), pp. 679–698, (1986)

[31] *Cha, S-H. and Srihari, S. N.:* On Measuring the Distance Between Histograms. In: Pattern Recognition, 35(6), pp. 1355–1370, (2002)

[32] *Chen, S.E. and Williams, L.:* View Interpolation for Image Synthesis. In: Proc. ACM SIGGRAPH, 26(2), pp. 279–288, (1993)

[33] *Comninos, P.:* Mathematical and Computer Programming Techniques for Computer Graphics. Springer-Verlag London Ltd (2006)

[34] *Cornelis, N. Cornelis, K., and Van Gool, L.:* Fast Compact City Modeling for Navigation Pre-visualization. In: Int. Conf. Computer Vision and Pattern Recognition (CVPR)(2), pp. 1339–1344, (2006)

[35] *Cox, T.F. and Cox, M.A.A.:* Multidimensional Scaling, Chapman and Hall, USA, (2001)

[36] *Curless, B. and Levoy, M.:* A Volumetric Method for Building Complex Models from Range Images. In: Proc. ACM SIGGRAPH, 30, pp. 303–312, (1996)

[37] *Deans, S.:* The Radon Transform and Some of its Applications. Wiley, New York, (1983)

[38] *D'Errico, J.:* Surface Fitting using Gridfit. www.mathworks.com/matlabcentral/fileexchange/8998, (2006)

[39] *Desbrun, M., Tsingos, N., and Gascuel, M. P.:* Adaptive Sampling of Implicit Surfaces for Interactive Modeling and Animation. In: Computer Graphics Forum, 15(5), pp. 319–325, (1996)

[40] *DiBenedetto, E.:* Real Analysis. Birkhäuser, Boston, (2002)

[41] *Dyn, N., Levin, D., and Rippa, S.:* Data Dependent Triangulations for Piecewise Linear Interpolation. In: IMA Journal of Numerical Analysis, 10(1), pp. 137–154, (1990)

[42] *Eck, M. and Hoppe, H.:* Automatic Reconstruction of B-spline Surfaces of Arbitrary Topological Type. In: Proc. 23rd Annual Conf. Computer Graphics and Interactive Techniques, pp. 325–334, (1996)

[43] *Edelsbrunner, H. and Mücke, E.P.:* Three-dimensional Alpha Shapes. In: ACM Transactions, on Graphics, 13(1), pp. 43–72, (1994)

[44] *Edelsbrunner, H.:* Weighted Alpha Shapes. Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, Illinois, (1992)

[45] *El Hakim, S., Brenner, C., and Roth, G:* A Multi-sensor Approach to Creating Accurate Virtual Environments, 53, pp. 379-391, (1998)

[46] *Förstner, W. and Gülch, E.:* A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features. In: International Archives of Photogrammetry and Remote Sensing, Proc. of the ISPRS Intercommission Workshop on Fast Processing of Photogrammetric Data, ISPRS Journal of Photogrammetry, pp. 281–305, (1987)

[47] *Furukawa, Y. and Ponce, J.:* High-fidelity Image-based Modeling. Technical Report 2006-02, UIUC, (2006)

[48] *Furukawa, Y. and Ponce, J.:* Accurate, Dense, and Robust Multi-View Stereopsis. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 1–8, (2007)

[49] *Gill, P.E. and Murray, W.:* Algorithms for the Solution of the Nonlinear Least-squares Problem. In: SIAM J. Numer. Anal., 15, pp. 977–992, (1978)

[50] *Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S.:* Multi-View Stereo for Community Photo Collections. In: Proc. IEEE Int. Conf. on Computer Vision (ICCV), pp. 1–8, (2007)

[51] *Gold, S. and Rangarajan, A.:* A Graduated Assignment Algorithm for Graph Matching. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 18(4), pp. 377–388, (1996)

[52] *Gopi, M., Krishnan, S., Silva C.:* Surface Reconstruction Based on Lower Dimensional Localized Delaunay Triangulation. In: Computer Graphics Forum 19(3), pp. 467–478, (2000)

[53] *Gress, A. and Klein, R.:* Efficient Representation and Extraction of 2-Manifold Iso-surfaces using kd-trees. In: Proc. of the IEEE Conf. of Graphical Models, 66(6), pp. 370–397 (2004)

[54] *Gruen, A.:* Adaptive Least Squares Correlation: A Powerful Image Matching Technique. In: South African Journal of Photogrammetry, Remote Sensing and Cartography, 14(3), pp. 175-187, (1985)

[55] *Gruen, A. and Baltsavias, E.:* Geometrically Constrained Multi-photo Matching. In: Proc. of Intercommission Conference on "Fast Photogrammetric Processing", Interlaken, Switzerland, (1987) and: Photogrammetric Engineering and Remote Sensing (PE & RS), 54(5), pp. 633–641 (1988)

[56] *Guthe, M., Borodin, P., and Klein, R.:* Fast and Accurate Hausdorff Distance Calculation Between Meshes. In: J. WSCG, pp. 41–48, (2005)

[57] *Han, L., and Schumaker, L.:* Fitting Monotone Surfaces to Scattered Data using C^1 Piecewise Cubics. In: SIAM J. Numer. Anal., 34, pp. 569–585, (1997)

[58] *Han, I., Yun, I.D., and Lee, S.U.:* Modified Hausdorff Distance for Model-based 3-D Object Recognition from a Single View. In: J. Visual Comm. Image Repres. (15), pp. 27–43, (2004)

[59] *Hansen, P.C. and O'Leary, D.:* The Use of the L-curve in the Regularization of Discrete Ill-posed Problems. In: SIAM J. Sci. Comput. 14(6), pp. 1487–1503, (1993)

[60] *Harris, C. G. and Stevens, M. J.:* A Combined Corner and Edge Detector. In: Proc. of fourth Alvey Vision Conference, pp. 147-151, (1998)

[61] *Hartley, R. and Zisserman, A.:* Multiple View Geometry in Computer Vision, Cambridge University Press, (2000)

[62] *Heinrichs, M., Hellwich, O., and Rodehorst, V.:* Efficient Semi-Global Matching for Trinocular Stereo. In: Photogrammetrie - Fernerkundung - Geoinformation, (6), pp. 405–414, (2007)

[63] *Heipke, C.:* A Global Approach for Least Squares Image Matching and Surface Reconstruction in Object Space. In: Photogrammetric Engineering & Remote Sensing (PE & RS), 58(3), pp. 317–323, (1992)

[64] *Heipke, C., Mayer, H., Wiedemann, C., and Jamet, O.:* Evaluation of Automatic Road Extraction. In: International Archives of Photogrammetry and Remote Sensing 32 (Part 3-2W3) pp. 47–56, (1997)

[65] *Hernández, C. and Schmitt, F.:* Silhouette and Stereo Fusion for 3D Object Modeling. In: Computer Vision and Image Understanding (CVIU), Special issue on "Model-based and image-based 3D Scene Representation for Interactive Visualization", 96(3), pp. 367–392, (2004)

[66] *Hilton, A., and Illingworth, J.:* Marching Triangles: Delaunay Implicit Surface Triangulation. Technical Report CVSSP 01, University of Surrey, UK, (1997)

[67] *Hirschmüller, H.:* Stereo Processing by Semi-Global Matching and Mutual Information. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 30(2), pp. 328–341, (2008)

[68] *Hirschmüller, H.:* Stereo Vision in Structured Environments by Consistent Semi-global Matching. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), (2), pp. 2386–2393, (2006)

[69] *Hirschmüller, H. and Scharstein, D.:* Evaluation of Stereo Matching Costs on Images with Radiometric Differences. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, (TPAMI), 31(9), pp. 1582–1599, (2009)

[70] *Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W.:* Surface Reconstruction from Unorganized Points. In: Proc. ACM SIGGRAPH, 26(2), pp. 71–78, (1992)

[71] *Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W.:* Mesh Optimization. In: Proc. ACM SIGGRAPH, 27, pp. 19–26, (1993)

[72] *Horn B. and Schunck B.:* Determining Optical Flow. In: Artificial Intelligence, (17), pp. 185–203, (1981)

[73] *Hoschek, J. and Lasser, D.:* Fundamentals of Computer Aided Geometric Design. AK Peters, Wellesley, MA, USA, (translated by L. L. Schumaker), (1993).

[74] *Kang, S.-B., Szeliski, R., and Chai, J.:* Handling Occlusions in Dense Multi-view Stereo. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR), (1), pp. 103–110, (2001)

[75] *Kazhdan, M.:* Reconstruction of Solid Models from Oriented Point Sets. In: Symposium on Geometry Processing, pp. 73–82, (2005)

[76] *Keriven, R. and Faugeras, O.:* Variational Principles, Surface Evolution, PDEs, Level Set Methods, and the Stereo Problem. Technical report N.3021 at Institut national de recherche en informatique et automatique (INRIA), (3), (1996) and in: IEEE Transactions on Image Processing 7 (3), pp. 336–344, (1998)

[77] *Klaus, A., Sormann, M., and Karner, K.:* Segment-based Stereo Matching Using Belief Propagation and a Self-adapting Dissimilarity Measure. In: IEEE Int. Conf. on Pattern Recognition (ICPR), Hong Kong, China, pp. 15–18, (2006)

[78] *Klein, J. and Zachmann, G.:* Point Cloud Surfaces using Geometric Proximity Graphs. In: Computers & Graphics 28(6), pp. 839–850, (2004)

[79] *Kolmogorov, V.:* Graph Based Algorithms for Scene Reconstruction from Two or More Views. PhD thesis, Cornell University, UK, (2003)

[80] *Kolmogorov, V. and Zabih, R.:* Multi-camera Scene Reconstruction via Graph Cuts In: European Conference on Computer Vision (ECCV) in Copenhagen, Denmark, pp. 65–81, (2002)

[81] *Kolmogorov, V. and Zabih, R.:* Computing Visual Correspondence with Occlusions using Graph Cuts. In: Proc. IEEE Int. Conf. on Computer Vision (ICCV), pp. 508–515, (2001)

[82] *Kruskal, J. B.:* On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In: Proc. of the American Mathematical Society, 7(1), pp. 48–50, (1956)

[83] *Kutulakos, K. N. and Seitz, S.:* A Theory of Shape by Space Carving. In: International Journal of Computer Vision (IJCV), 38(3), pp. 199–218, (2000)

[84] *Lavery, J.:* Shape-preserving Approximation of Multiscale Univariate Data by Cubic L_1 Spline Fits. In: Computer Aided Geometric Design, 21, pp. 43–64, (2004)

[85] *Lavery, J.:* Shape-preserving, Multiscale Interpolation by Bi- and Multivariate Cubic L_1 Splines. In: Computer Aided Geometric Design, 18, pp. 321–343, (2001)

[86] *Leberl, F., Kluckner, S., Pacher, G., Grabner, H., Bischof, H., and Gruber, M.:* Detecting Cars in Aerial Imagery for Improvements of Orthophotos and Digital Elevation Models. In: Proc. ASPRS Annual Conference, (2008)

[87] *Lei, C., Selzer, J., and Yang, Y. H.:* Region-Tree Based Stereo Using Dynamic Programming Optimization. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR), New York, USA, pp. 2378–2385, (2006)

[88] *Lin, Y.-M., Zhang, W., Wang, Y., Fang, S.-C., and Lavery, J.:* Computationally Efficient Models of Urban and Natural Terrain by Non-iterative Domain Decomposition with L_1-Smoothing Splines. In: Proc. 25th Army Science Conf., Department of the Army, Washington DC, USA, PP–08, (2006)

[89] *Lindstrom, P. and Turk, G.:* Fast and Memory Efficient Polygonal Simplification. In: Proc. of IEEE Visualization, pp. 279–286, (1998)

[90] *Loop, C. and Zhang, Z.:* Computing Rectifying Homographies for Stereo Vision. Technical Report MSR-TR-99-21, Microsoft Research, (1999)

[91] *Lorensen, W. E. and Cline, H. E.:* Marching Cubes: a High Resolution 3D Surface Construction Algorithm. In: Computer Graphics, 21, pp. 163–170, (1987)

[92] *Lowe, D. G.:* Distinctive Image Features from Scale-Invariant Keypoints. In: International Journal of Computer Vision (IJCV), 60(2), pp. 91–110, (2004)

[93] *Lu, D., Zhao, H., Jiang, M., Zhou, S., and Zhou, T.:* A Surface Reconstruction Method for Highly Noisy Point Clouds. In: Lecture Notes in Computer Science, Springer, (2005)

[94] *Lucas, B., Kanade, T.:* An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proc. 7th International Joint Conference on Artificial Intelligence (IJCAI), pp. 674–679, (1981)

[95] *Matas, J. and Chum, O.:* Randomized Ransac with $T_{d,d}$-test. In: Image and Vision Computing, 22(10), pp. 837–842, (2004)

[96] *Matoušak, M.:* Epipolar Rectification Minimising Image Loss. PhD thesis in Computer Vision at Center for Machine Perception, CTU, Prague, Czech Rep., (2007)

[97] *Mayer, H. and Bartelsen, J.:* Automated 3D Reconstruction of Urban Areas from Networks of Wide-Baseline Image Sequences. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 37 (B5), pp. 633–638, (2008)

[98] *Mayer, H. and Ton, D.:* 3D Least-Squares-Based Surface Reconstruction. In: Photogrammetric Image Analysis (PIA-07), (3), Munich, Germany, pp. 69–74, (2007)

[99] *Manessis, A., Hilton, A., Palmer, P., McLauchlan, P. and Shen, X.:* Reconstruction of Scene Models from Sparse 3D structure. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR), (1), pp. 666–673, (2000)

[100] *Mederos, B., Velho, L., and De Figueiredo, L. H.:* Smooth Surface Reconstruction from Noisy Cloud. In: Journal of the Brazilian Computing Society, (2004)

[101] *Meidow, J.* Consideration of Uncertainty in Computer Vision: Necessity and Chance. In: Open German Russian Workshop (OGRW) in Ettlingen, Germany (2007) and in Pattern Recognition and Image Analysis (PRIA), pp. 216–221, (2008)

[102] *Meketon, M.:* Least Absolute Value Regression. Bell Labs Technical Report, (1986)

[103] *Morris, D. D. and Kanade, T.:* Image-consistent Surface Triangulation. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 332–338, (2000)

[104] *Mount, D. M. and Arya, S.:* ANN: A Library for Approximate Nearest Neighbor Searching, Version 1.1.1, http://www.cs.umd.edu/∼mount/ANN/, (2006)

[105] *Nistér, D.:* Automatic Dense Reconstruction from Uncalibrated Video Sequences. PhD thesis, Royal Institute of Technology KTH, Stockholm, Sweden, (2001)

[106] *Osher, S. and Paragios, N.:*, Geometric Level Set Methods in Imaging. Vision and Graphics, Springer, (2002)

[107] *Overveld, V. and Wyvill, B.:* Potentials, Polygons and Penguins. An Efficient Adaptive Algorithm for Triangulating an Equipotential Surface. In: Proc of 5th Annual Western Computer Graphics Symposium, 23(3), pp. 31–62, (1993)

[108] *Pajarola, R.:* Overview of Quadtree-based Terrain Triangulation and Visualization. Technical Report UCI-ICS-02-01, Information & Computer Science, University of California Irvine, (2002)

[109] *Penkov, B. I. and Sendov, B. H.:* Hausdorff Metric and its Applications. In: Num. Meth. Approx. (1), pp. 127–146, (1972)

[110] *Pollefeys, M.:* Self-calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences. PhD thesis, ESAT-PSI, K. U. Leuven, Belgium, (1999)

[111] *Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G., and Towles, H.:* Detailed Real-Time Urban 3D Reconstruction From Video. In: International Journal of Computer Vision (IJCV), 78(2-3), pp. 143–167, (2008)

[112] *Šara, R.:* Finding the Largest Unambiguous Component of Stereo Matching. In: Proc. seventh European Conference on Computer Vision (ECCV) (2), Springer-Verlag LNCS 2352, pp. 900–914, (2002)

[113] *Scharr, H.:* Towards a Multi-camera Generalization of Brightness Constancy, In: Complex Motion, 1. Int. Workshop (IWCM), Günzburg, Germany, pp. 78–90, (2004)

[114] *Scharstein, D. and Szeliski, R.:* Stereo Matching with Nonlinear Diffusion. In: International Journal of Computer Vision (IJCV), 28(2), pp. 155–174, (1998)

[115] *Scharstein, D. and Szeliski, R.:* A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms. In: International Journal of Computer Vision (IJCV), 47(1), pp. 7–42, (2002). Images, ground truth and evaluation results available at: http://vision.middlebury.edu/∼schar/stereo/web/results.php

[116] *Schlüter, M.:* Von der $2\frac{1}{2}$ zur 3D Flächenmodellierung für 3D Rekonstruktion im Objektraum. PhD thesis at TU Darmstadt, Germany, (1999) (short version in English: Multi-Image Matching in Object Space on the Basis of a General 3-D Surface Model Instead of Common 2.5-D Surface Models and its Application for Urban Scenes. In: International Archives of Photogrammetry and Remote Sensing, (32), (2000)

[117] *Schnabel, R., Wahl, R. and Klein, R.:* Effcient RANSAC for Point-Cloud Shape Detection. In: Computer Graphics Forum 26(2), pp. 214–226, (2007)

[118] *Schumaker, L.:* Computing Optimal Triangulations using Simulated Annealing. In: Computer Aided Geometric Design, 10(3-4), pp. 329–345, (1993)

[119] *Seidel, R:* Constrained Delaunay Triangulations and Voronoi Diagrams with Obstacles. Technical Report 260, Inst. for Information Processing, Graz, Austria, (1988)

[120] *Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R.:* A Comparison and Evaluation of Multi-view Stereo Reconstruction Algorithms. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), (1), pp. 519–526, New York, USA, (2006)

[121] *Solbrig, P., Bulatov, D., Meidow, J., Thönnessen, U., and Wernerus, P.:* Online Annotation of Airborne Surveillance and Reconnaissance Videos. In: The 11th International Conference on Information Fusion (FUSION 2008), Köln, Germany, pp. 1131–1138, (2008)

[122] *Späth, H.:* Two Dimensional Spline Interpolation Algorithms, A.K. Peters, Wellesley, Massachusetts, (1995)

[123] *Stewénius, H., Schaffalitzky, F., and Nistér, D.:* How Hard is 3-view Triangulation Really. In: Proc. 10th IEEE Int. Conf. on Computer Vision (ICCV), San Diego, USA, (2005)

[124] *Strecha, C.:* Multi-view Stereo as an Inverse Inference Problem, PhD Dissertation, KU Leuven, Belgium, (2007)

[125] *Strecha, C., von Hansen, W., Van Gool, L., Fua, P., and Thönnessen, U.:* On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Anchorage, USA, (2008). Images and ground truth available at: http://cvlab.epfl.ch/∼ strecha/multiview/denseMVS.html

[126] *Surazhsky, V. and Gotsman, C.:* A Qualitative Comparison of Some Mesh Simplification Software Packages. Technical Report, (2005)

[127] *Szeliski, R. and Coughlan, J.:* Spline-based Image Registration. In: International Journal of Computer Vision (IJCV), 22(3), pp. 199–218, (1997)

[128] *Tian, Q. and Huhns, M. N.:* Algorithms for Subpixel Registration. In: Graphical Models and Image Processing (CVGIP), 35(2), pp. 220–233, (1986)

[129] *Turk, G. and Levoy, M.:* Zippered Polygon Meshes from Range Images. In: Proc. ACM SIGGRAPH, 28, pp. 311–318, (1994)

[130] *Trummer, M., Denzler, J., and Munkelt, C.:* KLT Tracking Using Intrinsic and Extrinsic Camera Parameters in Consideration of Uncertainty. In: Int. Conf. Computer Vision Theory and Applications (VISAPP), pp. 346–351, (2008)

[131] *Vanderbei, R., Meketon, M., and Freedman, B.:* A Modification of Karmarkar's Linear Programming Algorithm. In: Algorithmica 1, pp. 395–407, (1986)

[132] *Veksler, O.:* Stereo Correspondence by Dynamic Programming on a Tree. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR) (2), pp. 384–390, (2005)

[133] *Viola, P. A. and Wells, W. M.:* Alignment by Maximization of Mutual Information. In: International Journal of Computer Vision (IJCV), 24(2), pp. 137–154, (1997)

[134] *Wang, J. and Oliveira, M. M.:* Filling Holes on Locally Smooth Surfaces Reconstructed from Point Clouds. In: Image Vision Comput., 25(1), pp. 103–113, (2007)

[135] *Wrobel, B.:* Digital Image Matching by Facets Using Object Space Models. In: Symposium on Optical and Optoelectronic Applied Science and Engineering – Advances in Image Processing, (804), SPIE, pp. 325–333, (1987)

[136] *Zabih, R. and Woodfill, J.:* Non-parametric Local Transforms for Computing Visual Correspondence. In: Proc. of the European Conference of Computer Vision (ECCV), Stockholm, Sweden, pp. 151–158, (1994)

[137] *Zhang, H., Čech, J., Šara, R., Wu, F., and Hu, Z.:* A Linear Trinocular Rectification Method for Accurate Stereoscopic Matching. In: Proc. of British Machine Vision Conference (BMVC), pp. 281–290, (2003)

Chapter 8

Appendix

Selected Algorithms Used in the Dissertation

for $i = 1 : M$ **do** *% number of pixels*
  evaluate $I_2^\omega = \mathcal{I}_0(\omega(\mathbf{x}_i)$
  for $k = 1 : K$ **do** *% $K+1$ number of cameras*
    for $j = 1 : S$ **do** *% number of depth labels*
      obtain $\mathbf{x}_{ik}(j)$ from $\mathbf{x}_i$ and d_j *% with eq. of Sec. 4.1*
      if $\mathbf{x}_{ik}(d) \in \mathcal{I}_k$ **then**
        evaluate $I_2^\omega = \mathcal{I}_k(\omega(\mathbf{x}_{ik}(j))$ *% e. g. bilinear interp.*
        and compute $c_k(i,j)$ from I_1^ω, I_2^ω *% with eq. of Sec. 2.2*
        Set $C(k,j) = c_k(i,j)$
      else
        set $C(k,j) = \infty$
      end if
    end for
  end for
  for $j = 1 : S$ **do**
    aggregate $C(k,j)$ into $E_{data}(\mathbf{x}, j)$ *% using e. g. (4.20)*
    store $\mathcal{A}(j,i) = E_{data}(\mathbf{x}, j) + E_T(\mathbf{x}, j)$ *% using (4.21) and (4.22)*
  end for
end for

Algorithm 8.1: Dense simultaneous pixel matching.

initialize $C(j) = c(1, j), P(i, j) = \emptyset$
for $i = 1 : M - 1$ **do**
  for $j = 1 : S$ **do**
    compute: $C_1(j) = \min_{j'} (C(j') + c_s(j', j))$
    and set $P(i, j) = \arg\min_{j'} (C(j') + c_s(j', j))$
  end for
  for $j = 1 : S$ **do**
    $C(j) = C_1(j) + c(i, j)$
  end for
end for
$j_M = \arg\min(C(j))$
for $i = 1 : M - 1$ **do**
  $j_{M-i} = P(M - i, j_{M-i+1})$
end for

Algorithm 8.2: Dynamic programming algorithm.

procedure rtdqtSplit(T)
if exists $B = \text{friend}(T)$ **then**
  $u = s(B)$
  if u == 1 **then**
    split(B)
  else if u == 0 **then**
    $P = \text{parent}(B)$ *% since $s(T) = 1$ and $s(B) = 0$, $s(P) = 1$*
    % according to definition of RTDQT
    rtdqtSplit(T) *% and so B becomes active*
    split(B)
  end if
end if
split(T)

procedure split(T)
$s(T) = 0$
$s(\text{children}(T)) = 1$
$g(\text{children}(T)) = g(T) + 1$ *% increase generation*

Algorithm 8.3: One step of the (recursive) algorithm for restricted top-down quadtree triangulation. For necessary definitions, see text.

Get N = total number of triangles in $\mathcal{T}_k$
for $j = 1 : N$ **do** *%Initialize*
  Set $a(j) = 0, r(j) = 0, o(j) = 0$ *% area, redundancy, occlusion counter*
end for
for $\mathbf{x} \in \mathcal{T} \cup I_m$ **do**
  determine j such as $\mathbf{x} \in T_j$ *% see Sec. 4.3.3*
  retrieve $\mathcal{D}_m(\mathbf{x})$ and calculate $\mathbf{X}$ *% using (4.9) and (4.2)*
  set $a(j) = a(j) + 1$ and set status = 1
  while status and $k < m - 1$ **do**
    $k = k + 1$
    project $\mathbf{X}$ with P_k to obtain $\mathbf{x}_k$
    if $\mathbf{x}_k \in T_k$ and T_k surface-consistent **then** *% $T \in I_k$!*
      retrieve $\delta = \mathcal{D}_m(\mathbf{x}_i) - d(\mathbf{X})$ *% $d(\mathbf{X})$ from (4.1)*
      if $|\delta| < \varepsilon d(\mathbf{X})$ **then** *% $\mathbf{X}$ is appr. the same point*
        set $r(j) = r(j) + 1$, set status = 0
      else if $\delta > \varepsilon d(\mathbf{X})$ **then** *% $\mathbf{X}$ blocks T*
        set $o(j) = o(j) + 1$, set status = 0
      end if
    end if
  end while
end for
for $j = 1 : N$ **do**
  if $o(j) > 0.1a(j)$ or $o(j) + r(j) > 0.99a(j)$ **then**
    T_j is marked as inconsistent with the surface
  end if
end for

Algorithm 8.4: The LIFT algorithm performs geometric evaluation of $\mathcal{T}$ into redundant, consistent and inconsistent with the surface by means of depth maps of previous reference frames. The input is the camera matrices P_k, the corresponding triangulations $\mathcal{T}_k$, the depth maps $\mathcal{D}_k$ and a positive scalar threshold ε.

Set $k = 0$ *% number of iterations*
Set $\mathbf{b} = 0, \omega = 0, \varepsilon = \infty$
while $k < k_{\max}$ and $\varepsilon > \varepsilon_{\max}$ **do**
  $k = k + 1$
  $W =$diag$(1 - |\omega_i|)$ *% ω_i is the ith element of ω*
  solve $W\mathcal{A}\mathbf{b}_{new} = W\mathbf{c}$ for $\mathbf{b}_{new}$ *% least squares solution*
  if $\|\mathbf{b}_{new} - \mathbf{b}\|_1 > \varepsilon_{\max}$ or $k < k_{max}$ **then** *% the normalized L_1-norm*
    compute $\mathbf{r} = \mathbf{c} - \mathcal{A}\mathbf{b}, \mathbf{v} = W^2\mathbf{r}$ *% residual $\mathbf{r}$, temporal vector $\mathbf{v}$*
    $\alpha = \max_i \left(\max\left(\frac{v_i}{1-\omega_i}, \frac{v_i}{1+\omega_i}\right)\right)$
    $\omega = \omega + \mathbf{cv}/\alpha$ *% recompute primal affine weights*
  end if
  set $\mathbf{b} = \mathbf{b}_{new}$
end while

Algorithm 8.5: Primal Affine Algorithm. Given a matrix $\mathcal{A}$ and data vector $\mathbf{c}$, obtain a solution vector $\mathbf{b}$ for (5.2). Two additional parameters are: the maximum number of iterations $k_{\max}$ and the error tolerance $\varepsilon_{\max}$ normalized by a number of nodes $(I+1)(J+1)$.

Acknowledgment

The main impulse for writing this thesis was given to the author by his employer; automatically obtaining textured 3D-models from (particularly airborne) video-sequences was one of the main tasks of the Reconnaissance, Precision and Standoff Capability project at the Research Institute of Optronics and Pattern Recognition (FOM), from 1.1.2010 Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, (IOSB). This project was sponsored by the Federal Office of Defense Technologies and Procurement (BWB). The author expresses his gratitude to the responsible officials of FOM/IOSB and BWB for their confidence and financial support.

From the scientific point of view, I would like to thank, first of all, my supervisor professor Christian Heipke from IPI (Institute of Photogrammetry and GeoInformation) of Hannover for his fair, objective, and competent support and evaluation in the course of my work. He was always accessible for my questions about and beside the dissertation. His strategic analysis, critical insight and capability to realize even rather sophisticated mathematical relations often helped me to choose better formulations and directions of research. His extensive knowledge and enormous experience, among other areas, in the image-based part of my thesis helped me a lot in my work and were very useful for our joint publication [28]. Last but not least, he certainly contributes to a friendly, cooperative, teamwork-oriented atmosphere at the Institute, so that it has always been a pleasure to travel there and to meet my colleagues to which I also express my thanks for their support and interesting conversations.

John Lavery from the Army Research Laboratory in North Carolina, USA, supported me a lot in the area of shape reconstruction. During his stay at FOM (2007-2008), he drew my interest to his main area of research – L_1 splines – and his code for L_1-spline-based surface reconstruction was successfully integrated into the reconstruction pipeline described in this dissertation. Our numerous discussions and our joint publications [24, 25, 26] not only helped me to understand better some of the methods for shape reconstruction mentioned here, but also to learn how to write scientific papers. I thank John Lavery also for his scientific advice and corrections of grammar and spelling of this thesis and I am happy about the continuation of this productive scientific cooperation.

Among my colleagues at FOM-IOSB, I wish to thank Jochen Meidow for his assistance and wise, elaborate comments about outline and form of this thesis. He proposed to me to use Hausdorff distance for quantitative evaluation of reconstruction results and he was the main author of the bundle adjustment software which allowed me to have better input data for the sequences *Gottesaue, Speyer* and *Turntable Houses*. Furthermore, the sequence *Infrared* was recorded on his behalf and reconstructed by means of a SLAM-algorithm [9] coded by him. I also thank Eckart Michaelsen for reading parts of my thesis and pointing out several areas of improvement. Peter Schlippe carried out several flights and recorded the video sequences *Gottesaue* and *Speyer*.

Within my project group, I wish to thank Peter Solbrig as a very fair, patient, insightful

head of project group who allowed me neglecting – often during days or weeks – my work on the project every time when it was necessary. Also our department heads, Ulrich Thönnessen and (later) Karsten Schulz, were always fair and insightful. Peter Solbrig and Peter Wernerus – whose implementation of the semi-global optimization algorithm [67] and whose work as my coauthor in [28, 29] I appreciate a lot – shared the same office with me which not only gave us a steady possibility of interesting, fruitful discussions, but also contributed to a creative, unstressed atmosphere.

Several important functions used in the library were downloaded from the internet. These are: the Gridfit routine due to J. D'Errico, graph-cuts-based disparity estimation due to V. Kolmogorov, the method of water-tight iso-surface extraction due to M. Kazhdan and the ANN-library due to D. M. Mount. I thank the first two authors for fruitful discussions and all authors for their excellent work.

I also express my deep gratitude to my parents, my sister and my girlfriend. They always knew how to support me and to bring me to positive thoughts, joy and happiness. They also felt when I needed their company and when I needed to be alone. My last thank goes to my good friends from near and far, it helped me enormously to have them and to feel them think about me and give me the inspiration and necessary distraction.

Curriculum Vitae

Personnel Details

Name	Dimitri Bulatov
Address	Fraunhofer IOSB Gutleuthausstr, 1 – 76275 Ettlingen – Germany
Email	dimitri.bulatov@iosb.fraunhofer.de
Date of birth	21 March 1978
Place of birth	Moscow, Russian Federation
Nationality	German, Russian

Work Experience

Since 09/2005	Institute of Optronics and Pattern Recognition (from 1.1.2010: Fraunhofer IOSB), Ettlingen, Germany Research assistant in the Scene Analysis Division, responsible for 3D reconstruction from video sequences
09/2001 - 08/2005	University of Würzburg, Germany Tutor at the faculty of mathematics, responsible for composing, correcting and explaining tutorials and exams

Education

1999 - 2005	University of Würzburg and Universidad Autonoma de Madrid, Spain (Erasmus program) Diploma in Mathematics about star-like complex functions Minor subject: physics Additional diploma in Spanish Philology
1996 - 1999	Roentgen High school of Würzburg High school graduation in subjects: Mathematics, Physics, English and History
1985 - 1995	Primary school (Nr. 444) and (mathematical) Secondary School (Nr. 91) in Moscow, Russia