Hoang Anh Duc TON

3D Least Squares Based Surface Reconstruction

der Bundeswehr
Universität München

Neubiberg 2011

3D Least Squares Based Surface Reconstruction

Vollständiger Abdruck der von der

Fakultät für Bauingenieur- und Vermessungswesen der

Universität der Bundeswehr München

zur Erlangung des akademischen Grades

eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation

vorgelegt von

M.Sc. Duc Ton

Neubiberg 2011

Prüfungskommission:

Vorsitzender: Univ.-Prof. Dr.-Ing. Friedrich Kröll

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Helmut Mayer

2. Univ.-Prof. Dr.-Ing. Christian Heipke (Leibniz Universitat Hannover)

Die Dissertation wurde am 15.12.2011 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 15.12.2011 angenommen.

Abstract

This thesis presents a fully three dimensional (3D) surface reconstruction algorithm from wide-baseline image sequences. Triangle meshes represent the reconstructed surfaces allowing for an easy integration of image- and geometry-based constraints. We extend the successful approach for 2.5D reconstruction of HEIPKE (1990) to full 3D. To take into account occlusion and non-Lambertian reflection, we apply robust least squares adjustment to estimate the model. The input for our approach are images taken from different positions and derived accurate image orientations as well as sparse 3D points (BARTELSEN and MAYER 2010). The first novelty of our approach is the way we position additional 3D points (unknowns) in the triangle meshes constructed from given 3D points. Owing to the precise positions of these additional 3D points, we obtain more precise and accurate reconstructed surfaces in terms of shape and fit of texture. The second novelty is to apply individual bias parameters for different images and adapted weights for different image observations to account for differences in the intensity values for different images as well as to consider outliers in the estimation. The third novelty is the way we factorize the design matrix and divide the meshes into layers to reduce the run time. The essential element for our model is the variance of the intensity values of image observations inside a triangle. Applying the approach, we can reconstruct accurate 3D surfaces for different types of scenes. Results are presented in the form of VRML (Virtual Reality Modeling Language) models, demonstrating the potential of the approach as well as its current shortcomings.

Zusammenfassung

Diese Arbeit präsentiert einen vollständig dreidimensionalen (3D) Algorithmus zur Oberflächenrekonstruktion aus Bildfolgen mit großer Basis. Die rekonstruierten Oberflächen werden durch Dreiecksgitter beschrieben, was eine einfache Integration von Bild- und Geometrie-basierten Bedingungen ermöglicht. Die vorgestellte Arbeit erweitert den erfolgreichen Ansatz von HEIPKE (1990) zur 2,5D Rekonstruktion zur vollständigen 3D Rekonstruktion. Verdeckung und nicht-Lambertsche Spiegelung werden durch robuste kleinste Quadrate Ausgleichung zur Schätzung des Modells berücksichtigt. Ausgangsdaten sind Bilder von verschiedenen Positionen, abgeleitete genaue Orientierungen der Bilder und eine begrenzte Zahl von 3D Punkten (BARTELSEN and MAYER 2010). Die erste Neuerung des vorgestellten Ansatzes besteht in der Art und Weise, wie zusätzliche Punkte (Unbekannte) in dem Dreiecksgitter aus den vorgegebenen 3D Punkten positioniert werden. Dank den genauen Positionen dieser zusätzlichen Punkte werden präzisere und genauere rekonstruierte Oberflächen bezüglich Form und Anpassung der Bildtextur erhalten. Die zweite Neuerung besteht darin, dass individuelle Bias-Parameter für verschiedene Bilder und angepasste Gewichtungen für unterschiedliche Bildbeobachtungen verwendet werden, um damit unterschiedliche Intensitäten verschiedener Bilder als auch Ausreißer zu berücksichtigen. Die dritte Neuerung sind die verwendete Faktorisierung der Design-Matrix und die Art und Weise, wie die Gitter in Ebenen zerlegt werden, um die Laufzeit zu reduzieren. Das wesentliche Element des vorgestellten Modells besteht in der Varianz der Intensitätswerte der Bildbeobachtungen innerhalb eines Dreiecks. Mit dem vorgestellten Ansatz können genaue 3D Oberflächen für unterschiedliche Arten von Szenen rekonstruiert werden. Ergebnisse werden als VRML (Virtual Reality Modeling Language) Modelle ausgegeben, welche sowohl das Potential als auch die derzeitigen Grenzen des Ansatzes aufzeigen.

Contents

Chapter 1

Introduction

1.1 Motivation

This thesis deals with dense 3D surface reconstruction. The purpose is to compute a 3D surface model of a scene from multiple images. The thesis focuses on two main problems in dense surface reconstruction, namely the matching algorithm and the technique applied to optimize the objective function in order to derive a photorealistic reconstructed shape.

The topic of recovering the surface shape from images has received tremendous attention in the computer vision and photogrammetry community particularly in recent years. In spite of this, the topic is still interesting with many remaining open questions. While there are many approaches which can successfully solve the problem in a specific direction, there are still weaknesses and unsolved problems in each approach. I.e., there is no perfect approach that can handle all issues and produces the best possible output. For example, some approaches achieve very good results, but are computationally very expensive. Some methods produce very good results, but do not work for all types of objects, because of strong assumptions such as Lambertian surfaces, low image noise, or no occlusions. In addition, some methods only take into account the image information and mostly ignore geometric cues which sometimes are very important for the reconstruction. In the next paragraphs, we will summarize several recent approaches to clarify our motivation.

Among recent publications describing the reconstruction of surfaces from 2D images are many volumetric algorithms (FAUGERAS and KERIVEN 1998, KOLMOGOROV and ZABIH 2002, KUTULAKOS and SEITZ 1999, SEITZ and DYER 1997) which apply 3D grids for representing the geometry to simultaneously reconstruct surfaces and to obtain dense correspondences. On the other hand, a lot of algorithms use polygon meshes or depth maps in reconstructing the shape and the reflectance properties of an object (FUA and LECLERC 1994, HEIPKE 1990, AKBARZADEH et al. 2008). All of these approaches have potential, but they still have limitations.

For the 3D grids, (KUTULAKOS and SEITZ 1999, SEITZ and DYER 1997) are two typical approaches that use a voxel representation for the geometry of a scene. Space carving (KUTULAKOS and SEITZ 1999) or voxel coloring (SEITZ and DYER 1997) work directly on voxels enforcing image consistency and visibility. These methods are purely local and therefore

rely either on numerous viewpoints or on well textured surfaces to achieve satisfying results. In addition, they mostly ignore the noise of real images. Even more important is that they don't have a way to impose spatial coherence which is problematic, because image data are often ambiguous.

Level-set methods also apply 3D grids for scene representation. E.g., the approach of Faugeras (1998) intrinsically embodies multiple views and naturally handles topological changes and occlusions. However, it is not clear under what conditions their method converges as the proposed functional is non-convex.

The general limitation of the class of methods making use of 3D grids lies in the initial discretization, because the degree of precision is limited by the resolution of the grid. To increase the quality, higher resolutions are necessary. This, however, results in a time-consuming computation particularly for spatially extended scenes.

For the depth map representation, Kolmogorov and Zabih (2002), Roy (1999), and Ishikawa and Geiger (1998) propose direct discrete minimization formulations that are solved by graph-cuts. These approaches achieve disparity maps with accurate contours, but with limited depth precision. The improvement for graph cuts proposed by Boykov and Kolmogorov (2003) overcomes some of the above limitations. Paris et al. (2003) propose a continuous functional, but restricted to open surfaces.

Methods based on polygon meshes represent surfaces by connected planar facets or triangulated meshes. Fua and Leclerc (1994) and Heipke (1990) among others use this type of scene representation. Fua and Leclerc (1994) represent surfaces by triangulated grids. They do not only combine different sources of information (image-based and geometry-based) in the reconstruction, but they also control the influence of each information source in the reconstruction at each step of the iteration. Owing to this property, they can recover both the shape and the reflectance properties of complicated surfaces. This is difficult for methods in which only one source of information is explicitly considered. The method of Fua and Leclerc (1994) also allows to take into account self-occlusion and shading. Even though it has big advantages over other methods as it can reconstruct complicated surfaces, it still has some limitations. First, it has the strong assumption that the data used to initialize surfaces can easily be separated into separate groups which correspond to specific objects, i.e., surfaces. However, real scenes often consist of several objects with the topology unknown in advance. Thus, this method is not suitable for a highly complex topology. Second, it is possibly time-consuming, because it makes use of hand-collected geometric information as input if high quality surfaces are to be obtained.

Heipke (1990) also uses triangulated meshes, but in a different way than Fua and Leclerc (1994). The main idea of his method is to use least squares optimization to adjust the geometry of the object surface in order to minimize the differences between the brightness of the projected images and a derived average image. On one hand, Heipke's method comprises the advantages of triangulated meshes as in Fua and Leclerc (1994). On the other hand, the method partly overcomes limitations of Fua and Leclerc (1994) as it relaxes their strong assumptions. However, its disadvantages are that it ignores occlusions and non-Lambertian

reflection. Additionally, Heipke's method is proved to be successful only for 2.5D reconstruction, i.e., one z-value for each x-y position, and has not been applied for full 3D reconstruction.

The discussed limitations and strengths of all above methods imply that it seems promising to extend Heipke's method to full 3D surface reconstruction while applying the 3D triangulated mesh representation of Fua and Leclerc (1994) and taking care of occlusions and non-Lambertian reflection. We summarize our proposed approach in the next section.

1.2   Summary of Proposed Approach

The objective of our research is to set up a system for surface reconstruction from images from different view points. Our approach which was first introduced in Ton and Mayer (2007) has the following properties:

- The surface shape is recovered together with the corresponding texture image from multiple images by means of least squares adjustment.

- The surface can be constructed from many images, even when the images are taken from possibly widely differing viewpoints.

- The system deals with viewpoint dependent effects such as non-Lambertian reflection of the surface and occlusions by means of robust estimation.

We employ the 3D triangular facets mentioned in the previous section proposed by Fua and Leclerc (1994) and extend the classical least squares matching approaches of (Wrobel 1987, Ebner and Heipke 1988, Heipke 1990) in the direction of full 3D reconstruction from wide-baseline image sequences. Our way to deal with occlusions is inspired by Schlüter (1998).

Given sparse 3D points are used as input for the reconstruction. To extract these 3D points from the scene, we use the approach of Bartelsen and Mayer (2010) resulting in reliable, precise, and accurate points. Bartelsen and Mayer (2010) apply robust bundle adjustment and least squares matching to extract 3D points. They assume that the camera calibration is known and also provide the information which cameras view which extracted points as well as the order of all cameras.

On the projections of the given sparse 3D points in the images we apply 2D Delaunay triangulation to connect the 3D points and obtain a triangulated mesh. Because the Delaunay algorithm is designed for individual images, we adjust it so that it can be applied for multiple images. This is based on knowledge about which points, i.e., surface parts, are seen through which images. We only obtain a limited number of 3D points from the technique of Bartelsen and Mayer (2010). Thus, if we would only use such given points for surface reconstruction, we would not make use of all information available from the images. In addition, because of the limited number of extracted 3D points, we cannot get highly precisely reconstructed surfaces and texture properties, especially for strongly curved areas.

Therefore, based on these extracted 3D points, we determine the positions of other points from the images, which become the unknowns of our adjusment. The main step of our technique is then to estimate the position of the unknowns: The more precise and reliable the unknowns are determined, the better will be the fit of the reconstructed surface.

We define the unknowns as points which lie on a side of the triangles derived from the given sparse 3D points. An unknown implies the generation of a new 3D point. From this new 3D point, new triangles are created. Every time a new unknown is positioned, the triangulated mesh is split. By splitting triangles by means of unknowns and projecting a number of points inside the triangles into the images, the image observations are obtained. To take into account occlusions when positioning unknowns, we start from the first camera and end the iterative procedure at the last camera. Thus, we first consider triangles that can be observed by the first camera. We define unknowns for this group of triangles, i.e., part of the mesh. The unknowns of the first camera then become the given 3D points when applying the procedure to obtain unknowns for the second camera, etc. In addition, we employ a coarse to fine procedure with a hierarchy of resolutions for the triangulation linked to adequate levels of image pyramids to expand the range of convergence of the adjustment.

To define an unknown, we need to answer three questions: The first question is whether an unknown is needed in an existing triangle of the mesh, i.e., whether a triangle should be split. The other two questions are: (1) On which edge of the split triangle should we position the unknown, and (2) where on this edge should the unknown be? We will discuss the answers to these questions in Chapter 4. To precisely position the unknowns, we move the corresponding vertices of the triangulation resulting from the densification of the triangulation in the direction of their normals. To get the best estimate of the unknowns, we apply robust M-estimators and extend the optimization procedure proposed by HEIPKE (1990). We also apply a smoothing term to account for the ill-posedness of 3D surface reconstruction due to noise and occlusions by enforcing a low curvature in the sense that the vertices of the triangulation should be close to the average plane of their direct neighbors. An analysis of a local neighborhood of the unknowns leads to an additional smoothness observation in which the regularization is based on additional observations modeling the local curvature of the surface.

1.3   Outline of the Thesis

The dissertation is made up of six chapters. The second and third chapter give the theoretical background in which we discuss in detail methodologies and models that we apply in our thesis. Specifically, in **Chapter 2** we present the fundamental model and method we use in this dissertation. As we employ robust least squares optimization to deal with non-Lambertian reflections and occlusions, in this chapter, we will describe this technique and discuss its strengths and weaknesses compared to other techniques.

In **Chapter 3**, we focus on basic stereo matching issues as the main focus of this thesis. As foundation for our methodology, we describe a multi-view stereo taxonomy. We summa-

rize the methodology as well as the advantages and disadvantages of several recent stereo algorithms to make clear why we have devised the proposed algorithm the way we did. The presented 3D surface reconstruction approaches use voxel coloring, space carving, level sets, triangulated meshes, and particles. At the end of this chapter, we discuss the technique of HEIPKE (1990) which is the basis of our approach.

The remaining part of the thesis is dedicated to our new approach, experiments conducted with the approach and results obtained by applying the approach. More specifically, in **Chapter 4** we describe our novel approach for surface reconstruction which permits to reconstruct a fully 3D surface from multiple images. Our method uses a triangulated grid, least squares optimization, and regularization by smoothing the surface. In the first section, we describe foundations for our approach: Camera model, formation of images, sub-pixel interpolation, pose estimation, and 3D point determination. In the next two sections, we present the basic idea of our approach and how we derive the partial derivatives for the least squares design matrix. We then describe how we build the triangulated mesh from the given sparse 3D points. In the fifth section, we present a coarse-to-fine strategy consisting of three steps to position additional points, i.e., the unknowns of the least squares estimation in the triangulated meshes built in the previous section. In the first step, a number of additional 3D points in each triangle of the 3D meshes and their corresponding image observations are constructed. This allows us to determine which triangles should have unknowns, i.e., should be split, based on the deviation of the intensity values inside a triangle (second step). In the third step, unknowns are initially positioned in the selected triangles. The last section of this chapter explains how robust least squares adjustment is applied in order to obtain accurate and precise positions for the unknowns in the 3D meshes. We describe how we achieve robustness for occlusion and non-Lambertian reflection by controlling bias parameters and different weights for different image observations. Finally, we show how we factorize the design matrix to reduce calculation time.

In **Chapter 5**, we discuss experiments with our approach. The first section presents preparations including why and how the data set of given 3D points extracted from images by the method of BARTELSEN and MAYER (2010) is manually adjusted so that it is suitable for our approach. Second, it is discussed which percentage of triangles in a mesh should be selected as unknowns. Third, it is shown how different layers in a mesh are created to reduce the time of the estimation. Fourth, the robust least squares adjustment model is specified including initial values for bias parameters and weights, weight adjustment during estimation, and the influence ratio between image observations and smoothness observations to control for the smoothness of the surfaces. The main part of Chapter 5 is the second section where we present and discuss all results, i.e., reconstruction of 3D surfaces by application of the approach. Particularly, we demonstrate the improvement by reconstructing surfaces with additional 3D points (unknowns) compared to the surfaces reconstructed from the given 3D points.

Chapter 6 consists of the conclusion and our outlook for future research.

1.4   Main Findings

To show the advantages of our approach, we apply it for the reconstruction of 3D surfaces for four different types of scenes. The first is a corner of Trinity building consisting of different types of structures on its surface (glass windows, stone columns, bushes), but weak texture. The second scene shows a cylinder covered by many advertising posters. It has a simple structure and mostly a strong texture. The third scene consists of a street corner in Leuven, Belgium. Besides weak texture and a complicated 3D structure, the images show both fixed and moving occlusions (people). The fourth scene is the main gate of the Cathedral in Cologne, Germany. The scene has a sophisticated surface with many small statues.

The experimental results demonstrate that our approach can successfully reconstruct the 3D surfaces of these objects. The creation of additional 3D points additionally to the given sparse 3D points is the basic principle of the reconstruction. It is shown that the additional points significantly reduce the error in the intensity values of the reconstructed meshes. As result, we obtain more precise and accurately reconstructed surfaces.

The main findings for our approach can be summarized as follows: First, the approach is proved to be suitable for different types of objects with from weak to strong texture as well as from simple to complicated structure. Second, the approach robustly deals with problems due to non-Lambertian reflection and occlusion which is a remaining issue of many current algorithms. Third, we show that the combination of a mesh representation with least squares adjustment allows to successfully extend Heipe's approach for 3D reconstruction. Fourth, the 3D given points extracted by the approach of Bartelsen and Mayer (2010) are essential for the success of our approach. Fifth, in order to obtain a very precise 3D surface, we apply different bias parameters for different images and different weights for different image observations. By this means, we account for the differences in the intensity values for images from different camera positions and reduce the effect of outliers on the estimation. Finally, to reduce the run time, we factorize the design matrix and divide the meshes into layers.

Chapter 2

Fundamentals

In this chapter, we describe the fundamental technique which is applied in this thesis: Robust least squares adjustment. In the first section, we present general weighted least squares adjustment. In order to deal with outliers which always exist in image matching, the estimation has to be robust. Thus, in the second section, we focus on robust estimation. We summarize techniques that are currently used in scene reconstruction and explain why we choose an M-estimator.

2.1 Least Squares Adjustment

2.1.1 General Least Squares Adjustment

Least squares (LS) adjustment is the principal mathematical method used in this dissertation. It is a mathematical optimization technique which attempts to find a best function, i.e., a "best fit", for observed data for overdetermined systems. The most important application of this method is data fitting.

Basically, the n×1 vector $E(L)$, i.e., the expected values of n observations $(l_1, l_2, ..., l_n)$ is assumed to be related to the n×1 vector $\varphi(X) = (\varphi_1, \varphi_2, ..., \varphi_n)^T$ by the following functional model:

$$E(L) = \varphi(X), \tag{2.1}$$

where

- $X = (x_1, x_2, ..., x_m)^T$ is an m×1 vector of m unknown parameters and

- φ_i describes the relationship between the unknown parameters and X and an observed value l_i.

As we assume that for real noisy data there is always an error between the expected value and the observed value, we need to add residuals to correct for the error. The LS adjustment

function can thus be written as:

$$E(L) = \varphi(X) = L + \epsilon \tag{2.2}$$

where

- $L = (l_1, l_2, ..., l_n)^T$ is an n×1 vector of observed values and

- $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_n)^T$ is an n×1 vector of residuals for each observed value.

The general idea of the "best fit" in LS is to minimize the sum of the squared residuals (S), which are the differences between the observed value (l_i) and the expected value provided by the model ($\varphi_i(X)$):

$$S = \sum_i (l_i - \varphi_i(X))^2 \tag{2.3}$$

Minimizing S implies $\frac{\partial S}{\partial \varphi_i(X)} = 0$. When the system is non-linear, as in our case, these derivatives are functions of both the unknowns X and the parameters of φ_i and, therefore, one cannot obtain a closed solution for the gradient equations. To solve the problem, one needs to choose initial values for the unknowns X. Denoting the vector of these initial values as X_0, an n×1 vector of corrections to the unknowns Δx can be defined as:

$$\Delta x = X - X_0. \tag{2.4}$$

The unknowns are iteratively refined. At each iteration, the model in equation (2.2) is linearized by an approximation using a first-order Taylor series expansion of the unknowns X_j:

$$l + \epsilon = A\Delta x, \tag{2.5}$$

in which A is the n×m Jacobian matrix of the partial derivatives of the φ_i according to the unknowns X_j:

$$A = \begin{vmatrix} \frac{\partial \varphi_1(X)}{\partial X_1} & \frac{\partial \varphi_1(X)}{\partial X_2} & \cdots & \frac{\partial \varphi_1(X)}{\partial X_m} \\ \frac{\partial \varphi_2(X)}{\partial X_1} & \frac{\partial \varphi_2(X)}{\partial X_2} & \cdots & \frac{\partial \varphi_2(X)}{\partial X_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varphi_n(X)}{\partial X_1} & \frac{\partial \varphi_n(X)}{\partial X_2} & \cdots & \frac{\partial \varphi_n(X)}{\partial X_m} \end{vmatrix}. \tag{2.6}$$

Substituting these expressions into the condition that the partial derivatives have to be zero, the solution for LS estimation is obtained as the following estimate:

$$\widehat{\Delta x} = \left(A^T A\right)^{-1} A^T l. \tag{2.7}$$

Please note that an implicit requirement for LS is that the residuals ϵ in each measurement are random. In general, observed values then are assumed to be normally distributed. Following this assumption, the residuals are also assumed to be normally distributed for all values of

Δx. Thus, the variances of errors are assumed to be constant (σ_0^2) and, therefore, in the above calculation for LS, unit weights are used for all observations. Substituting the above value of Δx into equation (2.5) to obtain a value for $\epsilon = A\Delta x - l$ and then using these corrected values, the estimated value for $\widehat{\sigma}_0$ can be computed as:

$$\widehat{\sigma}_0 = \sqrt{\frac{\epsilon^T \epsilon}{n - m}}. \tag{2.8}$$

According to the Gauss-Markov theorem estimators are efficient unbiased estimates under these assumptions. However, in reality, these assumptions often clearly do not hold, especially in computer vision, where each point might have its different characteristic, i.e., when observations are heteroskedastical rather than homoskedastical making the estimates biased. Therefore, when each observation cannot be treated equally, it is appropriate to apply suitable different weights. Next, we describe how Weighted LS solves the heteroskedasticity problem.

2.1.2   Weighted Least Squares Adjustment

Weighted Least Squares (WLS) makes it possible to incorporate weights associated with each observations into the fitting function. This solves the heteroskedasticity by downweighting the squared residuals for observations with larger variances. The size of the weight represents the precision of the information contained in the associated observation. WLS minimizes the weighted squared error when fitting the criterion to find estimates for unknowns by allowing for the introduction of weights, i.e., the appropriate level of influence of each observation in the estimation. Particularly, the weight for an observation is determined relatively to the weights of other observations.

Denoting by P an $n \times n$ matrix of weights that are applied for n observations of l_i. For the remainder we assume that P has the off-diagonal elements equal set to zero and the sum of diagonal elements is 1. For WLS estimates for unknowns are obtained by minimizing the weighted sum of squared errors ϵ:

$$S = \sum_{i}^{n} P_{ii}(l_i - \varphi_i(X))^2 \tag{2.9}$$

According to AITKEN (1935) $\widehat{\Delta x}$ is the efficient unbiased estimate if each weight P_{ii} is equal to the reciprocal of the variance of the measurement:

$$P_{ii} \propto \frac{1}{\sigma_i^2} \tag{2.10}$$

Minimizing equation (2.9) by setting the partial derivatives equal to zero and solving the systems in the same way as in the previous subsection, we get:

$$(A^T P A)\widehat{\Delta x} = A^T P l \tag{2.11}$$

$$\widehat{\Delta x} \;=\; (A^T P A)^{-1} A^T P l. \tag{2.12}$$

Following this model, the covariance matrix is defined as $C_P = \sigma_0^2 P$, and an estimate $\widehat{\sigma}_0$ for the posterior RMSE of the weights is calculated by:

$$\widehat{\sigma}_0 \;=\; \sqrt{\frac{\epsilon^T P v}{n - m}}. \tag{2.13}$$

Finally, the covariance matrix of the unknown parameters is $C_{XX} = \widehat{\sigma}_0^2 \left(A^T P A \right)^{-1}$.

Advantages/Disadvantges of WLS

As other least squares methods, WLS has advantages and disadvantages. The first and main advantage of WLS over other methods in 3D reconstruction is its ability to handle regression when data points are of different quality. In these cases, it provides the most precise estimates for the unknowns compared to other methods. This is also the main reason why we chose WLS in our study. Second, like all other least squares methods, WLS estimates are also efficient and unbiased and the method also has the ability to provide different types of statistical intervals for estimation, prediction, calibration, and optimization which are essential issues in scene reconstruction.

The biggest disadvantage of the method lies in the the assumption that the weights are known exactly, because in real applications one must use estimated weights instead. We know that when estimated weights are used, the optimality that we get for known weights can not strictly be achieved. Yet, if we can precisely estimate weights, we still can obtain a significant improvement for the unknown estimates compared to those obtained from general LS when equally weights are applied to all data points. Thus, a basic problem in applying this method is to obtain highly precise estimates for weights. Like for other LS methods, the second disadvantage of WLS is its sensitiveness to outliers. When an outlier is not detected and dealt with appropriately, it may have a possibly devastating negative impact on the unknown estimates as WLS can put more influence on an outlier in the estimation. We therefore will discuss how weights are estimated in the next paragraphs and how we deal with outliers in Section 2.2.

Estimation of weights

In general, there are three methods to estimates weights: The first method is the direct one and is based on the assumption that there are replicates in the data. Thus, following equation (2.10), the most obvious way is to set the weight of each observation equal to the reciprocal of the sample variance obtained from the set of replicate measurements to which the observation belongs:

$$P_{ij} \propto \frac{1}{\widehat{\sigma}_i^2} \;=\; \frac{1}{\left[\frac{\sum_{j=1}^{n_i} (l_{ij} - \bar{l}_i)^2}{n_i - 1} \right]}. \tag{2.14}$$

with P_{ij} the weights indexed by predictor variable levels i and replicate measurements j, $\widehat{\sigma}_i$ the sample standard deviation of the response variable at the ith combination of the predictor variable values, n_i the number of replicate observations, and $\bar{l}_i$ the mean of the responses at the ith combination of the predictor variable values. This method, however, rarely works well due to usually extremely estimated weights (NIST/SEMATECH 2010). Thus, the estimated weights cannot correctly control the level of influence that each observation has on the unknown estimates. The method can only work given a large number of replicates at each combination of predictor variables.

The second method is called *Better Strategy for Estimating the Weights*. Its purpose is to find a function that relates the standard deviation of the response at each combination to the predictor variables. If we can write:

$$\widehat{\sigma}_i^2 \approx \zeta_i(X), \tag{2.15}$$

then we can set weights as:

$$P_{ij} = \frac{1}{\zeta_i(X)}. \tag{2.16}$$

This approach usually provides more precise estimates than the first method, because it requires fewer estimated quantities and there is more data for the estimation.

Finally, when there are only few or no replicate measurements for each combination of predictor variable values, the third approach named *Estimating Weights Without Replicates* is applied. With this approach, approximate replicate groups are formed so that we can estimate weights. To form replicate groups, there are three approaches: (1) Formation of groups based on plots of the response against the predictor variables; (2) Division of data into equal-sized groups of observations after sorting by the values of the response variable, and (3) selection of replicate groups based on the range of predictor variable values. By this approach, estimated weights depend on the way we form replicate groups. However, the resulting fitted values of the estimation are typically sensitive to even small changes in the definition of the weights when weights are based on a simple, smooth function. For the experiments in Chapter 5 we will explain in detail (i) our smooth function used to estimate weights, (ii) the way that we choose initial values for unknowns, and (iii) the convergence condition for our regression.

2.2 Robust Estimation

For the common computational problem in computer vision to estimate the parameters of a model from image data there are some key difficulties among which are: (i) The data typically contains outliers, i.e., observations that do not belong to the model being fitted. (ii) Initial guesses for the models must be generated automatically. (iii) Multiple occurrences of the models can be represented in the data. Because of these difficulties, robust techniques need to be applied in computer vision to solve these problems.

Robust estimation is defined as a technique which is insensitive to differences from the idealized assumptions of the algorithm. In computer vision, an algorithm is considered robust if it can tolerate outliers, i.e., data which does not obey the assumed model. This definition is similar to the one used in statistics for robustness (HAMPEL et al. 1986). Robust techniques have been used in computer vision for at least thirty years. The best known and popular methods which are used today originated from older methods which tried to solve specific image understanding or pattern recognition problems. The earliest method that has been applied to deal with outliers and multiple solutions in parameter estimation problems is the Hough Transform which received a US Patent in 1962 (HOUGH 1962). However, today the most popular regression methods in computer vision are the family of random sample consensus (RANSAC) invented by FISCHLER and BOLLES (1981) to solve the perspective n-point problem.

In statistics, robust techniques haven been investigated since the early 1960s with the introduction of M-estimators by Huber in 1964 who followed maximum likelihood considerations. Other classes of robust techniques include L-Estimates which are linear combinations of order statistics and R-Estimates which are based on statistical rank tests (cf. HUBER (1996) for the relevant references). At the beginning, M-estimators were introduced to estimate the 'location' or 'center' of a distribution, only later they were generalized to regressions. In 1984 another popular family of robust estimators named least median of squares (LMedS) was proposed by ROUSSEEUW (1984). These robust techniques in statistics have been applied in computer vision since the end of the 1980s.

By definition, M-estimators consists of a wide class of estimators, which are obtained by minimizing sums of functions of the data. Least squares estimators and many maximum-likelihood estimators belong to the class of M-estimators. As in this thesis our fundamental technique are the WLS, we will use M-estimators for robust estimation. Therefore, this section is structured as follow: (i) In the first part we describe M-estimators and explain how M-estimators help us to solve problems listed above. (ii) We then briefly discuss other robust techniques like the Hough Transform, RANSAC, and LMedS and compare them with M-estimators to show why we choose M-estimators.

2.2.1   M-estimators

In this part, we will only focus on the class of M-estimators that have mostly been suggested in computer vision. According to the M-estimators proposed by Huber, the robust formulation of our objective function in Equation (2.9) is generalized as follows:

$$[\widehat{\Delta x}] = \Delta x \text{argmin} \ \sum_{i}^{n} \rho(\frac{1}{s} g(l_i)) \tag{2.17}$$

with $g(l_i) = l_i - \widehat{l_i} = l_i - A\widehat{\Delta x_i}$ and s the scale parameter depending on σ, the standard deviation of the unknown. For simplicity we set $e_i = \frac{g(l_i)}{s}$.

In the above model, ρ is called the loss function which has the following properties: (i) Nonnegative with $\rho(0) = 0$, (ii) symmetric: $\rho(e) = \rho(-e)$, and (iii) nondecreasing with $|e|$. Thus, when $\rho(e) = e^2$, this model corresponds to the least squares objective function in Equation (2.9). When $\rho(e) = |e|$, we obtain an L_1 estimator.

Given the estimator $\rho(e)$, there are two functions that need to be considered in M-estimates: The influence function which is defined as $\psi(e) = \frac{d\rho}{de}(e)$ and the weight function: $w(e) = \frac{1}{e}\frac{d\rho}{de}(e)$, so $w(e) = \frac{\psi(e)}{e}$. The influence function describes the sensitivity of the overall estimates (Δx) to data with error e while the weight function provides weights for the iteratively reweighted least squares method which is used to solve for the unknowns X. We will discuss this iterative procedure later.

For the M-estimate, the estimated value for Δx in Equation (2.12) is:

$$\widehat{\Delta x} = (A^T P A)^{-1} A^T P l \tag{2.18}$$

with P now the n×n diagonal matrix of non-negative weights:

$$P_{ii} = w(e_i) = \frac{1}{e_i}\frac{d\rho(e_i)}{de} \geq 0 \quad \text{with} \quad e_i = \frac{\widehat{g(l_i)}}{s} \quad i = 1, ...n. \tag{2.19}$$

It is obvious from Equation (2.19) that for M-estimates one needs to apply an iterative procedure, because the residuals ($\widehat{g(l_i)}$) are required to calculate the weights. Given a scale s, the Iterative Weighted Least Squares for M-estimation proceeds as follows:

1. Choose initial estimates X_0 to obtain Δx_0, as least squares estimates.

2. At each iteration t, calculate residuals $\epsilon_i^{(t-1)} = g(l_i)^{(t-1)} = e_i \cdot s$ and the associated weights $P_{ii}^{(t-1)} = w(e_i)$ from the previous iteration.

3. Solve for updated weighted-least squares estimates:

$$\widehat{\Delta x}^{(t)} = (A^T P^{(t-1)} A)^{-1} A^T P^{(t-1)} l \tag{2.20}$$

4. Check the convergence criterion for the estimates. If $\|\widehat{\Delta x}^{(t+1)} - \widehat{\Delta x}^{(t)}\|$ is less than the tolerance, we stop.

5. Replace $\widehat{\Delta x}^{(t)}$ with $\widehat{\Delta x}^{(t+1)}$. Return to Step (ii).

The asymptotic covariance matrix of the unknowns is then calculated as:

$$C_{XX} = \frac{E(\psi^2)}{[E(\psi')]^2}(A^T A)^{-1}. \tag{2.21}$$

We use $\sum[\psi(e_i)]^2$ to estimate $E(\psi^2)$ and $[\sum \psi'(e_i)/n]^2$ to estimate $[E(\psi')]^2$. The resulting estimate for the asymptotic covariance matrix $\widehat{C_{XX}}$ is not reliable for small samples.

Objective functions and different types of M-estimators

There are two functions associated with M-estimators (cf. Fox (Jan 2002)): The loss (objective) function ρ associated with the corresponding influence function ψ – the derivative of ρ – and the weight function w. Different objective functions result in different types of M-estimators. In computer vision, in general three M-estimators are usually considered: (i) The familiar least squares estimators, (ii) the *Huber* estimator, and (iii) the Turkey *bisquare* (or *biweight*) estimator. Table 2.1 summarizes the objective and weight functions for these three estimators.

Method	Objective function	Weight function
Least squares	$\rho_{LS}(e) = e^2$	$w_{LS}(e) = 1$
Huber	$\rho_H(e) = \begin{cases} \frac{1}{2}e^2 & \text{for } |e| \le k \\ k\,|e| - \frac{1}{2}k^2 & \text{for } |e| > k \end{cases}$	$w_H(e) = \begin{cases} 1 & \text{for } |e| \le k \\ k/\,|e| & \text{for } |e| > k \end{cases}$
Bisquare	$\rho_B(e) = \begin{cases} \frac{k^2}{6}\left\{1 - \left[1 - \left(\frac{e}{k}\right)^2\right]^3\right\} & \text{for } |e| \le k \\ k^2/6 & \text{for } |e| > k \end{cases}$	$w_H(e) = \begin{cases} \left[1 - \left(\frac{e}{k}\right)^2\right]^2 & \text{for } |e| \le k \\ 0 & \text{for } |e| > k \end{cases}$

Table 2.1: Objective and weight functions for least squares, Huber and bisquare estimators (Fox Jan 2002).

From Table 2.1 it is clear that both the least squares and the Huber loss function are increasing functions without any bound as e starts from zero. However the least squares loss function increases faster than the later. In contrast to the least squares and the Huber estimators, the bisquare estimator employs a level-off loss function for $|e| > k$. In terms of the weight functions, the ordinary least squares assigns equal weights to all observations while weights for Huber estimator decline when $|e| > k$. For bisquare, its weights also drop when e departs from zero and become zero when $|e| > k$.

For the Huber and bisquare estimator, k is called the *tuning constant*: A smaller value for k results in more resistance to outliers but in a trade off for lower efficiency when errors are normally distributed. To obtain a high efficiency for the normal case, it is recommended to set k equal to 1.345σ for the Huber and 4.685σ for the bisquare estimator with σ the standard deviation of the errors (Fox Jan 2002). In Figure 2.1, the objective and weight functions of these three estimators are plotted with $k = 1.345$ for Huber and 4.685 for the bisquare estimator. In computer vision, the bisquare function is the most popular choice.

Considering the objective function of M-estimators in equation (2.17), another factor that needs to be considered is the scale parameter s. In statistical theory, the scale parameter is often calculated as the product between $\widehat{\sigma}$, the robust estimate for the standard deviation of errors ϵ, and a tuning constant. However, as the tuning constant k in the Huber and bisquare estimators is derived from the asymptotic properties of the simplest location estimator which
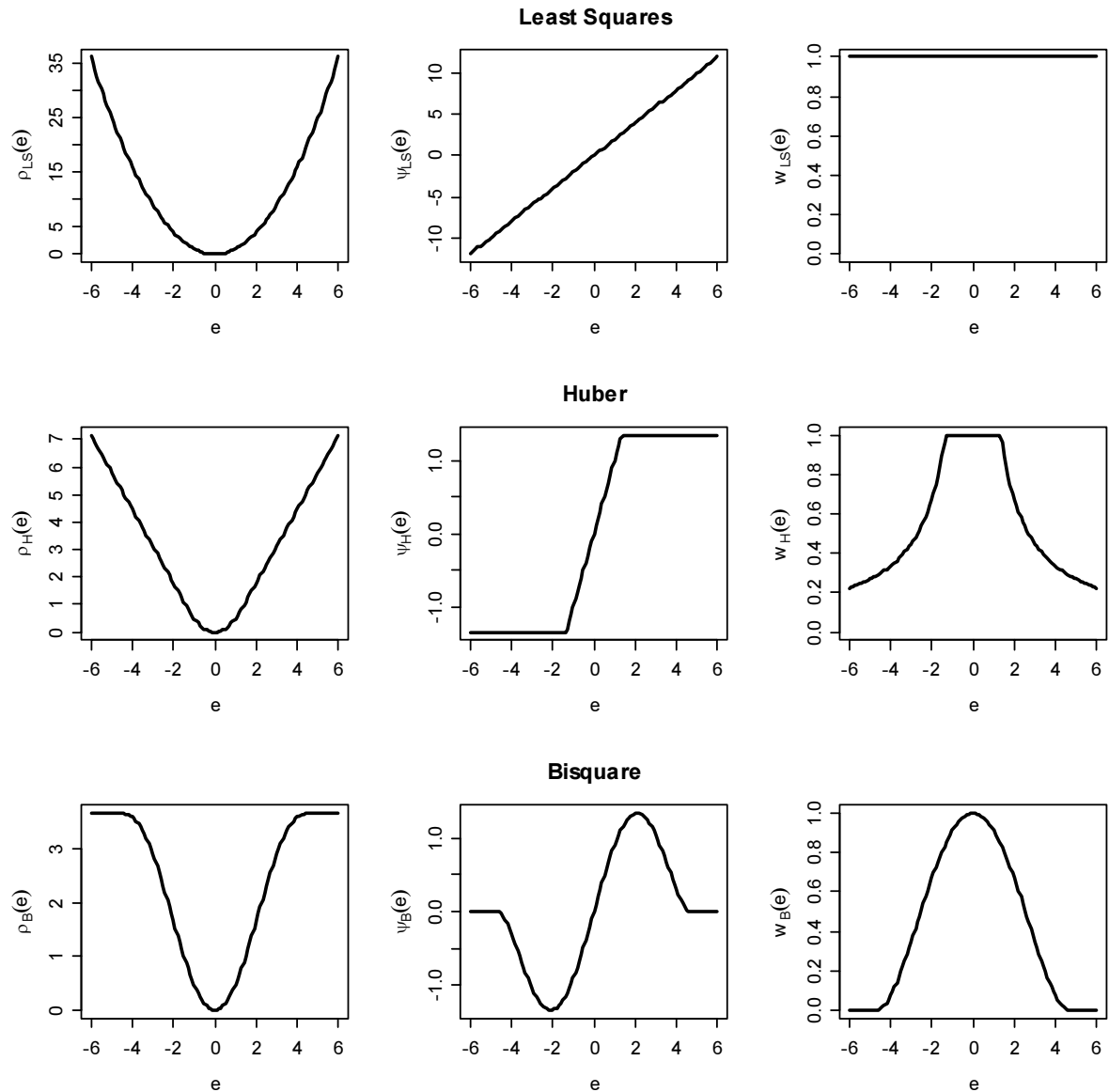
Figure 2.1: Objective (ρ) and weight (w) for least squares (top), Huber (middle) and bisquare (bottom) estimator. The tuning constant for these graphs are $k = 1.345$ for the Huber and $k = 4.685$ for the bisquare estimator (Fox Jan 2002).

is the mean of the population, its value is rarely meaningful. To avoid the problem of tuning constant, *redescending* M-estimators are proposed by MEER (2004).

Redescending M-estimators

Redescending M-estimators avoid the problem of tuning by using the inlier/outlier classification threshold as the scale parameter s. The loss function of redescending M-estimators are bounded:

$$\rho(e) = \begin{cases} 1 - (1 - e^2)^d & \text{for } |e| \leq 1 \\ 1 & \text{for } |e| > 1 \end{cases}$$

where $d = 1, 2, 3$. They are based on the assumption that they have continuous derivatives up to $(d - 1)$-th order. As for the general loss function of M-estimators, also $\rho(0) = 0$ holds for other redescending loss function. When $d = 3$, the redescending M-estimator becomes the bisquare estimator.

The redescending M-estimates only take into account data points with a distance less than s. This solves the tuning issue and yields better outlier rejection properties than any M-estimator with non-redescending loss functions (MARTIN et al. 1989, ZAMAR 1989). In addition, this property means that the redescending M-estimators have a very low breakdown point and their loss functions can be chosen to redescend smoothly to zero. As a result, they do not completely ignore moderately large outliers while completely rejecting gross outliers, thus improving the efficiency. I.e., the redescending M-estimations downweight the influence of data points with a large error. This makes redescending M-estimators slightly more efficient than the Huber estimator. Because of these advantages we choose this model in our study.

Considering equations (2.18) and (2.19) it is obvious that for M-estimators, s plays an essential role in obtaining a better efficiency and less bias compared to the classical estimates in the presence of outliers. The scale s is a strictly monotonically increasing function of the standard deviation (σ) of the errors ϵ. We can estimate σ together with the unknowns at every iteration of the M-estimation process. However, according to MEER (2004), this strategy is less robust than applying a fixed scale value for the main estimation process. The fixed scale strategy mostly applied in computer vision is the *M-estimator with auxiliary scale* in which the scale is either (i) arbitrarily set by the user, or (ii) derived from the data following a pilot estimation procedure. The most frequently used auxiliary scale is the *median absolute deviation* (MAD):

$$\widehat{s}_{mad} = c \times \underset{i}{\text{med}} |\widehat{g}(l_i) - \underset{j}{\text{med}}\, \widehat{g}(l_j)|. \tag{2.22}$$

with c a constant set by the user. If a normal distribution can be assumed for the residuals, c is to be set equal to 1.4826 to obtain consistent estimates for σ. However, in computer vision this assumption is not met as the percentage of outliers is usually high. Therefore, each application needs to have its own way to set the scale parameter to fit its data characteristics. We will discuss our strategy in setting the scale parameter in Chapter 5.

2.2.2   Other Robust Techniques

In this part, we will briefly describe other robust techniques that help us to solve the outlier problem and compare them with M-estimators to show that these techniques are indeed all related to M-estimators with auxiliary scale.

The Least Median of Squares (LMedS)

According to the LMeds method, the objective function in Equation (2.17) is replaced by the following function to solve for the unknowns Δx in our model:

$$\left[\widehat{\Delta x}\right] = \Delta x \text{argmin } i \text{med } g(l_i)^2 \tag{2.23}$$

The LMedS estimator yields the solution with the smallest value for the median of squared residuals computed from all data points. Thus, it finds in the space of the data the narrowest band that contains at least half of the observations (see Figure 2.2(a)). It is therefore very robust to both false matches and outliers due to bad localization. Unlike M-estimators, this estimator cannot be reduced to a weighted least squares problem. I.e., we cannot write a straightforward equation for the estimation of the unknowns. Instead, the whole space of possible estimates that can be generated from the data has to be generated. Due to the often large spaces, one usually only analyzes a randomly chosen subset of the data.

The scale parameter s does not appear in the objective function of LMedS. Particularly, instead of setting up a threshold as in the redescending M-estimators, LMedS imposes a lower bound on the percentage of inliers, namely fifty percent. This elimination of the need to guess the amount of measurement noise results in a somewhat better robustness.

The condition that at least fifty percent of data points must be inside of the band can be related to the scale parameter:

$$\sum_{1}^{n} \rho_{zo}(\frac{1}{s}g(l_i)) = \frac{1}{2} \tag{2.24}$$

with ρ_{zo} the zero-one loss function corresponding to $d = 0$ in the above equation for ρ_e and s a function of the residuals $s[g(l_1), ..., g(l_n)]$. Defining $s = i\text{med}|g(l_i)|$, one obtains the LMedS estimator as:

$$[\widehat{\Delta x}] = \Delta x \text{argmin } s[g(l_i), ..., g(l_n)] \quad \text{subject to } (2.24). \tag{2.25}$$

Denoting the minimum of s in Equation (2.25) as $\widehat{s}$, one gets:

$$[\widehat{\Delta x}] = \Delta x \text{argmin } \sum_{1}^{n} \rho_{zo}(\frac{1}{\widehat{s}}g(l_i)) \tag{2.26}$$

Thus, the LMedS estimator becomes the M-estimator with auxiliary scale. An important shortcoming of the LMedS estimator in computer vision is that when the inliers are no longer the absolute majority in the data, the LMedS fit is incorrect and the residuals used to compute $\widehat{s}$ are not reliable.

Random Sample Consensus – RANSAC

RANSAC was invented before LMedS. Similar to LMedS estimation, it also follows a procedure based on subsets. However, contrary to LMedS in which the scale is calculated from

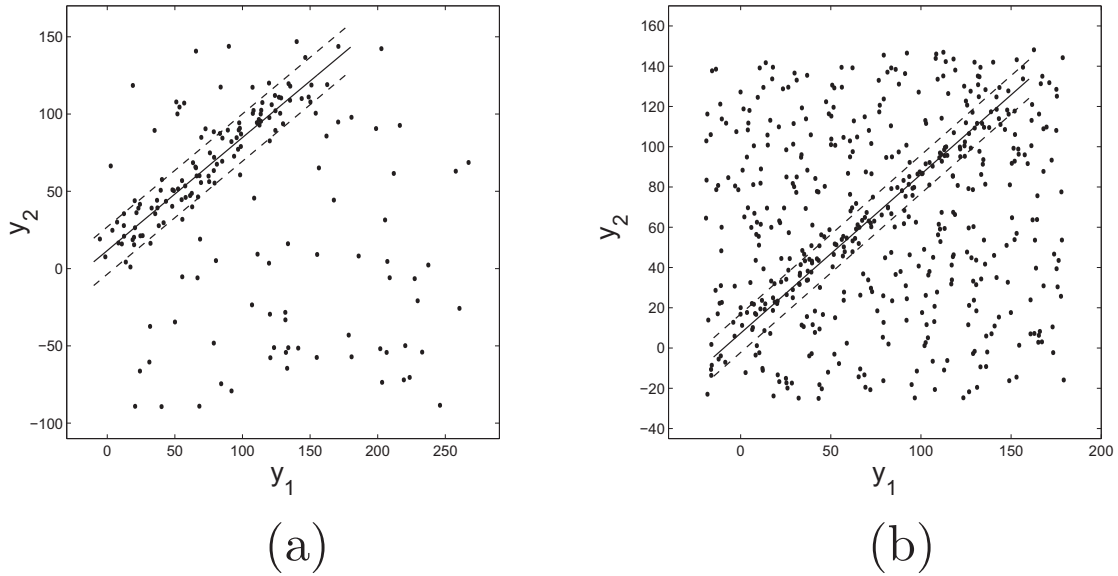(a)   (b)

Figure 2.2: The difference between LMedS and RANSAC. (a) LMedS: finds the location of the narrowest band which contains half the data. (b) RANSAC: finds the location of the densest band with width specified by the user (MEER 2004).

a set which is conditioned by fifty percent of inliers (Equation (2.24)), RANSAC uses a scale which is defined beforehand by the user:

$$[\widehat{\Delta x}] = \Delta x \text{argmin} \ \sum_{1}^{n} \rho_{zo}(\frac{1}{s}g(l_i)) \quad \text{given} \ \ s \tag{2.27}$$

RANSAC is prefered to LMedS in most vision applications as it can handle situations with much more than fifty percent of outliers in which LMedS will necessarily fail as shown in Figure 2.2(b). There are two main problems when applying RANSAC and LMedS in vision applications. First, the use of zero-one loss function in both LMedS and RANSAC yields very poor local robustness properties. The second issue lies on the general problem of handling multistructured data, i.e., data with multiple occurrences of an object.

Hough transform

The Hough transform is not only the earliest robust estimation, but also the only method designed to handle multistructured data. The basic idea is to replace problems in the input domain of the regression with location problems in the space of parameters. The purpose is to find possibly imperfect instances of objects of a certain class of shapes by a voting procedure.

Here, it is only shown how the Hough transform can be linked to M-estimators. For this purpose, we will focus on the *randomized* Hough transform (RHT). Under RHT, the feature space is established by elemental subsets.

The traditional RHT quantizes the parameter space into bins, i.e., it is an accumulator. The bins with the largest number of votes give the parameters of the significant structures in the input domain. The RHT can be formally written as:

$$[\widehat{\Delta x}]_h = \Delta x \operatorname{argmax}_h \sum_1^n \kappa_{zo}(s_\alpha, s_{\beta_1}, ..., s_{\beta_{p-1}}; g(l_i)) \tag{2.28}$$

with $\kappa_{zo}(u) = 1 - \rho_{zo}(u)$ and $s_\alpha, s_{\beta_1}, ..., s_{\beta_{p-1}}$ the size (scale) of a bin along each parameter coordinate and h indicating the different local maxima. The RHT defined in equation (2.28) is a type of redescending M-estimator with auxiliary scale with the criterion maximization rather than minimization.

Even though the Hough transform is widely used in computer vision especially to detect planes and cylinders of 3D objects, it has some limitations in application. First, it is only efficient when a larger number of votes is generated for each object. This requires that the bins must not be too small. Otherwise, some votes will fall in neighboring bins and thus reduce the visibility of the main bin. Second, when there are more parameters (e.g., > 3), on average a low number of votes will fall in a single bin. Finally, the efficiency of the Hough transform depends on the quality of the input data which in most cases is difficult to control in computer vision.

To conclude, following the above analysis of the advantages/disadvantages of different estimation techniques together with our purpose of a full reconstruction of the 3D surface and the texture of a scene, we decided to apply M-estimators with least squares. We will describe our data characteristic in Chapter 4.

Chapter 3

Former Work on Surface Reconstruction

The main issue of this thesis namely surface reconstruction by means of multiview stereo matching has been a topic of intensive research over the last few years and a number of potential algorithms has been developed. To show how our algorithm differs from others, we describe in the first section of this chapter six properties of stereo reconstruction algorithms. In the next section, we discuss the advantages and disadvantages of recent algorithms in 3D reconstruction to motivate our algorithm and to explain our algorithm's properties. In the third section, we describe the approach of HEIPKE (1990) on which we have based the development of our reconstruction technique, but extended towards fully 3D surfaces. We explain the strengths and weaknesses of Heipe's approach and briefly discuss how we overcome the weaknesses in our algorithm.

3.1 A Multi-view Stereo Taxonomy

In this section, we follow SEITZ et al. (2006) to classify current multi-view stereo reconstruction algorithms according to six main properties: (1) Scene representation, (2) photoconsistency measurement, (3) visibility model, (4) shape prior, (5) reconstruction algorithm, and (6) initialization requirements. These six properties will help us to discuss different stereo algorithms.

3.1.1 Scene Representation

A scene representation is defined here in terms of the geometry of an object or a scene. Choosing an appropriate representation is very important for scene reconstruction. Particularly, there are some criteria that a representation should meet (FUA and LECLERC 1995): First, it should be possible to represent any continuous surface, closed or open and of arbitrary genus. Second, the representation should allow to generate a surface from a standard data set such as a set of points. Third, it should be possible to integrate information from multiple images.

A lot of multi-view algorithms use voxels, level-sets, polygon meshes, or depth maps to present the geometry of a scene. While some algorithms use a single representation, others employ different representations for various steps in the reconstruction.

Voxels and level-sets (Seitz and Dyer 1997, Faugeras and Keriven 1998) are both based on a discretization of 3D space, i.e., 3D grid representations. They are different in term of the definition of the grid function. This function tells whether a grid cell is a valid point of the scene or not. In the voxel representation, the scene is represented by a discrete function defined for every grid cell. The function is positive when a grid cell is a valid point of the scene (colored gray in the left image of Figure 3.1), otherwise it is negative (colored white).

For level-sets, the grid function is based on the distance to the closest surface. The function imposes a negative value for all grid cells which lie inside the object (indicated by the light gray colored cells in the right part of Figure 3.1), a positive value for grid cells which are outside the object (darker colored cells), and zero for the scene points (medium gray colored cells).

The third representation are polygon meshes, e.g., Fua and Leclerc (1994). Surfaces are represented as sets of connected planar facets or triangular meshes (see Figure 3.2). Because polygon meshes are efficient in storage and rendering, they are a popular representation in multi-view algorithms. They are also very suitable for visibility computations.

Multi-depth map representation has been applied by several methods, among them by Strecha (2007), Kolmogorov and Zabih (2002), as well as Gargallo and Sturm (2005). The depth maps are defined as depth values attached to each pixel of the images. This type of representation avoids resampling the geometry in a 3D domain (see Figure 3.3).

3.1.2   Photo-consistency Measure

To evaluate the visual compatibility of a part of the scene when projected into different images, one needs to define a measure. The type of measure can be changed and does not depend on a specific algorithm for scene reconstruction. I.e., one can take a measure from one strategy and apply it in another strategy. Most of the current reconstruction algorithms use photo-consistency measures which compare pixel intensities in one image to those in another image to check how well they correlate. According to Seitz et al. (2006), there are two main types of photo-consistency measures: Scene space and image space measures. This classification is related to the scene representation, because it is based on the integration method.

For the scene space measures, points, but mostly patches or volumes are projected into the input images and then the similarity (agreement) between those projections is evaluated. Different functions have been applied to measure the agreement. While Seitz and Dyer (1997) use the variance of the projected pixels in the input images, Faugeras and Keriven (1998) exploit the sum of squared differences or normalized cross correlation. These measures are usually integrated over a surface, thus are more suitable for smaller surfaces.
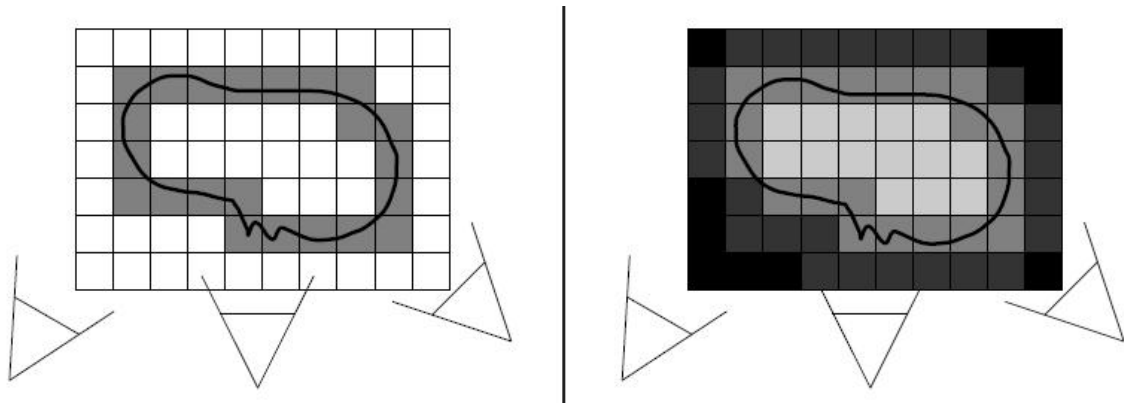
Figure 3.1: Scene representation by voxels (left) and level-sets (right) (STRECHA 2007).

For image space measures, the estimated scene geometry is used to warp the input image and thus to compute forecasted images eliminating projective distortion. The prediction error, i.e., the difference between the predicted and measured images, is the image space measure in PONS et al. (2005). While this is very similar to the scene space measures, the difference lies in the domain of integration. Unlike scene space measures, image space measures are integrated over the set of images and thus more weight is attached to scene areas that are frequently observed.

Lambertian reflection is the common assumption of almost all traditional stereo algorithms. Under this assumption, there is a constant brightness of conjugate pixels in different images. The use of cross correlation weakens this assumption, because it allows for linear brightness and constrast changes and it is robust in the presence of highlights. Currently, there are some algorithms that relax this assumption. Some methods also consider silhouettes or shadows.

3.1.3 Visibility Model

To evaluate the photo-consistency, algorithms also need to specify the views that need to be considered, i.e., the visibility. The visibility model is introduced as there may be occlusions and often only very few scene elements or even none are visible from all cameras. The visibility model helps to obtain a high-quality reconstructed shape which is close to the real shape. According to SEITZ et al. (2006), there are three different techniques to handle visibility: (i) Geometric, (ii) quasi-geometric, and (iii) outlier-based methods.

For the geometric approach, the authors try to determine the visibility of scene structures, i.e., sets of images where they can be seen. The simplest way is to use the current estimate of the geometry of a surface to predict the visibility for each point on this surface. FAUGERAS and KERIVEN (1998) among others apply this. According to SEITZ and DYER (1997), the visibility can also be simply computed by setting constraints on the positions of cameras. The geometric approach includes two step estimations: Geometry and visibility estimation.

The quasi-geometric method differs from the geometric approach by using approximate geometric reasoning instead of the estimated geometry. One popular technique is to limit photo-

Figure 3.2: Wire-frame representation of a mesh (black lines) connected from predefined 3D points (black points).

consistency analysis to a group of nearby cameras (KUTULAKOS and SEITZ 1999). This not only reduces occlusion effects, but also excludes oblique views and improves the calculation speed. Another popular quasi-geometric technique is to use the visual hull which is a rough estimate of the surfaces to predict the visibility. Again, this method is iterative with a two step estimation.
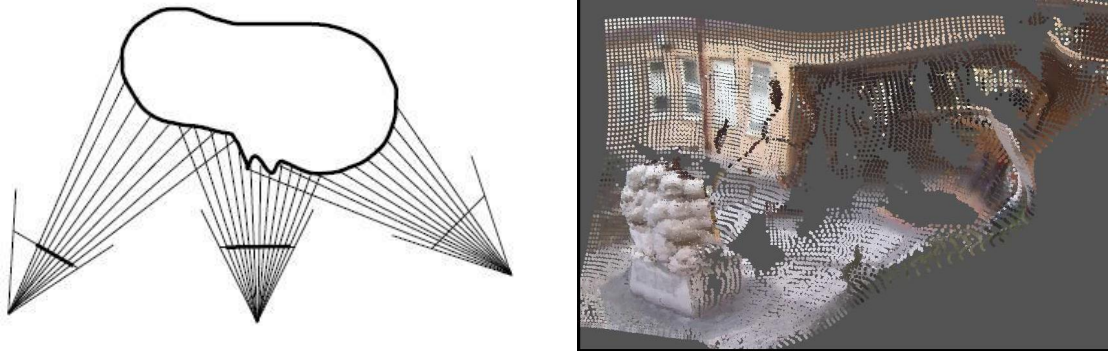
Figure 3.3: Scene representation by depth map (Strecha 2007) (left) and a 3D point rendering of the set of all depth maps (Gargallo and Sturm 2005) (right).

The third method considers occlusions as outliers (Kutulakos and Seitz 1999) and ignores geometric reasoning. This is useful when points are visible in more images than they are occluded. Thus, rejection of outliers provides the views where points are visible.

3.1.4 Shape Prior

In an un-textured or weakly textured region, the image data usually convey insufficient information. Thus, it is very difficult to obtain correct matches across images. Therefore, to recover the scene geometry in such regions, it is important to use a shape prior imposing a desired shape characteristic on the reconstructed scenes, biasing it towards a plausible solution. Priors are less important in multi-view stereo where multi-view constraints can be employed.

There are several different types of shape priors. For techniques that employ a scene space consistency measure, their priors are minimal surfaces with small overall surface regions. Such priors normally result in a smoothing particularly for points of high curvature. For mesh representation algorithms, the minimal surface priors try to shrink triangles or prefer spheres or planar shapes (Fua and Leclerc 1994).

In contrast, methods that apply voxel representations mostly use maximum surface priors: Voxels are removed only when they are not photo-consistent (Seitz and Dyer 1997). Therefore, they produce the largest possible photo-consistent scene reconstruction. Maximum surface priors are helpful for high curvatures or thin structures, because there is no assumption on the smoothness of the surfaces.

Instead of having a global prior for the whole surface, some methods impose local smoothness priors. E.g., approaches give neighboring image pixels the same or a similar depth by applying a smoothness term additionally to the image space consistency measure. This type of prior has the disadvantage that it often has a bias toward fronto-parallel surfaces. However, this bias can be eliminated by imposing surface based priors (Fua and Leclerc 1994).

3.1.5   Reconstruction Algorithm

According to SEITZ et al. (2006), reconstruction algorithms are grouped into four categories. SEITZ and DYER (1997) follow the first category in which a cost function on a 3D volume is computed and then surfaces are extracted from that volume. In this category algorithms differ in terms of the applied cost function and the way surfaces are extracted.

In the second class of reconstruction algorithms, surfaces are iteratively extracted through minimizing a cost function. This is employed for all types of representation: Voxels, level-sets, and polygon meshes. Following the voxel representation, e.g., space carving (KUTULAKOS and SEITZ 1999) eliminates inconsistent voxels from an initial volume while other techniques such as YANG et al. (2003) not only remove but also add voxels to minimize cost functions. Level-set methods (FAUGERAS and KERIVEN 1998), on the other hand, try to minimize their cost function based on a set of partial differential equations on a volume. They often start from a large initial volume and then shrink inward. When needed, they can also expand the initial volume to minimize the cost function. Under the mesh representation (FUA and LECLERC 1994) the scene is described as an evolving mesh, which can move and deform as a function of internal and external parameters.

The third class consists of methods that employ image space consistency measures to calculate a set of depth maps. In order to obtain a single consistent 3D scene representation, these techniques impose consistency constraints between depth maps (KOLMOGOROV and ZABIH 2002), or fuse the set of depth maps into a 3D scene.

For the fourth category of algorithms, first a set of feature points is extracted and matched, and then a surface is fitted to the reconstructed points.

3.1.6   Initialization Requirements

In general, images and the internal as well as external calibration parameters, i.e., projection matrices, are the input for a multi-view stereo algorithm. For most algorithms, additional information is required to initialize the reconstruction or to narrow the geometric extent of the scene to be recovered. When and which additional information is required depends on the specific algorithm. Most algorithms require a rough bounding box of the object. For voxel and level-set representations one needs to define the voxel grid while for mesh representation, the initial set up of the mesh is required. To reconstruct the visual hull based on silhouettes, several algorithms require foreground/background information. Depth map based representations, on the other hand, require information about the depth range, which can be calculated by projecting the bounding box into the images. For methods that follow partial differential equations, a set of initial 3D points is required. We will discuss the input information for our algorithm in Chapter 4.

Of the six properties, the scene representation is the most important, while other properties

can be changed and do not depend on a specific representation. I.e., choosing an appropriate representation is the main task. Because of this, we discuss in the next section several current algorithms which are typical examples of the three representations voxels, level-sets, and meshes.

3.2 Current Algorithms with Different Scene Representation

3.2.1 Reconstruction with Voxel Representation

The algorithms of SEITZ and DYER (1997) and KUTULAKOS and SEITZ (1999) both apply a voxel representation. While SEITZ and DYER (1997) use the color of the voxels in their reconstruction, KUTULAKOS and SEITZ (1999) apply space carving. In this subsection, we describe these two strategies including their advantages and disadvantages.

Voxel Coloring

SEITZ and DYER (1997) propose to use voxels in conjunction with their color to reconstruct a photorealistic scene with the goal that it is indistinguishable from what one can observe from the same view point. To preserve the photorealistic characteristic, a reconstruction algorithm needs to meet two criteria: (i) Color, texture, and pixel resolution need to be preserved, and (ii) it should be possible to integrate numerous and widely-distributed input images. In terms of these criteria, the method takes into account occlusions to obtain a consistent reconstruction. This could not be guaranteed by earlier algorithms.

The method avoids the image correspondence problem owing to its discretized scene space with voxels (points) traversed in a fixed order according to visibility. The 3D surfaces are defined implicitly as finite sets of voxels and the 3D space is partitioned into a series of voxel layers which increase with the distance from the camera. All voxels in a layer have the same volume and each voxel is assigned a unique color (radiance). The algorithm assumes that both scene and lighting are stationary and surfaces are (approximately) Lambertian. Under these assumptions, the radiance at each point is isotropic and can be described by a scalar value (color).

The reconstruction starts with a dense map, which is built by assigning colors to voxels in a 3D volume to achieve consistency with a set of basis images, as shown in Figure 3.4. It is the input information for calculating voxels belonging to the surface. To obtain a consistent reconstruction, first the color invariance constraint is enforced according to which voxels are colored by a unique color obtained from all images where a voxel is visible. The second key element to solve the correspondence problem is the ordinal visibility constraint which is a constraint on the configuration of camera viewpoints. It means that occluding voxels are processed before those they occlude. Thus, the algorithm turns from the determination of
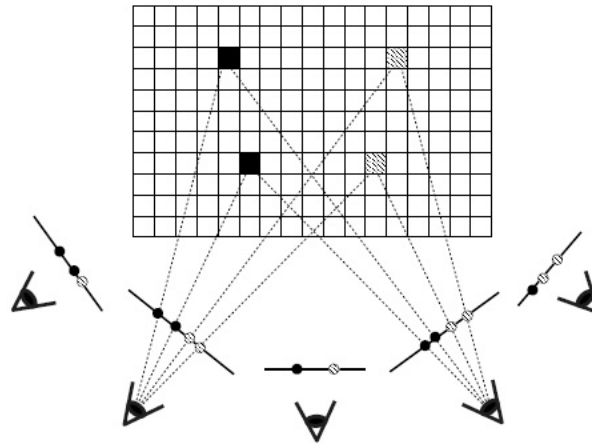
Figure 3.4: Given a set of basis images and a grid of voxels, voxel coloring aims to assign color values to voxels in a way that is consistent with all of the images (SEITZ and DYER 1997).

the surface to checking whether a voxel of the surface has an invariant color in all images in which the voxel can be observed. As a result, the strategy of voxel coloring is to assign colors to sets of voxels until no voxel with an invariant color is found. Owing to the above two elements, the strategy is suitable for reconstructing a consistent dense correspondence map from a set of input images under the presence of occlusions.

On one hand, this method has several advantages: First, occlusions are explicitly modeled and accounted for. Second, the cameras can be positioned far apart without degradation of accuracy or runtime. Third, the method can integrate numerous images to yield dense reconstructions that are accurate over a wide range of target viewpoints. On the other hand, the method also has some disadvantages: First, a major weakness is the lack of means to impose spatial coherence. This is problematic, because image data are almost always ambiguous. Second, the method requires a precise camera calibration, thus, it is sensitive to high-frequency regions such as image edges. In addition, to be able to represent high-frequency image content, one needs a higher voxel sampling rate increasing the runtime. Third, different voxel resolutions can lead to a varying reconstruction quality. By increasing the sampling rate, the reconstruction quality can be improved to the limit given by image quantization and calibration accuracy at the cost of an increased runtime. Fourth, the assumed Lambertian reflection model is not well suited to reconstruct practical specular surfaces. Finally, this method only reconstructs one of the possibly numerous scenes that are consistent with the input images. I.e., it is sensitive to the aperture problem which occurs in regions of near-uniform color.

Space Carving

Like SEITZ and DYER (1997), also KUTULAKOS and SEITZ (1999) apply a voxel representation, but in a different way. Their purpose is again to reconstruct photorealistic shapes from multiple photographs (*n* input photographs) taken from known, but arbitrarily-distributed view points. The method takes into account the complex interactions between occlusion, parallax,

shading, and view-dependent effects on the appearance of scenes. Specifically, the goal is to reconstruct from n-views scenes without any constraint on a scene's shape or the viewpoints of the input photographs.

The main idea of the method is to make use of the assumption that the scene radiance belongs to a general class of radiance functions called locally computable. I.e., the radiance at any point is assumed to be independent of the radiance at all other points in the scene. Owing to this, the scenes which can be analyzed can be narrowed to those where shadows, transparency, and inter-reflection effects can be neglected. Once a local computable radiance function is known, it can be determined easily whether a specific shape is photo-consistent with a set of photographs.

By applying the locally computable function, a maximum shape called the photo hull can be computed from n photographs of an unknown scene. The photo hull contains by definition the set of all photo-consistent reconstructions and subsumes all other members of this photo hull. For the photo hull computation, besides the assumption of a locally computable radiance function, viewpoints of each photograph have to be known in a 3D world reference frame. The main idea is that by carving space in a well-defined manner, one can compute the photo hull.

The algorithm has two constraints: The first is the visual hull constraint which requires that no point of the scene can project to a background pixel, i.e., the scene is restricted to the visual hull. The visual hull concept comes from LAURENTINI (1994). It is defined as the volume of intersection of the cones corresponding to n photographs. The second is the radiance constraint. It states that a non-transparent and a non-mirror-like surface reflects light in a coherent manner. According to the visual hull constraint, the visual hull constrains the shape in the input photographs when no prior information about the scene's radiance is available. The visual hull constraint is only useful to separate scene from background pixels. For the non-background pixels, the second constraint is needed providing a special class of functions for the scene radiance.

Given an initial volume which contains the scene, the method iteratively removes (carves) parts (voxels) until the volume converges to the photo hull or until no non-photo consistent voxel can be found on the carved surface any more. The algorithm also takes into account visibility updating it for all input cameras. Like SEITZ and DYER (1997), KUTULAKOS and SEITZ (1999) evaluate voxels in order of visibility: The occluders are visited before the occluded voxel. The algorithm also considers the multi-view visibility order for cases when the convex hull constraint is violated by evaluating voxels in the order of increasing distance to the camera hull.

Because of the updating of the visibility, the method is time-consuming and memory-intensive. Therefore, multi-sweep space carving is used which ensures that all cameras are considered. Multi-sweep carving performs six sweeps through the volume, corresponding to increasing and decreasing directions for three dimensions. To make sure that all cameras which are visible for a voxel are considered, voxel consistency checks are employed which combine the voxel visibility information obtained from individual sweeps.

Space carving has four advantages: First, it can reconstruct all possible shapes of a scene without requiring any constraints on the geometry, topology, or camera configuration which is a limitation of the voxel coloring method. Second, the method allows the reconstruction of non-smooth, free-form shapes from arbitrarily positioned and oriented cameras. Third, the usage of photo-consistency criteria guarantees faithful image projections and helps to solve the complex interactions between occlusions and parallax as well as shading effects. However, the method also has some weaknesses: First, while it is proven to be effective under low image noise conditions, in reality there are image noise and quantization as well as calibration errors which can severely affect the photo hull. Second, the method is restricted to locally computable radiance functions and ignores non-locally computable functions. As a result, it cannot handle shadows and (inter-)reflections. Third, it has a high runtime and is complex to implement. Fourth, like voxel coloring, it also doesn't have a way to impose spatial coherence.

3.2.2   Reconstruction with Level-set Representation

Faugeras and Keriven (1998) employ the level-set representation to solve the stereo problem for an arbitrary number of images. Their strategy is based on the variational principle. This makes it possible to incorporate hypotheses about objects in the scene.

The main idea is that Euler-Lagrange equations are deducted by means of the variational principle resulting in a set of partial differential equations (PDEs). The PDEs are used to deform an initial set of surfaces so that they move toward the objects in the scene. The approach has two main assumptions: First, objects imaged by a binocular stereo system are modeled as an unknown smooth function $z = f(x, y)$. Second, as in the voxel coloring technique, perfect Lambertian reflection is assumed.

According to Faugeras and Keriven (1998), the reconstruction is solved as follows: (i) Basically, they tackle the problem of surface reconstruction from multiple images by minimizing functions that describe the geometry of the scene. (ii) They then compute the Euler-Lagrange equations of these functions to obtain a set of conditions which lead to PDEs. (iii) Finally, they solve the PDEs via time evolution by a level-set method.

The function considered in this approach is expressed as a weighted minimal surface evolving over time and is given by $\int \int w(S(t))ds$. The time-evolving surface $S(t)$ is represented at time t by the zero level-set of an implicit function $\phi(S(t), t) = 0$. The function has a negative value for every grid cell inside the object (indicated by light gray cells in the top/right image of Figure 3.1), positive values for outside cells (dark cells), and is zero for grid cells which are scene points. Faugeras and Keriven (1998) employ a measure of correlation under the hypothesis, that the scene consists of fronto parallel planes. By considering the orientation of the tangent plane to the surface of the object they can relax the above constraint.

The main advantages of this approach are that it provides an efficient and robust way for reconstruction and it can deal with multiple objects. Like the voxel coloring method it also works in the presence of occlusions. The main limitations are that it is time-consuming

and difficult to control the topology. In addition, the Lambertian assumption is unrealistic practice.

Many techniques have applied 3D grids to represent geometry. The voxel and level-set representation are popular owing to their simplicity, uniformity, and ability to approximate any surface using a large number of images which are taken from arbitrary viewpoints. However, the initial discretization determines their resolution and the precision is limited by the resolution of the volume grid. The only way to achieve a better resolution is to increase its size. By this way, the resolution can be adapted to a detailed shape, leading to the best reconstruction. However, besides the inherent time-complexity this can also lead to problems when dealing with self-intersections and topological changes.

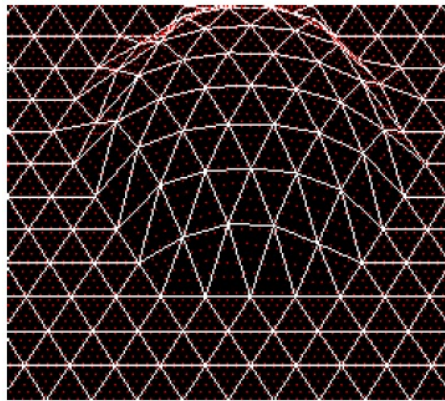### 3.2.3   Reconstruction with Triangulated Meshes

Figure 3.5: A wire-frame representation of the mesh (bold white lines) (Fua and Leclerc 1994).

In this section, two typical methods that apply triangulated meshes are discussed. One of the most interesting approaches using a triangulated mesh representation for 3D shape reconstruction is (Fua and Leclerc 1994). The surface is created in the form of a triangulated grid, in which each vertex corresponds to a Cartesian coordinate point (see Figure 3.5). Each vertex except those at the edges has six neighbors. The strategy combines both image-based and geometry-based information. The geometry-based information takes into account 3D points as attractors, 3D linear features, and 2D silhouettes. The image-based information considers stereo and shape-from-shading cues. Owing to the possibility to make use of different sources of information, the strategy allows to reconstruct complicated surfaces for which strategies that consider only one source cannot provide a unique solution. The employed image and geometric information can stem from many images possibly taken from very different viewpoints allowing to deal with self-occlusion and self-shadowing issues.

The main idea of the approach is to recover at the same time both the shape and the reflectance properties of a surface by deforming a generic object-centered 3D representation of the surface and then minimizing an objective function. Particularly, the objective function is a weighted sum of the contributions of the image-based and geometry-based information.

Variables of the objective function are the coordinates of the vertices of the mesh which represent the surface. The surface is first initially estimated and then optimized by changing the position of the vertices in the grid. The iteration stops when a local minimum is reached, i.e., the triangle grid best fits the real surface. At each iteration, the weights in the objective function are changed according to the current geometry of the surface and the degree of texturing within the area of the triangles. Because the magnitudes of the image and geometry information are not commensurate, the weights attached to each type of information need to be normalized.

To minimize the objective function, FUA and LECLERC (1994) add a regularization term and progressively reduce the influence of this term. The regularization is defined as a quadratic function of the vertex coordinates. This smoothes the energy landscape when its weight is large, improves the convergence of the optimization procedure and avoids overfitting and wrinkling. To speed up the computation and to avoid getting stuck in undesirable local minima, a coarse to fine scheme is employed using different triangle sizes. The scheme starts with large triangles and refines them by splitting them into four smaller triangles each.

The energy function which describes the geometry of the surface is calculated in the form of the curvature from the deviation of each vertex from the neighboring vertices. Thus, the obtained surface retains its smoothness and the approach can deal better with noise. Concerning the image information, the energy function employs both stereo and shape from shading. In each triangle, observations are created by sampling. They have a 3D position and are projected into each visible image. Each projection creates an image value used to build the energy function. The latter is calculated as the sum of the variances of the grey values of the observations. For shape from shading, the energy function minimizes the variation in albedo across the surface.

The approach has advantages not only over the approaches discussed above in Sections 3.2.1 and 3.2.2, but also over other 3D object-centered approaches. First, it avoids the need to rely on previously computed 3D data such as the coordinates of points derived from laser range finders. Second, the mesh representation allows to easily incorporate image-based and geometry-based sources of information in the reconstruction process. Therefore, all available information is made use of and helps to recover complicated surfaces. This kind of reconstruction is difficult with other approaches in which only one source of information is considered. Additionally, not only the surface of the object is recovered, but also the reflectance properties. Finally, it allows to select the sources of information to be used and to change their relative importance.

The approach, however, still has some disadvantages. First, like approaches which employ 3D object-centered representations, it assumes that the range data which is used to initialize the surfaces can be easily classified into separate groups that define specific objects. As a result, a separate 3D model can be fitted to each object. Obviously, this is a strong assumption for complex scenes because real scenes often contain several objects with the topology unknown in advance. Second, the approach requires manual collection of geometric information for difficult scenes and thus the process is time-consuming and not suitable for highly complex topologies. Third, according to FUA and LECLERC (1994), the recovered surface de-

pends on: (i) How the influence of different terms in the objective function is set, (ii) how they are relatively weighted, (iii) how the smoothing function is modeled, and (iv) on the initial guess (convergence).

In the next paragraph, we describe the approach of Fua (1997) which solves the first and main problem of the approach of Fua and Leclerc (1994).

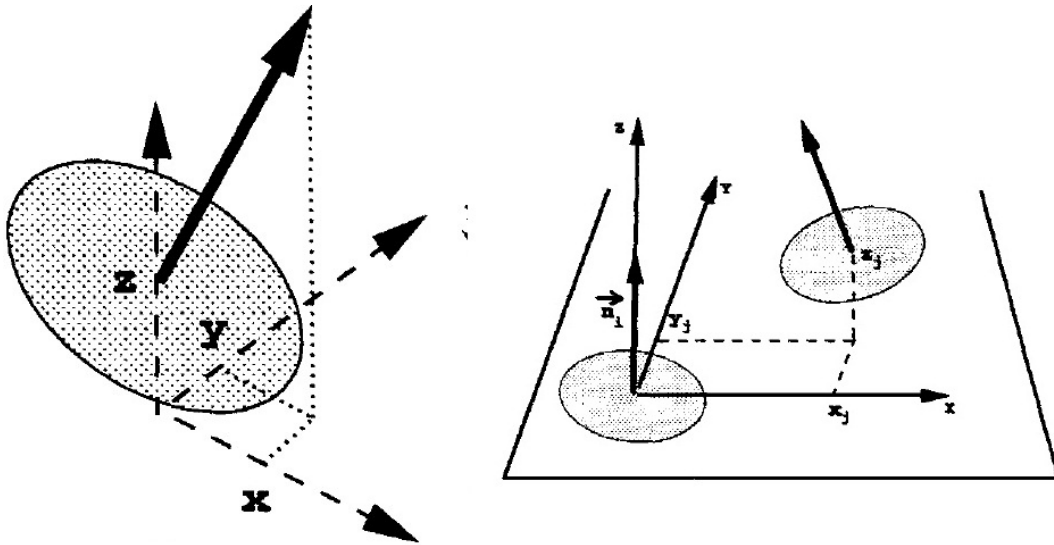### 3.2.4   Reconstruction of Local Surfaces with Particle System

Figure 3.6: Left: A particle is a disk. Right: The distance between two particles is primarily a function of the distance of the center of gravity of one particle from the tangent plane of another (Fua 1997).

To solve the first problem in the approach of Fua and Leclerc (1994) namely the classification of the surface in separate groups, Fua (1997) proposes to replace the triangulated meshes by a particle system. A particle is defined by Fua (1997) as a small planar disk described by the position of its center, the normal vector of the disk, and the radius (see Figure 3.6). The approach employs a metric which helps to effectively cluster local surface patches into global ones. I.e., the approach tries to recover complex surfaces by modeling them as sets of local surface elements who interact with one another. The approach thus does not require a priori knowledge of the surface's topology. It is assumed that the camera models are known in advance. To minimize the objective function of the image-based constraint and the regularization term, the particles are adjusted.

Following this approach, the reconstruction consists of three steps: Initialization, connection, and refinement. The initialization uses a given disparity map as input and computes 3D points. The volume covering the 3D points is split into voxels. For each voxel containing at least one 3D point, an average surface, i.e., particle, is created. The center position of the particle is derived as the center of the voxel projected on the average surface. In the connection step, the particles are linked and surfaces are created. The condition for the connection of two particles is that the distance between their centers is smaller than a threshold.

The result are local surfaces represented by lists of particles. The refinement step adjusts particles based on their projection into images resulting in ellipses. The objective function consists of the difference between the gray levels of pixels in different images compared to the average image. Before optimization outliers are eliminated and only then the positions and orientations of the particles are refined. Finally, a global surface is created.

Even though the approach can reconstruct surfaces of previously unknown topology, it still has some weaknesses: First, it is only based on image information and not on geometrical information when recovering the surfaces. Thus, there is a trade-off between the benefits and costs when following this approach. Second, the technique applied to cluster particles for both optimizing the set and rejecting outliers is too simple for complex configurations.

3.2.5 Semiglobal Matching

HIRSCHMÜLLER (2008) uses an image-based pixelwise representation together with Semiglobal Matching (SGM) to create a very dense reconstruction. The approach applies a Mutual Information (MI)-based matching cost to take care of possibly non-linear radiometric differences of the input images. The approach consists of five steps of which the last two are optional. The steps are: (1) Pixelwise matching cost calculation, (2) cost aggregation, (3) disparity computation, (4) multibaseline matching, and (5) disparity refinement.

For the first step it is assumed that the epipolar geometry is known. It is argued that the assumption of a constant disparity in the vicinity of an image pixel which is commonly, but often implicitly made, e.g., by approaches based on cross correlation, does often not hold in reality. As a result, the approach calculates the matching cost for individual pixels from their intensity I_{bp} and the correspondence I_{mq} in the matching image. By applying MI, the matching becomes insensitive to recording and illumination changes making use of the information in each image and their joint entropy. Matching cost calculation proceeds as follows: First, the match image I_m is warped according to an initial disparity image D. Second, the probability distribution of corresponding intensities is calculated by counting the number of pixels of all combinations of intensities, divided by the number of all correspondence. The probability distribution is calculated only for corresponding intensities to consider occlusions. Third, Gaussian smoothing is applied to account for noise. Fourth, based on the probability distribution, entropies of the two images are calculated analogous to the joint entropy. Fifth, the iteration is started with a random disparity image, matching costs for both images are calculated, and a new improved disparity image is derived. This new disparity image then serves as the base for the next iteration.

To deal with noise, cost aggregation is supplemented by a constraint for smoothness. Thus, the employed energy function $E(D)$ consists of the pixelwise cost and the smoothness constraint and the objective is to minimize $E(D)$. To avoid the streaking problem when combining the 1D optimizations of individual image rows obtained based on dynamic programing to the 2D disparity map, 1D matching costs in all directions are aggregated. Particularly, the approach sums the cost of all 1D minimum cost paths $L_r(p, d)$ in all directions r that end

in pixel p at disparity d to obtain the aggregated cost $S(p, d)$. To obtain a good coverage, between 8 and 16 directions are used.

The disparity image D_b corresponding to the base image I_b is calculated by minimizing the aggregated cost $S(p, d)$. Quadratic interpolation is applied for subpixel estimation. Similarly, the disparity image D_m for the match image I_m can be determined by minimizing the aggregated cost traversing the epipolar line corresponding to the pixel q of I_m. By computing also D_m the consistency can be checked and thus care of occlusions and false matches can be taken in the calculations of D_b and D_m. Particularly, each disparity of D_b is compared to its corresponding disparity of D_m imposing uniqueness by allowing only one to one mappings.

For multibaseline matching, in the fourth step matching costs between a base image and several match images are calculated. To account for occlusions, multibaseline matching is performed pairwise consecutively between the base image and each match image. The consistency check is applied only after pairwise matching to avoid wrong matches at occlusions. The resulting disparity images are combined taking into account the individual scale. The scales are linearly proportional to the length of the baseline between the base image and the match image if all images are projected onto a common plane that has the same distance to all optical centers. Finally, the disparity values are fused as the weighted means of the disparities with the scales used as weights. This helps to discard outliers as only disparities that lie within certain bounds are considered.

Finally, the disparity image is post processed to recover invalid or erroneous values. First, there may be outliers or small patches deviating severely from the surrounding disparity due to low texture, reflections, or noise. To remove these patches, the disparity image is segmented allowing neighboring disparities within one group to vary by one pixel. Disparities of all segments below a certain size are set invalid. Second, in the case of a structured indoor environment, foreground objects can be in front of a low or untextured background. In this case, there is no difference in the energy function if a disparity step is placed correctly next to the foreground object or a bit further away within the untextured background. To account for this problem, the matching cost is adjusted according to the intensity gradient. However, only a meaningful result can be obtained if untextured areas can be identified. For the latter, three assumptions are made, of which the third is rather weak. The approach taken is similar to others which apply image segmentation and plane fitting to polish an initial disparity. However, as the approach only modifies untextured areas above a certain size, the disparity image becomes more accurate and the approach less time consuming. In some cases, the above steps can introduce errors into the disparity image. These are particularly holes which need to be filled by interpolation for a dense result. Holes resulting from occlusions and mismatches must be interpolated differently. While for occlusions the background is extrapolated into the occluded regions, for mismatches holes are smoothly interpolated from all neighboring pixels.

For huge images, memory and time consumption are reduced by dividing the base images into tiles. The disparities for each tile are first computed separately, and then all tiles are merged together into the full disparity image before the multibaseline fusion. The tiles are divided overlappingly, to produce correct results in the cost aggregation step near the tile

borders. When merging the tiles, a weighted mean of the disparities from all tiles in the overlapping areas is computed. Weights are determined in terms of the distance between a pixel and the tile border. I.e., pixels near the tile border are practically ignored while those further away are blended linearly. The tile size is chosen based on the memory capacity. Besides huge images, fusion is also used when several disparity images are taken from different viewpoints for a 2.5D reconstruction. In this case, the information of all disparity images is fused into one consistent representation of the scene by orthographic projection onto a common plane.

SGM is one of the top-ranked stereo matching algorithms by the Middlebury Stereo Vision Page owing to it capabilities concerning occlusions, outliers, mismatches, and huge images (SCHARSTEIN and BLASIAK 2011). The interpolation method for solving occlusions is advantageous compared to other methods as it is independent of the applied stereo matching method. In addition, for huge images, the tiling allows to deal with very large images as every tile can be computed on a different core or computer which saves a lot of time in comparison to other approaches. However, the approach is limited to 2.5D models at the moment. Moreover, in order to obtain a precise and dense result, it needs to be determined whether holes in the disparity image are from occlusions or mismatches. For this, the disparity images for the base and match image need to be known which both again may contain errors.

3.2.6   Discussion of Current Algorithms

The above discussion about the advantages and disadvantages of recent algorithms shows that there is no dominant solution. For each algorithm, there are benefits as well as drawbacks and costs that need to be taken into account. However, the analysis shows that the triangulated mesh representation seems to have advantages compared to other representations for complex scenes, because it allows to combine different sources of information for the reconstruction. Moreover, the usage of a regularization term and the smoothing procedure help to gradually adjust the initially reconstructed surface to the real surface. This does not hold for the voxel representation methods. Additionally, the triangulated mesh representation is also more suitable for the objects we aim at and which will be described in the next section. Therefore, in this thesis we use the triangulated meshes of FUA and LECLERC (1994) as the basis of our algorithm. Our stereo matching algorithm which will be described in detail in Chapter 4 uses an image-based measure to evaluate the compatibility of the reconstruction. Our reconstruction algorithm is based on the minimization of partial differential equations with a minimal shape prior.

We apply a least squares estimator (cf. Section 2.1). Particularly, we use the technique of HEIPKE (1990) which is a successful stereo matching approach using least squares matching in 2.5D as another basis for our algorithm for full 3D reconstruction. Therefore, in the next section we briefly discuss (HEIPKE 1990).

3.3   2.5D Least Squares Matching

HEIPKE (1990) uses least squares optimization to adjust the geometry of the object surface in order to minimize the differences between the brightnesses of the individual images projected on the surface and their average. It is proven to be successful for 2.5D surface reconstruction. In the next paragraphs, we summarize Heipke's technique for 2.5D reconstruction using stereo.

HEIPKE (1990) defines the object surface as a 2.5D square grid. Every point is defined by its 2D coordinates X and Y on the grid and its height Z. The value Z of points inside a grid mesh is computed by bilinear interpolation from the four Z values of the grid points of the respective mesh. To deal with weak approximate values in conjunction with rough surfaces, a coarse to fine scheme is used employing image pyramids. It can be described as follows: (i) The grid meshes are split recursively into four smaller squares in X- and Y-direction if the surface is considered to be rough. (ii) A smoothness weighting defined as the inverse proportion of the difference between the Z value of a grid point and the Z values of the neighboring grid points is applied to smoothen a rough surface.

The method assumes that camera calibration and orientation are known. Following this assumption, one can calculate the projection of a grid point into the images, and thus its brightness and the average brightness for all images. The optimization is based on that the Z value of an adjusted observation can be bilinearly interpolated from the Z values of the neighboring grid points. By means of least squares optimization, the Z values of the grid points are changed so that the difference between the projected intensities is minimized. The adjusted values z for the improvement of Z, which can be positive or negative, are calculated by:

$$z = N^{-1}n, \tag{3.1}$$

with $N = A^T P A$, the normal equation matrix and $n = A^T P l$, the absolute vector. The intensity values of the points projected into the different images are the observations (l vector). While the rows of the design matrix A represent the observations, i.e., the image grid points which are projected into different cameras, the columns are for the unknowns, i.e., height grid points. The design matrix is different from zero only for those four points of the height grid that make up the mesh where an image point resides.

The values for the design matrix are constructed from the gradient of the intensity in x and y direction, the derivative of the projection from the 3D point into the image, and the bilinear transformation for the height of the grid point. It basically gives the differential of intensity when the height of a grid point is changed. In addition, the weight matrix P is used to deal with weakly texture areas and image noise.

An advantage of Heipke's approach is that the Z values of the grid points are calculated by highly accurate least squares adjustment. Therefore, the reconstructed surface is often close to the real one. Additionally, also an estimate of the accuracy can be computed. The second advantage of the approach is that owing to the smoothing by the weight matrix, it can handle complex surfaces with weak texture or with strong image noise. There are

problems with this for some of the approaches in Section 3.2. Third, as this approach applies both mesh representation and least squares, it takes full advantage of both. As a result, it can integrate different sources of information (image and geometry) in the reconstruction to obtain a reconstructed surface only minimally different from the real surface. Because of all these advantages, we apply this technique as a basis for our thesis.

Heipke's technique effectively works for 2.5D surfaces. So the main purpose of our algorithm is to extend it to full 3D reconstruction. Additionally, Heipke's technique handles occlusions and non Lambertion reflections only partly. Thus, the second purpose of our novel algorithm is to take these issues into account by exploiting the advantages of mesh representation and by using robust estimation.

Chapter 4

A Novel 3D Least Squares Matching Approach

In this chapter, we present our method for the reconstruction of fully 3D surfaces which was first introduced in a preliminary version in Ton and Mayer (2007). It focuses on the matching algorithm and uses given initial sparse highly precise 3D points extracted from images during pose-estimation as input for our method. An important contribution of our algorithm is our method for positioning additional unknown points on top of the given sparse 3D points.

This chapter includes six sections corresponding to different steps of our algorithm. In the first section, we describe foundations of our algorithm including the camera model, the model for image formation including sub-pixel interpolation, and finally pose estimation generating highly precise though sparse initial 3D points for the reconstruction. The second section presents the basic idea and the properties of our algorithm. We then discuss in detail how we derive the partial derivatives for the design matrix in the third section. In the fourth section, we describe the way we build triangulation meshes from given sparse 3D points. Based on these initial meshes, we position new unknown 3D points (cf. fifth section). In the last section, we show how we produce image observations from the obtained 3D unknown points. Also the robust least squares adjustment applied to obtain a precise and accurate reconstruction are also discussed in this section.

4.1 Foundations of the Proposed Algorithm

4.1.1 Camera Model

A basic task for 3D reconstruction is to transform an object in 3D space into the 2D image space of an observing camera. This implies that we also need to determine the parameters for the transformation from the 3D world to the 2D image. Parameters that need to be determined are: (i) Extrinsic or external parameters comprising the orientation (rotation) and the location (translation) of the camera; (ii) intrinsic or internal parameters which describe

characteristics of the camera like the coordinates of the principal point (the projection of the image center to the image plane), the principal distance (camera constant), and the skew and scales of the two image axes.

Following the common practice in the field, we use a pinhole camera model. The transfer from the 3D world to a 2D image by means of a pinhole camera is a projection process in which one dimension is lost (HARTLEY and ZISSERMAN 2003). We denote the 3D point as $L_i = (X, Y, Z)^T$. C is the center of the camera or the **focal point** and the distance from the focal point C to the image is the principal distance often also termed the **focal length** f even though this is not strictly correct from the point of view of optics. In Figures 4.1 and 4.2 we visualize the perspective projection model of a pinhole camera and the Euclidean transformation between the world and the camera coordinate system.
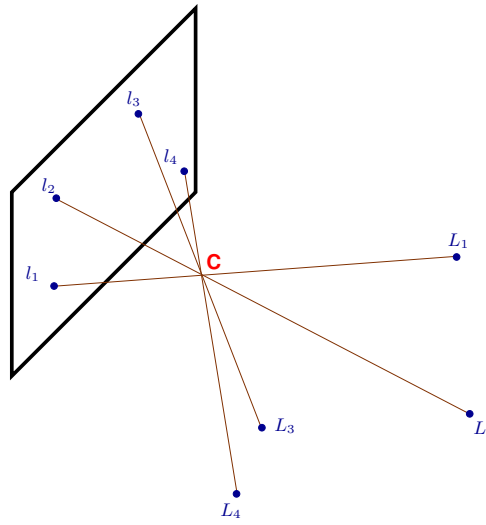

Figure 4.1: Perspective projection: Image points l_i are the intersections of the image plane with straight lines from the 3D points L_i through the pinhole camera center C (HARTLEY and ZISSERMAN 2003).

The projection can then be mathematically modeled by the central projection in which the 2D point $l_i = (u, v)^T$ is the intersection of the straight line connecting the 3D point L_i with the the the camera center C with the image plane. Thus, the three points: L_i, l_i and C are collinear. The projection of (homogeneous) 3D points L_i to image points l_i may be expressed in terms of a linear mapping of homogeneous coordinates (see Figure 4.2). To transform 3D Euclidean points into homogeneous coordinates, it suffices to add a 1 as the last element of each coordinate. Thus, we can express the 3D points by $\mathbf{L} = (X, Y, Z, 1)^T$, the 2D image points by $\mathbf{l} = (u, v, 1)^T$, and the homogeneous coordinate for the pinhole camera at the origin as $(0, 0, 0, 1)^T$. The general relationship between a 3D point $\mathbf{L}$ and its image $\mathbf{l}$ in the image plane can be expressed by:

$$\lambda \mathbf{l} = \underbrace{\mathbf{K} R^T [I| - T]}_{\mathbf{P}} \mathbf{L} = \mathbf{P} \mathbf{L} \tag{4.1}$$

with λ a scale factor proportional to the **depth** Z. $\mathbf{K}$ is the 3×3 camera **intrinsic (calibration) matrix** (see Figure 4.3) describing the characteristics of the camera:

$$\mathbf{K} = \begin{bmatrix} \gamma f & sf & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.2)$$

where (u_0, v_0) are the coordinates of the principal point, f is the focal length, γ the **aspect ratio** which captures non-quadratic pixels, and s is the skew of the two coordinate axes used to model non-rectangular pixels. According to this definition, there are 5 parameters in the intrinsic matrix. For most practical cameras $s \approx 0$, $\gamma \approx 1$, and the principal point is close to the center of the image.
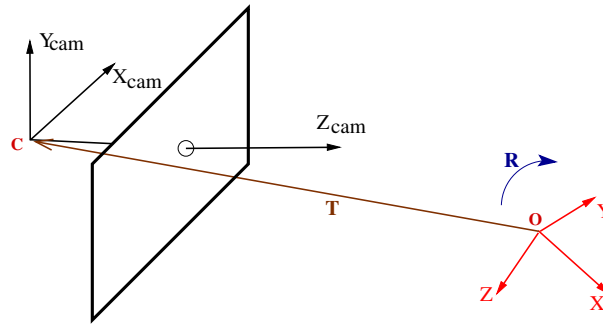

Figure 4.2: Euclidean transformation between the world and the camera coordinate frame.

The 3×4 **P** matrix in Equation (4.1), is called the (camera) projection matrix: $\mathbf{P} = \mathbf{K}R^T[I|-T]$. R is a 3×3 orthogonal matrix, i.e., a rotation matrix, which represents the orientation of the camera coordinate frame in the Euclidean object coordinate system. Finally, T is a 1×3 Euclidean vector describing the translation (or location) of the camera coordinate frame (see Figure 4.2). The three parameters in the rotation matrix **P** and the three parameters in the translation vector T compose the six extrinsic parameters of the camera. In total, we need to calculate eleven parameters in the pose estimation step (five intrinsic and six extrinsic).
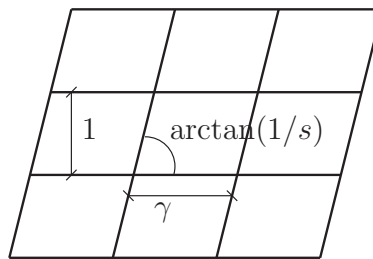

Figure 4.3: The intrinsic parameters γ and s according to (MEER 2004).

4.1.2   Image Formation and Sub-pixel Interpolation

An important part of our approach consists in the derivation of brightness values from digital images given 3D points. A digital image consists of a two-dimensional matrix G with elements g_{ij} called pixels (cf. Figure 4.4).
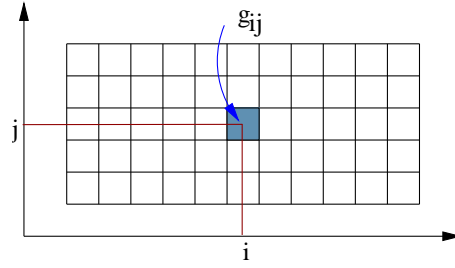
Figure 4.4: 2D pixels g_{ij} of a digital image.

The value of a pixel depends on the type of sensor, sensor or camera parameters (sensitivity to electromagnetic radiation, etc.), atmospheric parameters, object surface parameters (reflection), and illumination parameters (number, direction. and brightness of illumination sources).

For deriving brightness values for arbitrary image positions, a crucial step is image interpolation. It is necessary when the position is not exactly the center of a pixel, i.e., no integer value, but at a sub-pixel position. Usually, the value is derived by an interpolation function.

Several interpolation functions have been investigated for resampling images such as higher-order polynomial functions and bi-linear interpolation. In this thesis, we apply the bi-linear interpolation function as it has a low computational complexity, acceptable aliasing effects, and doesn't smooth the image function too strongly. We thus assume that the brightness function varies bi-linearly in a local neighborhood.
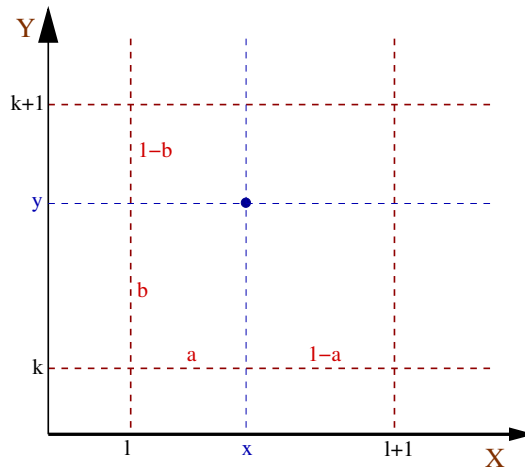

Figure 4.5: Bilinear interpolation.

Bi-linear interpolation (see Figure 4.5) determines the value at a sub-pixel position based on a weighted average of the four pixels in its 2×2 neighborhood. The new image pixel value is calculated as follows:

$$g = (1 - a)(1 - b)g_{(l,k)} + a(1 - b)g_{(l+1,k)} + (1 - a)bg_{(l,k+1)} + abg_{(l+1,k+1)} \qquad (4.3)$$

with a, b the fractional parts of the neighborhood (see Figure 4.5).

4.1.3   Pose Estimation and 3D Point Reconstruction

Our dense 3D reconstruction is based on given sparse, but highly precise and reliable points obtained by applying an automatic orientation procedure. Particularly, we apply the approach of BARTELSEN and MAYER (2010) which assumes that all images are taken knowing at least approximately their calibration (intrinsic) parameters. According to BARTELSEN and MAYER (2010), 3D points can be reconstructed by a procedure with five steps: (1) Point extraction and matching, (2) robust 3D reconstruction for pairs and triplets based on essential matrix and (calibrated) trifocal tensor estimation, (3) robust bundle adjustment, (4) linking of image triplets into a sequence, and finally (5) refinement by hierarchical processing.

BARTELSEN and MAYER (2010) follow MIKHAIL et al. (2001) by focusing on least squares adjustment, particularly robust bundle adjustment, and least squares matching (LSM). With this they obtain highly precise conjugate points. In addition, the combination of LSM and bundle adjustment results in reliable, precise, and accurate 3D points. Because of this, we use their sparse 3D points as input for our dense reconstruction. In the remainder of this section, we will explain each step more in detail.

Point Extraction and Matching

The objective of the first step is to obtain precise (relative) coordinates for conjugate points. This includes extracting interest points in each image, searching for conjugate points, and determining the exact subpixel matching position. The method can deal with images rotated around the optical axis of the camera by making use of orientation information derived by point extraction. The first stage is to extract Förstner points (FÖRSTNER and GÜLCH 1987). For a homogeneous point distribution, regional non-maximum suppression is used to remove weak points if their position is close (two to five pixels).

In the second stage, points are matched using LSM resulting in highly precise conjugate points. For large baseline scenarios, the full image size is used as search space. To reduce the number of candidates, unlikely candidates for conjugate points are sorted out by means of normalized cross correlation. Most of the correct points are retained by employing a relatively low threshold. By using all three color layers, the sensitivity of the test is improved. For the least squares matching, six affine geometric parameters are used together with two radiometric parameters in the form of bias and contrast. For two images, the second is matched to the first. For more than two images, the average image in the geometry of the reference image is employed for matching. For usual image sizes, image pyramids are used to reduce the complexity of the matching.

At the end of this step, highly precise image coordinates for the conjugate points are available. The next step aims at robust 3D reconstruction.

Robust 3D Reconstruction

Due to problems such as image noise, similar or repeating structures in the image, and occlusions leading to less than 20% correct matches in some of the more difficult scenes, a robust

Figure 4.6: Wide baseline triplet with conjugate matched points and epipolar lines.

approach is necessary. According to BARTELSEN and MAYER (2010), triplets are suggested as basic building block for this purpose. For triplets, one can use matching points in two images and project them into the third image. Thus, one can check if the matching points comply with the projected points. This helps to sort out wrong matches and obtain highly reliable matches through the intersection of three image rays. Even though triplets are employed as basic, the reconstruction starts with image pairs to narrow the search space via epipolar lines and thus avoids the bad combinatorics for three images.

To derive the (calibrated) relations between pairs or triples, the essential matrix $\mathbf{E}$ and the (calibrated) trifocal tensor $\mathbf{T}$ is employed. For a robust estimation, RANSAC (cf. Section 2.2.2) is used. Triplets are reconstructed sequentially and are linked by projecting points of the preceding triplet via the newly calculated $\mathbf{T}$ into the new last image. Thus, new points which have not been seen are added to the calculated 3D points.

RANSAC provides reliable results. However, it can be computationally demanding especially when the number of outliers is high and also its precision is often limited. To alleviate these problems, BARTELSEN and MAYER (2010) suggest to combine RANSAC with robust bundle adjustment in a way similar to the Expectation Maximization (EM) algorithm (DEMPSTER et al. 1977) .

Robust Bundle Adjustment

Bundle adjustment is a core asset of the approach of BARTELSEN and MAYER (2010) for extracting reliable and precise 3D points from images while reducing the runtime compared to the standard RANSAC algorithm. The idea is to adjust all promising reconstructions. Owing to the enforcement of highly precise results for a large number of points, one can be rather certain that the solution is not random.

When combining RANSAC with robust bundle adjustment, first $\mathbf{E}$ and $\mathbf{T}$ are estimated by RANSAC for junks of a couple of hundred iterations. On the computed optimum a robust bundle adjustment is run. By iterating this several times, several outcomes are obtained from

which the best solution is retained. For a more stable solution in the presence of outliers, outliers are reweighted and then possibly eliminated in the bundle adjustment.

Hierarchical Processing via Pyramids

When the image size is large, image pyramids are employed. As the percentage or direction of overlap between images is not known, the hierarchical scheme helps to guarantee a high probability for a correct result under mild, very realistic assumptions. BARTELSEN and MAYER (2010) suggest to compute image pyramids with a reduction factor of 2. They found that a stable and precise solution can be obtained on higher levels of the pyramid (e.g., with 120 pixels in one dimension). To make use of the information from the original resolution, the points are tracked via LSM down to the original solution, but only after the sequence has been completely oriented the higher levels. By this means, an efficient solution even for large images of many Mega pixels is obtained. After the tracking, a final bundle adjustment is applied considering also radial distortion.

The final result of the approach of BARTELSEN and MAYER (2010) are sparse 3D points which are the basis of our algorithm for dense 3D reconstruction. In the next sections, we describe the proposed algorithm in detail.

4.2 Basic Ideas and Outline of the Algorithm

The purpose of our approach is to reconstruct a dense fully 3D surface via a formulation in terms of least squares adjustment. To be able to work in full 3D, we triangulate the surface and move the vertices of the triangulation along a path independent from the definition of the coordinate system, namely the direction of the normals at the vertices of the triangulated surface.

In this thesis, the direction of the normal at the vertex N_u for the unknown u is estimated as the weighted average of the normal vectors of the planes (triangles) attached to the vertex. The weight b_i applied for each triangle is the number of observations obtained in the triangle which will be discussed in Section 4.5.1. E.g., for Figure 4.7 this means:

$$N_u = \frac{b_1 N_1 + b_2 N_2 + b_3 N_3 + b_4 N_4 + b_5 N_5}{\sum_{i=1}^{5} b_i}.$$

It is then normalized by $N_u^{norm} = \frac{N_u}{\|N_u\|}$.

The basic ideas of our algorithm can be summarized as follows:

- The reconstruction is based on a triangulated 3D surface.

- Points inside the 3D triangles are projected in the images resulting in the observations.
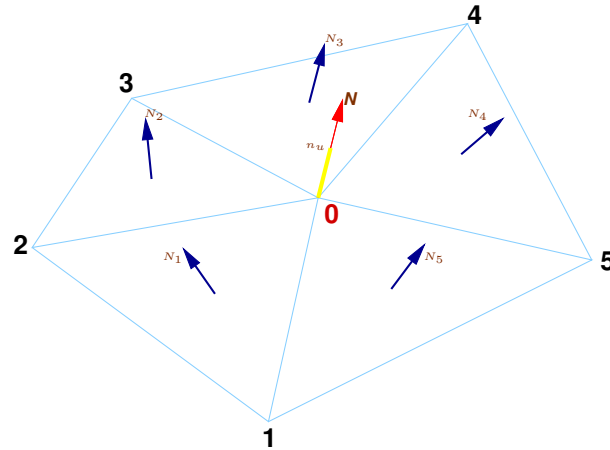
Figure 4.7: Relation of normal N_u at vertex **0** to normals of neighboring planes and unknown size of movement n_u.

- The vertices of the triangles move along their respective normals with the goal to obtain a summed squared gray value difference as small as possible. The differences are calculated between the observations and their average value supposed to be the reflectance value of the surface in a least squares sense. The sizes of movement are determined by the unknowns n_u. (cf. Figure 4.7).

- Additionally to the image observations representing the data term, the surface is regularized by observations enforcing its local smoothness in terms of curvature.

- To deal with outliers, e.g., in the form of local occlusions, and non-Lambertian reflection, robust estimation is used.

- To expand the range of convergence, we employ a hierarchy of resolutions for the triangulation linked to the adequate level of image pyramids. We apply Gaussian filters as smoothing kernels when generating image pyramids.

Our reconstruction procedure is as follows (cf. the flowchart of our algorithm in Figure 4.8): After the first step of constructing a mesh of triangles from given sparse 3D points, the algorithm solves the reconstruction problem with an iterative procedure starting at the first and ending at the last pyramid level.

For each pyramid level, the algorithm consists of two steps: First, unknowns are built splitting the triangles to adapt the density of triangulation to the different resolution levels. Second, we extract the triangles of the split mesh which are visible by each camera and then run least squares to adjust the positions of the unknowns by minimizing the variance of the difference in intensity. This step is done from the first camera until the last camera on each pyramid level. This is possible owing to the fact that we employ given sparse 3D points (cf. the preceding section) for which it is also known in which image they have been measured. We do not run least squares adjustment for the whole mesh. We instead apply the estimation for each part of the mesh that can be seen by each camera from the first until the last camera separately. We particularly start by running least squares adjustment for all meshes

```
                    ┌─────────────────┐
                    │     Start       │
                    │ First Pyramid level │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │ Build Unknowns  │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │ Extract Triangles │
                    │ Visible by Camera │
                    └─────────────────┘
                             │
                             ▼
                    ┌─────────────────┐
                    │ Run Least Squares │
                    │   Adjustment    │
                    └─────────────────┘
                             │
                             ▼
                        ◇  Next
                           Camera ?  ◇ ── Next camera
                             │
                    Process all cameras
                             │
                             ▼
                        ◇  Next
                      Pyramid Level  ◇ ── Next pyramid level
                           ?
                             │
                  Process all pyramid levels
                             │
                             ▼
                    ┌─────────────────┐
                    │      End        │
                    └─────────────────┘
```
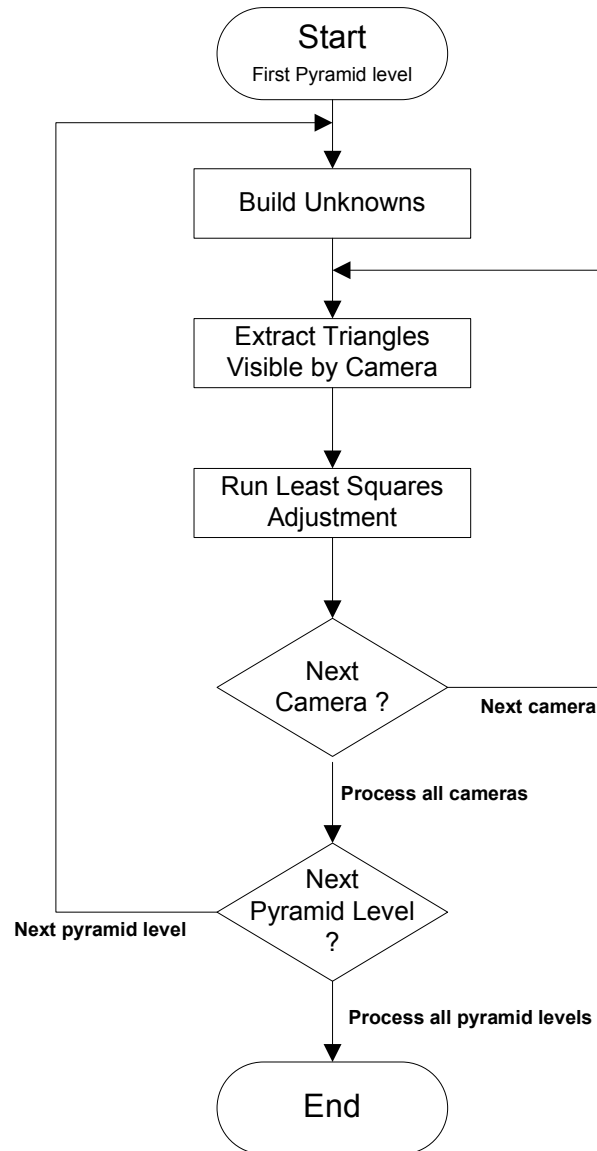
Figure 4.8: Flowchart of the reconstruction procedure.

which can be seen from the first camera. For the second and further cameras, least squares adjustment is run only for those meshes which can be seen from them and have not been adjusted before. With this, we limit the runtime to an acceptable size. Thus, global estimation with a large runtime is replaced by only regionally consistent estimations with a lower runtime. At the end of this step, we obtain additional precise and accurate 3D points. For a reliable estimation, we account for outliers by using robust estimation. The result of the whole procedure is a densely reconstructed 3D surface from all defined 3D points.

The next sections are organized as follows: First, we present the content of our design matrix for the least squares estimation which is constructed in the course of the algorithm. Second, we describe how we create triangulation meshes from the given 3D points, i.e., the first step in the flow chart. Third, in Section 4.5 it is discussed how unknowns are positioned.

The next step in the flow chart is the estimation with robust least squares adjustment which is presented in Section 4.6. In this section, we also present how we factorize the normal equation matrix and how we weight down outliers.

4.3 Partial Derivatives for the Design Matrix

The image observations are devised to describe how well the intensities in all images showing a point on the 3D surface fit to an average intensity computed from all these images by taking the difference between the individual intensities and the average. To minimize the squared summed of these observations, the vertices of a triangulation are moved along their local normals (cf. Figure 4.9).
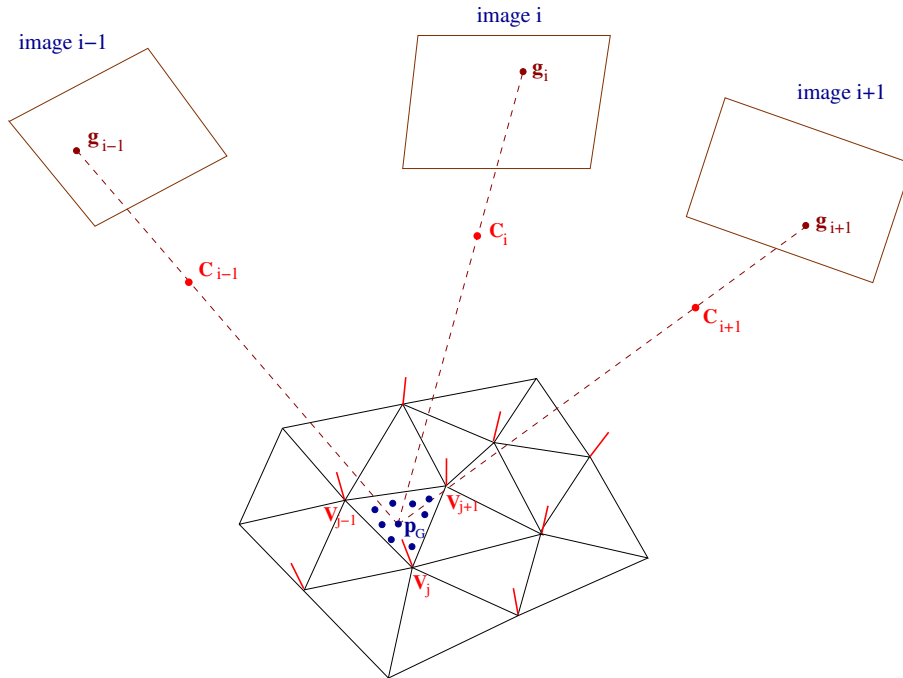
Figure 4.9: Basic principle of least squares 3D reconstruction from multiple images. The intensity at the 3D point P is computed as the (weighted) average G of the intensities in the individual images. The goal is to minimize the squared differences of the observations, i.e, the difference of the intensities in the individual image g_i to G by moving the vertices V_u of the triangulation along their normals (unknown size of movement n_u).

For this non-linear problem the design matrix consists of the partial derivatives of the intensity value I_i of observation i in an image according to the movement n_u of an unknown u. They can first be factorized according to the x- and y- direction of the image:

$$\frac{\partial I_i}{\partial n_u} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial n_u} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial n_u}$$

with

- $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ the image gradients in x and y direction which can, e.g., be estimated by the Sobel operator.

The derivatives of the image coordinate concerning the unknowns can further be factorized according to the movements in the X-, Y-, and Z- direction of the 3D point corresponding to the observation i:

$$
\begin{aligned}
\frac{\partial I_i}{\partial n_u} \quad = \quad & \frac{\partial I}{\partial x}\left(\frac{\partial x}{\partial X}\frac{\partial X}{\partial n_u} + \frac{\partial x}{\partial Y}\frac{\partial Y}{\partial n_u} + \frac{\partial x}{\partial Z}\frac{\partial Z}{\partial n_u}\right) \\
& + \frac{\partial I}{\partial y}\left(\frac{\partial y}{\partial X}\frac{\partial X}{\partial n_u} + \frac{\partial y}{\partial Y}\frac{\partial Y}{\partial n_u} + \frac{\partial y}{\partial Z}\frac{\partial Z}{\partial n_u}\right)
\end{aligned}
$$

with

- $\frac{\partial x|\partial y}{\partial X|\partial Y|\partial Z}$ describing how the position in the x- or y-direction in the image is affected by changing the 3D point coordinates X, Y, or Z, and

- $\frac{\partial X|\partial Y|\partial Z}{\partial n_u}$ the derivatives of the 3D point coordinates X, Y, and Z according to the unknown. The vertices move in the direction of N_u. The derivatives in X-, Y- and Z-direction depend on the distance of the 3D point from the line connecting the other two vertices of the triangle where the point is lying in.

We model the projection of (homogeneous) 3D points $\mathbf{L}$ to image points $\mathbf{l}$ by $\lambda \mathbf{l'} = \mathbf{PL}$ (cf. Equation (4.1)). As Bartelsen and Mayer (2010) we additionally employ a quadratic and a quartic term to model the radial distortion:

$$
ds = k_2(r^2 - r_0^2) + k_4(r^4 - r_0^4), \tag{4.4}
$$

with r the distance to the principal point and r_0 the distance where ds equals zero.

As the basic way of operation of Bartelsen and Mayer (2010) is using only images, all calculations are conducted in a relative coordinate system. The projection center of the first camera is used as its origin (see Figure 4.10). The rotation of the first camera is fixed and supposed to point to the negative z-direction. Finally, the distance of the first and the second camera is set to one.

4.4 Triangulation of Given Sparse 3D Points

As we use given sparse accurate 3D points (cf. Section 4.1), the first step for a full 3D approach is the linking of the points to form triangles. This step is not trivial as it is at least difficult, often even impossible to link points in 3D just based on proximity. E.g., consider a thin surface, where points on both sides of the surface should not be linked, but might be much closer than points on the same side of the surface. To avoid this problem, we split the set of images into overlapping triplets. For them we assume that the topology of the 3D

Figure 4.10: The position and orientation of the first camera C_0 supposed to define the origin and to point in the negative Z-direction together with the distance from C_0 to C_1 (scale) which is set to 1 are used to define the coordinate system.

points can be modeled in 2D in the images. We therefore can triangulate the points for the triplets in one of the images. To obtain compact triangles, we employ Delaunay triangulation. This reduces problems with elongated thin triangles.

We first project via equation (4.1) and the known camera parameters the given 3D points into the central image of the triplet. There they are triangulated. Triangulations for different triplets are manually stitched together which leads to full 3D triangulations. The following steps work on this basic global triangulation in as many images as available. We consider the given 3D points as control points and denote the obtained mesh as M.

Second, based on the input information from Section 4.1 about which points have been observed by which camera, we decide which part of mesh M, i.e., which triangles, are related to the first camera C_0, the second camera and so on until the last camera. This is the basis for our procedure which cycles through the parts of the mesh seen by each camera. We denote the part of mesh M that can be seen by camera C_0 as mesh M_0 consisting of q triangles (cf. Figure 4.11(a)).

Because there are not many given sparse 3D points, the surface cannot be precisely reconstructed based only on them. Thus, we need to position additional points, i.e., the unknowns. Particularly, we split the observed triangles into smaller ones to account for the possibly big differences of texture and thus intensity of the observations in a triangle. This is discussed in the next section.
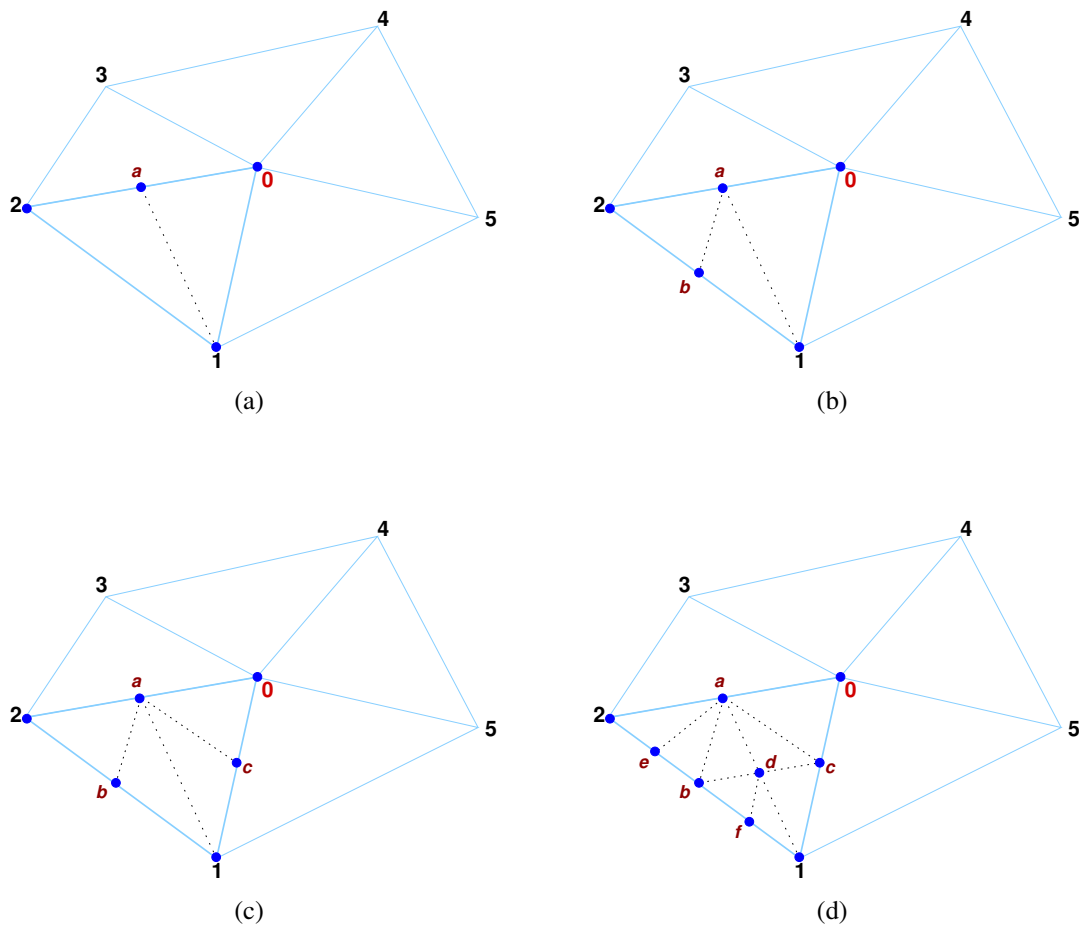
Figure 4.11: Part of the triangulation mesh that can be seen by the first camera C_0 marked by numbers **0** to **5** (a). The first level of generated observations are the points a, b, and c (a to c) and the second level are the points d, e and f (d).

4.5 Creation of Unknowns and Coarse-To-Fine Strategy

To account for big differences in texture and thus intensity, we split triangles by additional points (unknowns). However, this is usefully only if a triangle is in a sufficiently textured area, i.e., not an area of homogeneous intensity. For homogeneous areas, we do not need to split the triangle, because the image information is too weak. To determine the positions for unknowns, we first construct internal 3D points in each triangle (cf. Section 4.5.1). Based on these internal points we select which triangles are to be split by considering the average absolute deviation of the intensity values in each triangle in the mesh. This deviation is the sum of the average absolute differences of the grey value between the projected points in the images and their average value which is assumed to be the intensity value at the surface (cf. Section 4.5.2). Finally, for each selected triangle, we decide about the position of the unknowns. This includes the stages detailed in the following sections and is done for all triangles in the whole mesh M of each pyramid level.

4.5.1   Construction of Internal Points in Each Triangle

The construction of internal points is done consecutively for each triangle in the mesh from the first to the last triangle. To construct the internal points in each triangle, we first determine which camera is the main camera C_{m,T_1} of the triangle. From the input information we know which set of cameras has been used to determine the specific vertices of the triangle. For example, the three vertices of the first triangle T_1 in the mesh M (cf. Figure 4.12) have been seen by the four cameras $\mathbf{C}_0$, $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$. We define the set of k camera positions that see the triangle as $\mathbf{C} = \mathbf{C}_0 \cap \mathbf{C}_1 \cap \mathbf{C}_2 \cap \mathbf{C}_3$. The main camera of a triangle within these k cameras is defined as the camera in which the angle between the direction from the center of the triangle to the projection center of the camera and the normal vector of the triangle is smallest (see Figure 4.12). The normal vector of the triangle is the vector perpendicular to the plane on which the triangle resides.
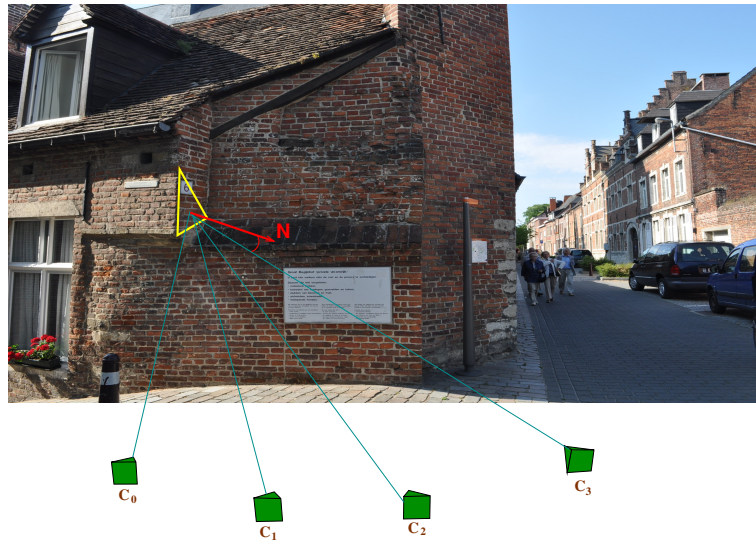
Figure 4.12: Determination of the main camera of a triangle (yellow edges). The triangle with normal vector N has been determined from four cameras (C_0, C_1,.., C_3). Camera C_1 is the main camera of this triangle because the angle between N and the line connecting the projection center of the camera and the triangle center is smallest.

Projecting the 3D triangle T_1 to its main camera C_{m,T_1} via Equation (4.1), we obtain a 2D triangle (triangle **021** in Figure 4.11). Observations which are the projected points of the internal points to the camera are generated by splitting the sides of the projected triangle at the center if the length of a side is beyond a given threshold which is set to 1.4 pixels (the length of the diagonal of a right angle triangle with a side length of 1 pixel for the other two sides). The 3D positions of the generated observations, i.e., the 3D internal points, are obtained by linear interpolation. This leads to the first level of observations (points a, b, and c) in Figure 4.11(a to c). If the length of the sides should still be above the given threshold, i.e., the triangle is rather big, we split the triangle again and obtain the second level of observations (points d, e, and f) in Figure 4.11(d). This is done recursively until the lengths of all sides are below the threshold. After this step, we have a number of observations in each triangle. We denote the 3D internal points obtained in triangle T_1 after this step as L_1

to $L_{n_{T_1}}$ (excluding the three vertices of the triangle).

4.5.2 Selection of Triangles For Unknowns

The decision on which triangles should be selected as unknowns is based on the average absolute differences in intensity values (called the average deviation) of the internal points of the triangles. When the average deviation is small, the intensity values are quite homogeneous in the triangle and thus it is not useful to position additional points in this triangle. Therefore, our rule is that only the α % of the triangles in mesh M with the highest average deviation are considered for the insertion of unknowns. How many percent of triangles are selected depends on the characteristic of the texture of the mesh. We will explain how we set this parameter in Chapter 5. In the next paragraph, we describe how we calculate the average deviation.

As the lighting might vary between images, the camera might use a different gain. Also surfaces might have a non-Lambertian bidirectional reflection distribution function (BRDF). Even though it is difficult or even impossible to account for all these issues, we found empirically, that using a bias parameter per image is sufficient for many scenes.

To determine the bias parameter for each image, we calculate the intensity deviation for each observation in the mesh. Denote the number of cameras that can see the whole triangulated mesh M as K and the number of camera images that can extract triangle T_1 as k. As there are k cameras that see triangle T_1, the projection of each 3D internal point L_i on the k cameras' images via Equation (4.1) provides us with a vector of k intensity values $I_{i,j} = g(x, y)$ for each projected point (observation) $l_{i,j}$ with $j = 1$ to k. For g, we use the bilinear interpolation function (cf. equation (4.3) in Section 4.1.1). Denote the bias parameter for each image j that can see the global mesh as β_j. The mean intensity value for an internal point L_i in triangle T_1 is $I_{i,M} = \sum_{j=1}^{k} I_{i,j}/k$. Denote by $\epsilon_{i,j}$ the deviation of an intensity value $I_{i,j}$ from its mean $I_{i,M}$. Thus, for each image we can determine the median value of all $\epsilon_{i,j}$ with $i = 1$ to N_j which is the total number of 3D internal points in the mesh that can be seen by camera j. Denote these K median values for K cameras that can see mesh M as $Med_0, Med_1 \ldots Med_{K-1}$. The bias parameter of an image is then defined as follows:

$$\beta_j = \begin{cases} 0 & \text{for j=0,} \\ Med_j - Med_0 & \text{for j} \neq 0. \end{cases}$$

After the bias parameter is defined for each image, we recalculate the mean intensity value for an internal point L_i taking into account the bias parameters as follows:

$$I_{i,M} = \frac{\sum_{j=1}^{k}(I_{i,j} - \beta_j)}{k} \tag{4.5}$$

Considering the bias parameter, the deviation of an intensity value is then calculated as $\epsilon_{i,j} = I_{i,j} - \beta_j - I_{i,M}$. The average absolute deviation of intensity values of internal points L_i in triangle T_1 from k cameras becomes then:

$$S_i = \frac{\sum\limits_{j=1}^{k} |\epsilon_{i,j}|}{k}. \tag{4.6}$$

The average absolute deviation of n_{T_1} internal points in triangle T_1 which measures the homogeneity of the intensity in a triangle is finally:

$$S_{T_1} = \frac{\sum\limits_{i=1}^{n_{T_1}} S_i}{n_{T_1}}. \tag{4.7}$$

We calculate it for all triangles in mesh M to obtain $S_{T_1}, S_{T_2},.., S_{T_Q}$ with Q the number of triangles in the whole mesh. We then sort the values. From the descending list, we select the top $\alpha\%$ triangles with the highest average deviation for the insertion of the unknowns. We position unknowns starting with the triangle that has the highest average deviation. In the next section, we describe how unknowns are positioned in the selected triangles.

4.5.3 Positioning of Unknowns in the Selected Triangles

The purpose of inserting unknowns in triangles with a high average deviation is to split these triangles to obtain lower deviations in the split triangles by adjusting the new unknowns. Thus, they create meshes which reflect the surface more precisely.

To define unknowns in the selected triangles, we split the sides of the triangle at the center of each side in the same way we generate internal points (cf. Section 4.5.1). However, there is an important difference: How many sides and which sides are split depends on the shape and the size of each triangle. We classify triangles into four types: (i) Triangles with all sides less than (empirically determined) 14 pixels, (ii) triangles with two sides less than 14 pixels, (iii) triangles with at least two sides longer than 14 pixels and the angle between these two sides less than a threshold, and (iv) triangles which do not belong to the first three types. We will describe how the threshold for the angle is set in Chapter 5.

We do not split triangles belonging to the first type because such triangles are small enough. For the second type, we split the triangle at the center of its longest side, i.e., only one unknown is positioned for this type of triangle and thus we obtain two new triangles (see Figure 4.13(a)). For the third type of triangle, two unknowns are inserted at the center of the two longest sides of the triangle which leads to three triangles after the split (Figure 4.13(b)). For the last category, we split the triangle at the centers of all three sides resulting in four new triangles. However, there are two ways of splitting this triangle into four triangles depending on which distance is the shortest (Figure 4.13(c), (d)).

For the newly created 3D points and triangles we apply the previous step to calculate the average deviation of intensity values. Based on these average deviations, we insert the new triangles into the descending list to check whether the new triangles still need to be split. A completed split triangle is presented in Figure 4.14. In this figure, the red points are unknowns, the blue points are internal points and the brown points are given (control) points.
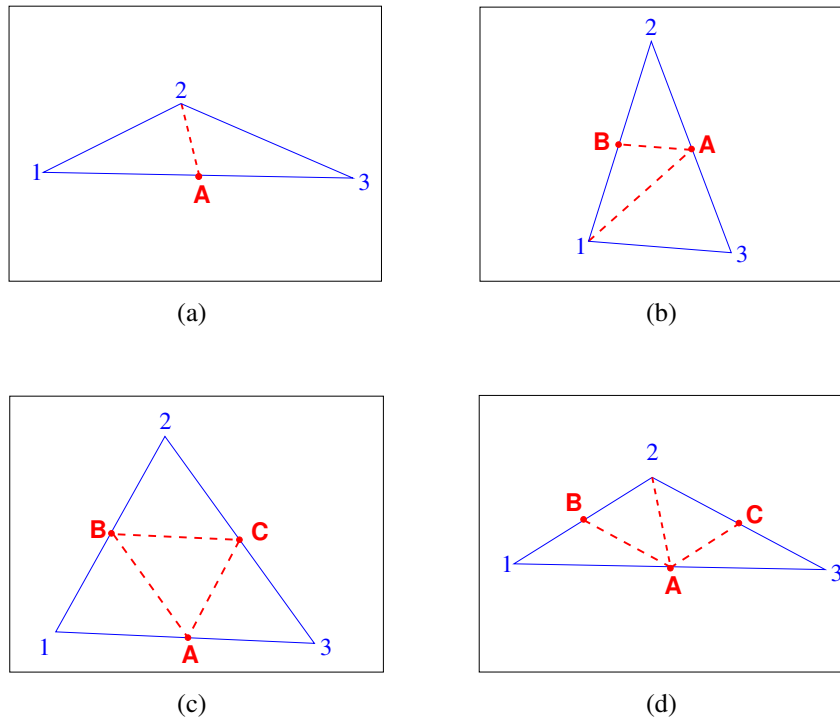

Figure 4.13: Generation of unknowns in a selected triangle. For a triangle which has only one side longer than 14 pixels, one unknown is inserted at the center of its longest side (a). For a triangle with at least two sides longer than 14 pixels and the angle between these two sides smaller than a threshold, two unknowns at the center of the first and the second longest sides are generated (b). For triangles which do not belong to the former three types, three unknowns at the center of each side are inserted depending on distances the unknowns are linked differently (c), (d).

As almost always outliers due to noise and occlusions exist, we regularize the ill-posed problem by enforcing the smoothness of the surface via additional observations relating the unknowns. Thus, before applying robust least squares adjustment to get precise positions for unknowns, we smooth unknowns. We discuss this in the next section.

4.5.4   Regularization of Unknowns by Smoothing

For each unknown, smoothness is described in terms of the deviation h_{smooth} of a vertex from an average plane in the direction of its normal N derived from the neighboring vertices. We denote $\mathbf{0}$ as one unknown in mesh M and five points adjacent to $\mathbf{0}$ as vertices $\mathbf{1}$, $\mathbf{2}$, $\mathbf{3}$, $\mathbf{4}$ and $\mathbf{5}$ (see Figure 4.15).
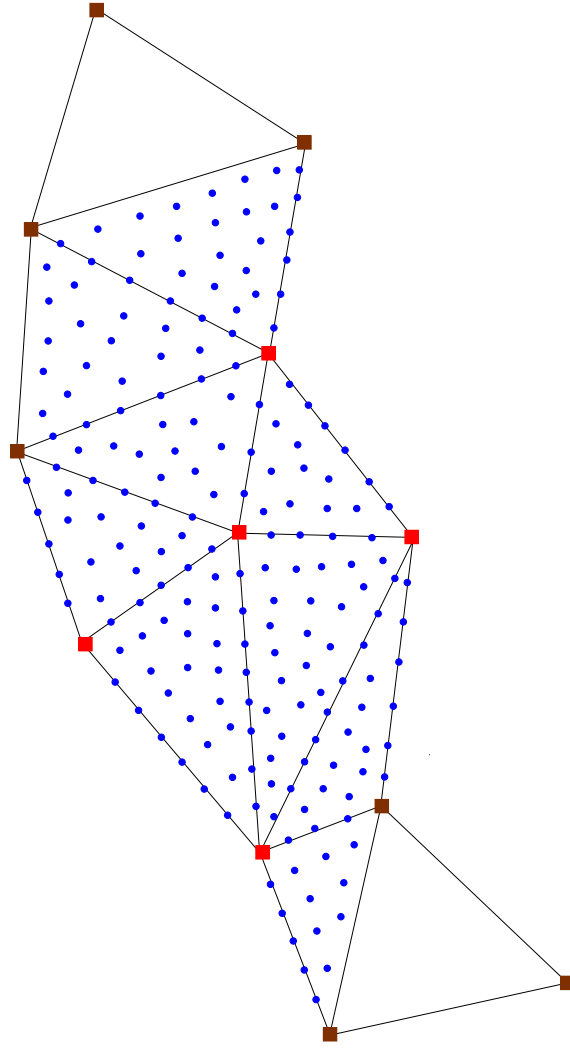
Figure 4.14: The completed splitting of triangles. The red points are the unknowns, the blue points the internal points, and the brown points the control points.

The average plane at $\mathbf{0}$ is computed as weighted average of the heights of the vertices h_i above the plane through $\mathbf{0}$ and perpendicular to its normal N. The adjacent vertices are projected along the normal N, resulting in the primed (red) numbers in Figure 4.15. h_i has a positive sign if the projection has the same direction as the normal vector N and a negative sign, otherwise. We denote the distances from $\mathbf{0}$ to the projected vertices as d_i. Weighting is done according to the inverse distance $1/d_i$ of the points. The average height of the vertices from $\mathbf{1}$ to $\mathbf{5}$ in Figure 4.15 is calculated by:

$$h_{smooth} = \sum_{i=1}^{5} \frac{h_i}{d_i} / \sum_{i=1}^{5} \frac{1}{d_i}. \tag{4.8}$$

To position $\mathbf{0}$ (smoothly) on the plane $h_0 = -h_{smooth}$ has to hold. I.e., the weighted average height of the adjacent vertices should be equal to the height of the unknown $\mathbf{0}$ above or below the plane. This means that by smoothing we expect lower average differences between h_i and
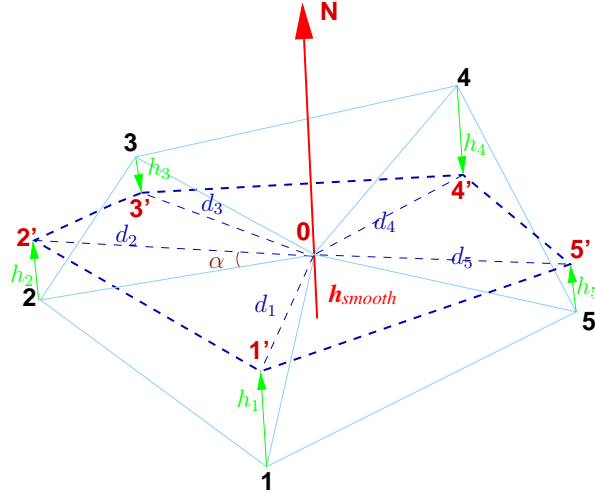
Figure 4.15: Smoothness – The (black) numbers denote the original vertices. The (red) primed numbers show their projection on the plane perpendicular to the normal N through the given point $\mathbf{0}$ with height h_i. h_{smooth} corresponds to the height of the given vertex above or below the (weighted) average plane.

h_0.

By the creation of unknowns, the mesh M is split to Q' ($Q' > Q$) triangles instead of Q triangles of the original mesh. However, it is possible that not all Q' triangles in the mesh have at least one unknown.

The purpose of the next step is to apply robust least squares adjustment to obtain precise and accurate positions for the unknowns by minimizing the variance of the intensity values in the mesh. To limit the runtime to an acceptable size, we restrict ourselves to a suboptimal solution and do not run least squares adjustment for the whole mesh (more detail in Section 5). We instead iteratively apply least squares for each part of the mesh that can be seen by each camera from the first camera until the last camera.

As a well-known feature of least squares matching is its rather restricted radius of convergence, we employ a coarse-to-fine strategy. It consists of using image resolutions adapted to the sizes of the triangles by selecting a corresponding level of the image pyramid (cf. Section 4.2). Different levels of densification of the triangles are considered by positioning unknowns according to the shapes of the triangles.

4.6 Robust Least Squares Adjustment

4.6.1 Determination of Observations

The purpose of this part is to determine the observations for the least squares estimation. Denote the number of triangles in mesh M_i that can be seen by camera C_j as q. Following the way described in Section 4.5.1 to obtain internal points and then observations in the images, we split a triangle that contains unknowns at the center of each side of the projected

triangle to generate observations in this triangle. The splitting only stops when the length of each side is below the threshold of 1.4 pixels. We apply this splitting procedure to generate observations for all V triangles in mesh M_i that contain at least one unknown. At the end of this step, we obtain n_T internal points in V triangles corresponding to m unknowns in mesh M_i. Each internal point L_i is projected to k cameras that can see the triangle to obtain k image observations and thus k intensity values $I_{i,j}$ with $j = 1$ to k. The next issue is to estimate the adjusted positions Δx for the m unknowns based on the obtained image observations to minimize the variance of intensity values in the mesh.

4.6.2   Factorization of the Design Matrix

To solve the least squares problem for the unknowns Δx, we must factorize the normal equation matrix $A^T PA$, with the design matrix A and the weight matrix P. As there might be thousands or even tens ("or hundredth") of thousands of unknowns, the factorization requires special attention. We apply the following procedure:

In the design matrix A an unknown is affected only by the observations belonging to neighboring triangles leading to a very sparse matrix. We make use of this by only computing those parts which are non-zero.

This implies that (only) unknowns are correlated which have common triangles. To obtain a banded normal equation matrix with a band-width as small as possible, for which efficient solutions are available, we traverse the triangles along the shorter side of the given area. For the example in Figure 4.16 this leads to the banded normal equation matrix sketched in Figure 4.17. One can regard the first unknowns to belong to the triangles marked in red in the lower left corner of the triangles in Figure 4.16, the next unknowns to the triangles marked in green right of it, the next to the blue triangles, etc. All vertices of the triangles, i.e., the unknowns, are linked only to two layers of triangles. This procedure leads to a normal equation matrix with just one band parallel to the main diagonal with the width of the band depending on the length of the layer. Thus, triangulation is traversed along the shorter side. We will present this procedure in detail in Chapter 5.

We thus obtain a positive definite band matrix. Using Cholesky factorization for banded symmetric matrices we solve for Δx.

To stabilize the solution, the Levenberg-Marquardt algorithm is employed. I.e., we multiply the elements on the main diagonal with a factor ranging from 1.0001 to 1.1 and take the result with the smallest average standard deviation σ_0. The non-linear optimization is iterated until the ratio between the σ_0 determined for two consecutive iterations falls below 1.01. We will specify our robust least squares model in detail in Chapter 5.

After the above procedure is applied for all meshes seen by all cameras, we obtain precise and accurate positions for additional 3D points (unknowns). From these new 3D points together with the given 3D points, we obtain a densely reconstructed 3D surface. In the next chapter, we present experiments and results for densely reconstructed 3D surfaces conducted with and obtained by the presented approach.
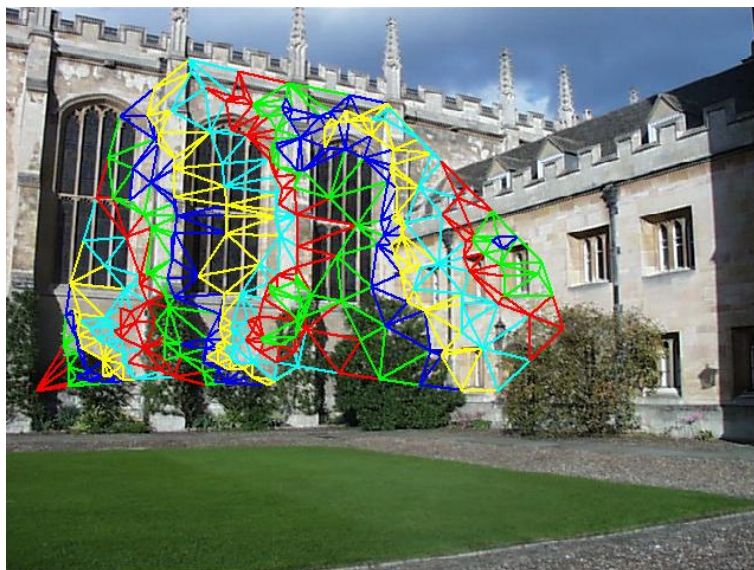
Figure 4.16: Traversal of triangles along the shorter side. The different colors correspond to different layers of the traversal. The traversal starts in the lower left corner (red triangles) – image Trinity from web-page Criminisi and Torr.
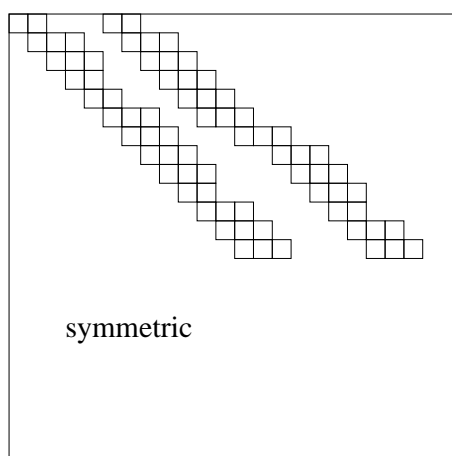

Figure 4.17: Banded structure of normal equation matrix resulting from traversal of the triangulation in Figure 4.16 along the shorter side.

Chapter 5

Experiments

This Chapter presents experiments with our approach. As discussed in Chapter 4, our approach consists of three steps: (i) Construction of the initial triangulated meshes using given sparse 3D points extracted from images, (ii) determination of the initial positions for additional 3D points (unknowns) in the grids, and (iii) application of robust least squares adjustment to adjust the positions of the unknowns to obtain precise, accurate, and dense 3D points.

In the first section we particularly discuss all practical issues of the approach. We describe how we solve problems related to the given 3D points, the selection of triangles for unknowns, the initial positions for unknowns, the creation of layers in the triangle meshes to speed up the calculation, and parameters for least squares adjustment. In the second section, we describe the images and scenes that we used and discuss the results that we have obtained by applying the approach.

5.1 Preparations

5.1.1 Generation of the 3D Input Data

We use given sparse 3D points extracted from images as input (cf. Section 4.1). Although these points are mostly precise and accurate, there are two issues that can affect our approach and thus need to be dealt with:

First, it can happen that points are matched incorrectly even in three or more images particularly due to repeating structures. This leads to incorrect 3D points which we delete from our input data set. Particularly, we first apply Delaunay triangulation to construct the triangle meshes from the 3D given points and then apply a VRML (Virtual Reality Modeling Language) viewer to find out which points are wrong and need to be deleted from the data set.

Second, particularly due to noise and occlusions 3D points might not be matched in all images where they are actually visible. This definitely affects our approach as we use the

information which camera can see which triangle when calculating the variances of the intensity values of the 3D points. To account for this, we manually change in which cameras the 3D points are visible in the data set to reflect the reality.

5.1.2   Selection of Triangles for and Positioning of Unknowns

The purpose of our approach is to determine additional precise and accurate 3D points (unknowns) on top of the given sparse 3D points in the triangulated meshes to obtain a more precisely reconstructed surface. The first task towards this goal is to derive the initial positions for these unknown points. For this, we need to select the triangles for which unknowns are to be added. We have found empirically that the selected triangles should be in the top 15% concerning the average deviation in intensity, because usually only 10 to 15% of the triangles have high average deviations. The remaining 85% of the triangles have quite low average deviations. We will show the distribution of average deviations of triangles for each reconstructed object in the result section to prove our choice.

After selecting triangles for adding unknowns, we need to determine the initial positions for the unknowns in those triangles. In Section 4.5.3 it was explained how unknowns are inserted in a triangle, i.e., how a triangle is split, depends on the shape and size of the triangle. Our rule is that we only add unknowns to triangles that have at least one side longer than 14 pixels. This rule is also empirically derived from our experiments which show that smaller triangles don't have enough observations in them to reliably determine the positions of the unknowns. More importantly, we usually also have quite a small average intensity variance in such triangles. I.e., these triangles are quite homogeneous and thus it is useless to split them. Besides putting a threshold on the side length, we also control the shape of the triangle: If the angle between two sides which are longer than 14 pixels is smaller than 60 degrees, the side opposite to this angle will not be split (cf. Figure 4.13(b)). This rule avoids to obtain triangles with very small angles.

5.1.3   Creation of Layers to Reduce Runtime

To determine precise and accurate positions from the initial positions of the unknowns, we apply robust least squares adjustment. However, it would be very time consuming if we would run the estimation for the whole mesh seen by all cameras. To reduce the calculation time, we conduct the estimation for each part of the mesh that can be seen by each camera from camera C_0 until camera C_K separately. Particularly, while we adjust all meshes which can be seen from C_0, for C_1 to C_k we adjust only those meshes which can be seen from the respective camera and have not been adjusted before. Thus, we replace global estimation with a large runtime against only regionally consistent estimations with a lower runtime. In addition, we also divide each sub-mesh into layers (cf. Section 4.6.2).

Particularly, we divide a sub-mesh (e.g., M_i) into layers from left to right. We chose this direction, because our images are wider than high. By projecting the 3D mesh M_i to the

camera, we obtain a 2D mesh. For the first layer, the left most vertex of the 2D mesh is selected as the first point. This point is connected with its adjacent vertices on the border of the mesh. However, only vertices with angles between the connecting line and the horizontal axis of larger than 60 degrees are selected for the first layer. After all adjacent vertices to the first point have been checked and selected, the first layer is obtained to the left of (cf. red part in Figure 4.16). The second layer includes all vertices that are adjacent to vertices in the first layer. This procedure continues until the whole mesh is divided into layers. Least squares estimation is then conducted on the layer basis.

5.1.4   Specifications for Robust Least Squares Adjustment

In this section, we describe in detail specifications for the robust least squares adjustment. We also present how we obtain the bias parameters β and weights P for the estimation.

Model Specifications

We perform the estimation on a layer and triangle basis. A triangle in the first layer is selected and internal points and unknowns in this triangle are projected to the cameras that can see this triangle to calculate the intensity values for the projected image observations. After this, the second triangle in the mesh is selected and so on. We choose the triangle to be handled in each layer at each time at random and the internal points lying on the border of two triangles or also unknowns which are vertices of more than one triangle are handled together with the triangle that is selected first.

Denote the number of internal points and unknowns that are calculated in a triangle v as n_v and m_v with $v = 1$ to V, i.e., the number of triangles in the mesh that contain unknowns. The total number of all internal points $\sum n_v$ and all unknowns $\sum m_v$ equal the total number of internal points n_T and unknowns m obtained for mesh M_i (cf. Section 4.6.1). Denote the number of cameras that can see triangle v as k_v. By this specification, we have a total of $\sum_{v=1}^{V} n_v \times k_v$ elements of intensity values $I_{v,i,j}$ in the observation vector L with $v = 1$ to V, $i = 1$ to n_v, and $j = 1$ to k_v.

Following the general least squares model (cf. equation (2.5)), our model is $l + \epsilon = A\Delta x$. l consists of the intensity values of image observation $I_{v,i,j}$ with i representing a 3D internal point L_i in triangle v and j representing a camera that can see triangle v. In our model, the initial values for the unknown parameters X_0 (cf. $\Delta x = X - X_0$) are the initial values of the 3D unknown points, determined by interpolation in the triangle.

Concerning the observations we have to take into account that the images have different average intensities to obtain an unbiased result for the estimation. In addition, we also need to account for possible occlusions. Thus, we insert a bias parameter β for each image and a weight P for each image observation into the general least squares model and apply robust least squares adjustment. This leads to vector β consisting of k_v elements β_j and vector P of $\sum_{v=1}^{V} n_v \times k_v$ elements $(p_{v,i,j})$.

With the effect of bias parameters and weights, the residual value $\epsilon_{v,i,j}$ for each observed value $I_{v,i,j}$ is then calculated as:

$$\epsilon_{v,i,j} = I_{v,i,j} - \beta_j - I_{v,i,M}, \tag{5.1}$$

with $I_{v,i,M}$ the mean intensity value for 3D point i in triangle v seen by k_v cameras:

$$I_{v,i,M} = \frac{\sum_{j=1}^{k_v} (I_{v,i,j} - \beta_j) p_{v,i,j}}{\sum_{j=1}^{k_v} p_{v,i,j}}. \tag{5.2}$$

The objective of the robust least squares adjustment is to minimize the variance of the residuals:

$$\sigma = \sqrt{\frac{\epsilon^T P \epsilon}{\sum_{v=1}^{V} k_v \cdot n_v - m_v}}. \tag{5.3}$$

In the next paragraph, we present how we set the values for β and P.

Bias Parameters and Weights for Robust Optimization

For the bias parameters, we use the values calculated in Section 4.5.2. P is a diagonal matrix and is always normalized to unity before being multiplied with the design matrix or the vector of the internal points L. Initially all weights are set to one. This means that all observations have the same weight regardless of whether there is occlusion or non-Lambertian reflection.

To account for this, we use robust estimation. We particularly base robust estimation on standardized residuals $\epsilon_i = \bar{\epsilon}_i / \sigma_{\epsilon_i}$ involving the standard deviations σ_{ϵ_i} of the residuals $\bar{\epsilon}_i$. As the computation of σ_{ϵ_i} for the individual observation is computationally costly, we substitute it by an estimate of the average standard deviation of the intensity value due to noise, particularly 3 gray values. We then follow (McGlone et al. 2004) to reweight the elements of P with $p_{v,i,j} = 1 / \sqrt{2 + \epsilon_{v,i,j}^2}$.

We stop least squares estimation either when there is no error reduction, i.e., the error in the current estimation is equal or higher than the error in the last estimation, or after the 10th iteration. The latter condition is based on our experiments.

5.2   Results

In this section, we present experiments for four different scenes: (i) A scene with different surface structures, but with weak texture, (ii) a scene with a single surface structure type (cylinder) with strong texture, (iii) a scene with different structures, occlusions, and weak texture, and (iv) a scene with a complex surface structure taken by cameras from larger

distances. The first scene is a corner of the Trinity college building in Dublin, Ireland taken from the Trinity web site. The building has glass windows and stone columns. In addition, the building is occluded by some bushes in front of its facade and the texture of the building is mostly weak. The 3D reconstruction of this scene is difficult also due to the big contrast between the glass windows and the walls. The slender stone columns in front of the surface also make the reconstruction more challenging in terms of the small width of the columns. The purpose of this scene is to show that our approach works well for a scene with different surface structures and weak texture.

The second scene consists of a cylinder covered by advertisement posters. This advertisement cylinder is situated close to a metro station in Munich, Germany. There is only one surface structure for this scene and the texture of the object is mostly very strong. The main problem in the 3D reconstruction of this scene is that parts are rather homogeneous, i.e., are weakly textured areas, where the image does not give a clear hint about the surface geometry. Thus, with this scene, we want to demonstrate that our approach can reconstruct highly homogeneous areas based on the regularization of the surface.

The third scene is a street corner named Schappenstraat in Leuven, Belgium. Besides some jutting walls, there are static and moving occlusions in the images. In some areas the texture of this object is also very weak. With this sample, we want to show the robustness of our approach concerning occlusions.

The final scene is the main gate of the Cathedral in Cologne, Germany. Besides the fact that the images are taken by cameras from larger distances, the scene has a complicated surface structure with many sophisticated small statues. By means of this scene we want to present the capability of our approach for the 3D reconstruction of sophisticated surface structures.

In the next subsections, we present the result of our experiments for each scene. In the first part of each subsection we give: (1) The source images of the scene, (2) the sparse 3D points extracted from the images by applying the approach of Bartelsen and Mayer (2010), (3) the triangle mesh created from the given 3D points, (4) the graph of (average) intensity deviations for the triangles in the mesh to show how triangles are selected for the insertion of unknowns, (5) the positions of unknowns and their directions, and finally (6) the final result after application of robust least squares adjustment. In the second part, we will compare our finally reconstructed surfaces with the input surface created from the given 3D points to demonstrate to which extent our approach improves the accuracy of the reconstructed surface. Particularly, we compare and discuss the result for different camera view points.

5.2.1   Trinity Corner

Experiment

The building images are taken from three different positions as shown in Figure 5.1. By applying the approach of Bartelsen and Mayer (2010), we obtain 279 sparse 3D points and the information about which camera can see which point (cf. Figure 5.2). We only use 223

points to create a triangulated mesh consisting of 421 triangles via Delaunay triangulation (cf. Figure 5.3).
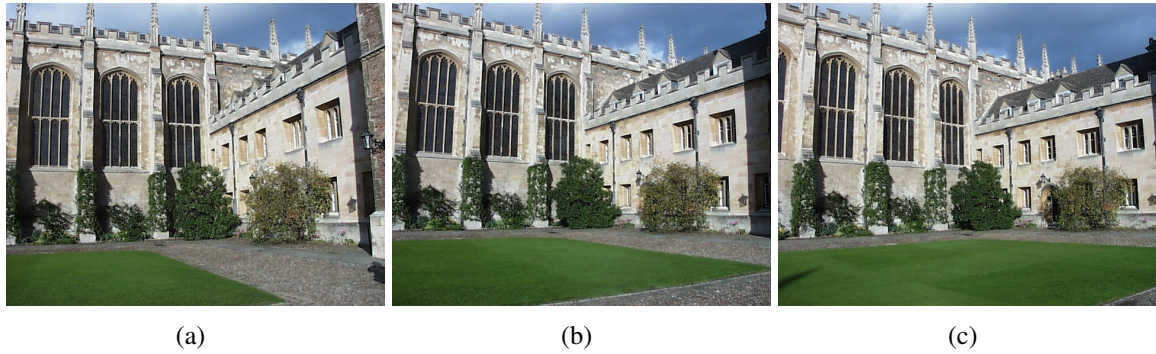

| (a) | (b) | (c) |

Figure 5.1: Three images of Trinity corner.

Figure 5.2: The three camera positions (green pyramids) and the 279 sparse 3D points.

To position additional 3D points (unknowns) in this mesh, we first determined in which triangles in the mesh unknowns should be inserted. According to Section 5.1, we base this on the (average) deviations of the intensity values in the triangles. For this scene, we use four pyramid levels (from 0 to 3). As the difference for the intermediate levels are minors, we will show only how we choose unknowns for pyramid levels 0 and 3.

Unknowns for pyramid level 3

Projecting the mesh into pyramid level 3, we obtained an ordered list of the (average) intensity value deviations calculated by equation (4.6) for the 21 triangles in the mesh. Figure 5.4 clearly shows that the deviations of the intensity values decrease significantly for the first 15% of triangles and then are mostly constant. We thus select the corresponding 15% triangles to split and to obtain a lower deviation by means of robust adjustment. Applying the procedure described in Section 4.5.3, we obtained only two unknowns in the mesh. The positions and directions of unknowns are marked in yellow in Figure 5.5.
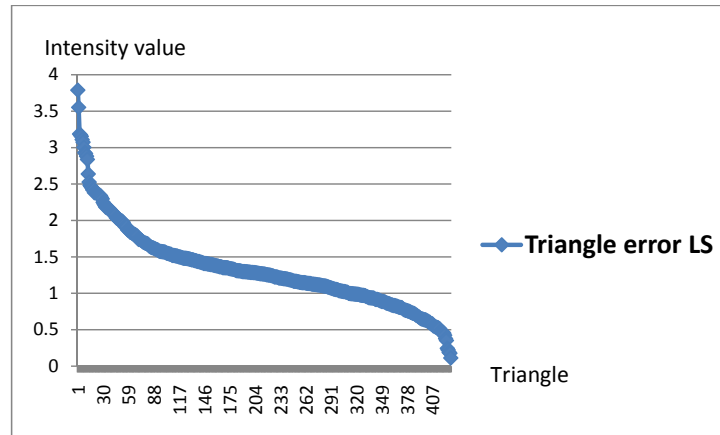
Figure 5.3: Triangulated mesh created from 223 points.
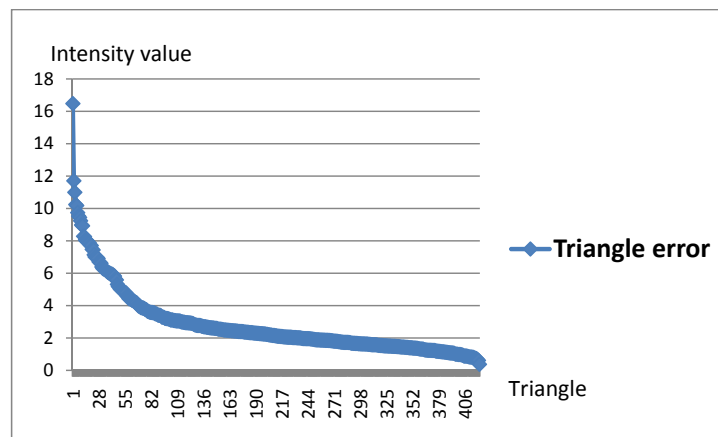
Figure 5.4: Plot of intensity deviations (calculated by equation (4.6)) for the triangles in the mesh at pyramid level 3 sorted in descending order.

Figure 5.5: Positions and directions of two unknowns in the mesh marked in yellow.

After applying least squares adjustment, the (average) deviations of the intensity values for the triangles in the new mesh with the two additional 3D points are plotted in Figure 5.6. By

comparing Figures 5.4 and 5.6 we find that there is an improvement in the deviations of the intensity values, i.e., the deviations are reduced. That only two unknowns are positioned is because of: (1) on pyramid level 3 the triangles are very small, and (2) we use a constraint on the lengths of the sides of the triangles to be split, i.e., only triangles that have at least one side longer than 14 pixels can be split.

Figure 5.6: Plot of intensity deviations (calculated by equation (4.6)) for the triangles in the mesh at pyramid level 3 with two additional 3D points sorted in descending order.

Unknowns for pyramid level 0

When we changed to pyramid levels 2 and 1, the number of selected unknowns had increased and the deviations have improved as the sizes of triangles become bigger. We present details of the result, particularly the number of unknowns and the deviations after each pyramid level, in the following part. After pyramid level 1, a mesh with 756 triangles with 400 vertices was created. Figure 5.7 presents the graph of the (average) intensity value deviations of all triangles in the initial mesh for pyramid level 0. It again clearly shows that the (average) deviations of the intensity values decrease significantly for the first 15% of triangles.
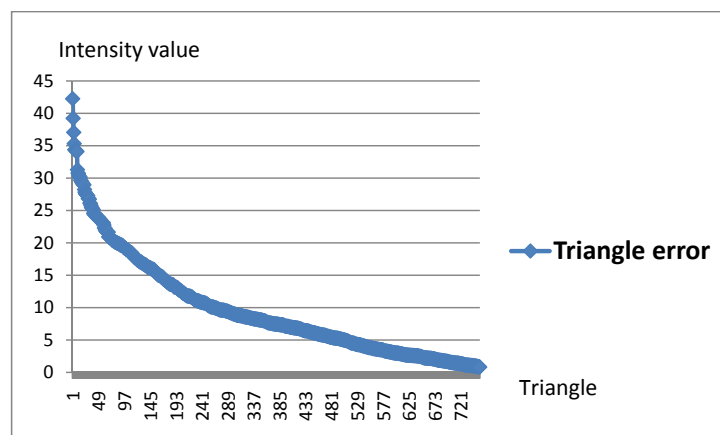

Figure 5.7: Plot of intensity deviations (calculated by equation (4.6)) for the triangles in the mesh at pyramid level 0 sorted by descending order.

Splitting these 15% triangles, we obtain 385 unknowns as shown in Figure 5.8. The posi-

tions and directions of the unknowns are marked in yellow. The (average) deviations of the intensity values for the triangles in the new mesh after least squares adjustment process is finally displayed in Figure 5.9. The two figures (Figures 5.7 and 5.9) show that the (average) deviations of the intensity values have been reduced significantly after splitting and robust estimation.

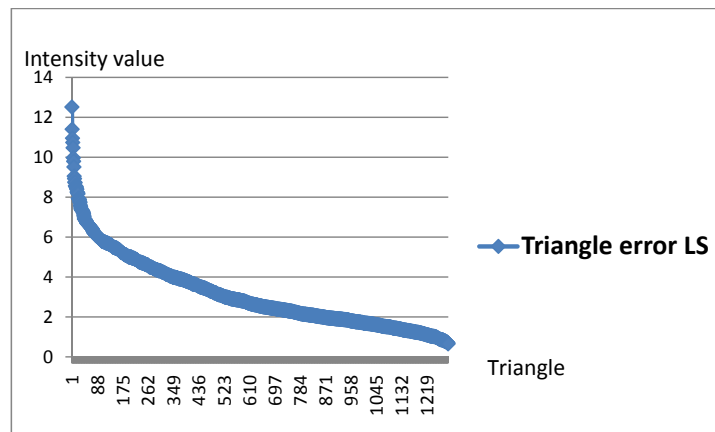Figure 5.8: Positions and directions of 385 unknowns marked in yellow at pyramid level 0.

Figure 5.9: Plot of intensity deviations (calculated by equation (4.6)) for triangles in the mesh at pyramid level 0 with 385 additional 3D points sorted in descending order.

Summary of results for four pyramid levels

To show how unknowns were created and how the (average) deviations were reduced in the process, in Table 5.1 we present a summary of the creation of unknowns and the (average) deviations of all triangles on each pyramid level.

The table shows that the creation of unknowns helps to reduce the average deviations of the intensity values in the mesh. After all four pyramid levels, a total of 573 unknowns has been inserted and positioned. Using these 573 additional points together with the 223 given 3D points we can reconstruct the 3D surface of Trinity building as shown in Figure 5.10. In the

Table 5.1: Summary statistics of unknown creation on each pyramid level for Trinity corner.

Pyramid level	No. Triangles	No. Vertices	No. Unknowns	Avg. Deviation
Level 3				
- Begin	421	223		6.7
- End	428	228	2	1.2
Level 2				
- Begin	428	228		7.3
- End	496	265	33	1.5
Level 1				
- Begin	496	265		8.5
- End	756	400	153	1.7
Level 0				
- Begin	756	400		10.6
- End	1293	673	385	1.9

next part, we will compare our reconstructed 3D surface with the original (input) 3D surface (from given 3D points) for different camera views.

Figure 5.10: The final reconstructed 3D surface of Trinity corner.

Discussion of results

When comparing the original 3D surface created from the given 3D points (cf. Figure 5.3) and the surface constructed from the given 3D points together with the unknowns (cf. Figure 5.10), we find that the latter represents the structure of the building much better than the former. Specifically, we can observe clear improvements in the depth of the stone columns, and the bushes as well as in the much clearer structure of the roof on the right part of the building. Moreover, we also observe that the mullions and transforms of windows as well as the stone columns are straight in the reconstructed surface while they are bent for the original surface. All this becomes even more clear in Figure 5.11, where we present the

original surface (from given 3D points) on the right and the reconstructed surface (from given 3D points and unknowns) on the left for two different views.

(a) Original view 1

(b) Reconstructed view 1

(c) Original view 2
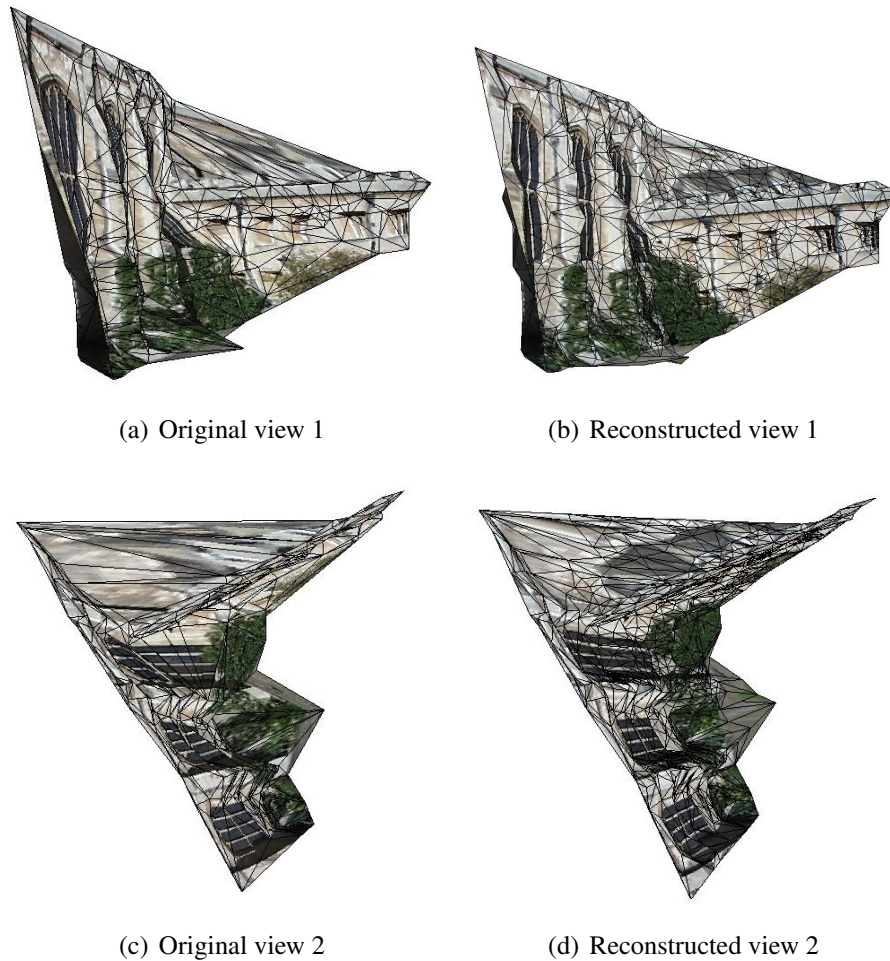

(d) Reconstructed view 2

Figure 5.11: Comparison between the original surface (from given 3D points) and the reconstructed surface (from 3D points and unknowns) for two different views.

Looking at the bushes in the first view (upper row), we find that the original surface does not clearly reflect their structure. The sizes of the original triangles in this area are just too big. On the reconstructed surface, this area is split into smaller triangles which results into a clearer structure for the bushes. This means that the average deviations of the intensity values, i.e, the errors in intensity values, of the triangles in this part of the mesh were initially high, and that the additional 3D points led to have a more precise reconstruction. Moreover, the structure of the roof on top of the right side of the building is also very clear in the reconstructed surface while it is very vague in the original surface.

In the second view, we cannot recognize a clear straight structure for the stone columns in the original surface and it seems that they are almost flat. However, in the reconstructed surface, many additional points were positioned in the area of the columns bringing out the depth of these columns. This all shows that our algorithm is suitable and works well for such type of scene.

5.2.2   Advertisement Cylinder

Experiment

The cylinder was acquired from 25 camera positions which gives us 25 images. Here, only three images are given as examples in Figure 5.12. From the 25 images, we extracted 9568 sparse points (cf. Figure 5.13). However, we only used 63 points as input leading to the triangulated mesh of 92 triangles in Figure 5.14. This was done to show that a small number of given points is sufficient for smooth surfaces.
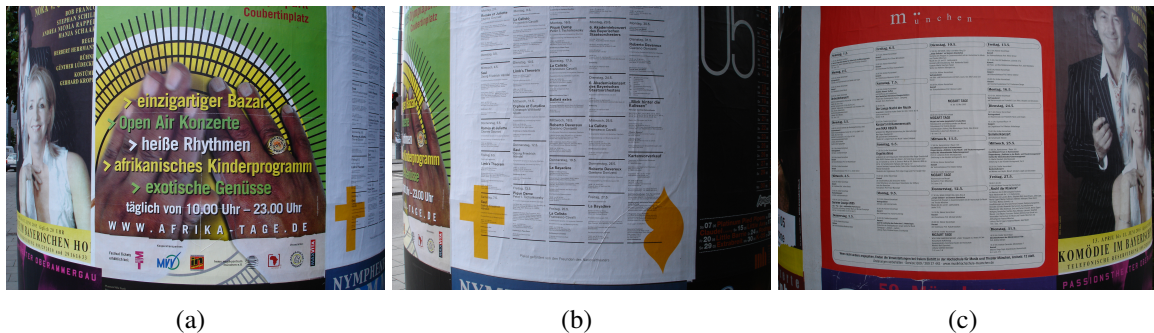

(a)   (b)   (c)

Figure 5.12: Three images of 25 for the advertisement cylinder.
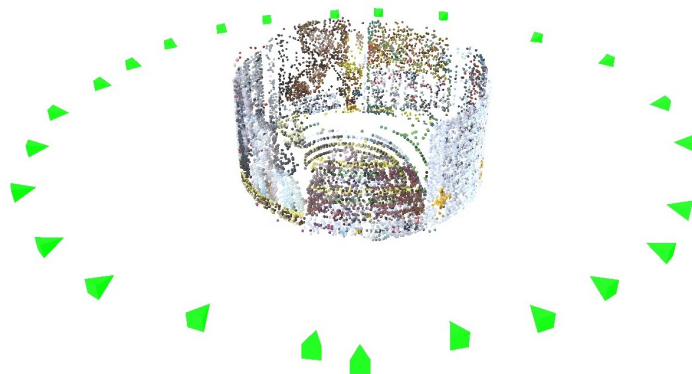
Figure 5.13: The 25 camera positions (green pyramids) and the 9568 sparse 3D points.

For this scene, five pyramid levels (0 to 4) have been used. We will first show how unknowns were chosen for pyramid levels 0 and 4 to give more insight how our algorithm works.

Unknowns for pyramid level 4

Projecting the mesh into pyramid level 4, we obtained an ordered list of the (average) intensity value deviations for the 92 triangles in the mesh. Figure 5.15 again clearly shows that the deviations of the intensity values drop significantly for the first 15% of triangles. When trying to split these 15% triangles, we only obtained the seven unknowns marked in yellow in Figure 5.16.

Figure 5.17 shows the average deviations of the intensity values for the triangles in the new mesh with seven additional 3D points after least squares adjustment. Comparing Figures

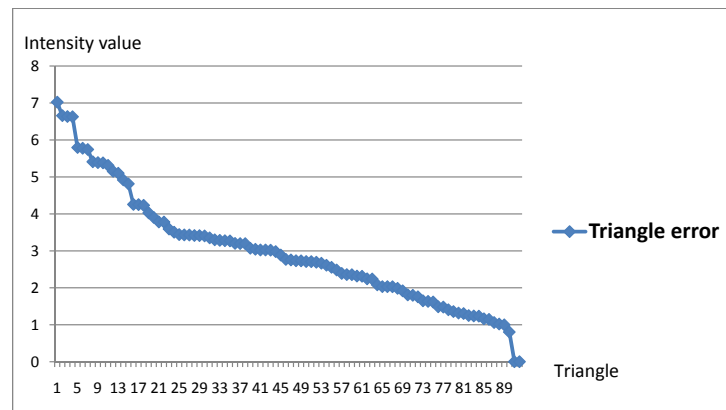Figure 5.14: Triangulated mesh created from 63 points.

Figure 5.15: Plot of intensity deviations for the triangles in the mesh at pyramid level 4 sorted in descending order.

Figure 5.16: Positions and directions of seven unknowns marked in yellow in the mesh.

5.15 and 5.17 we again find that there is a considerable improvement in the deviations of the intensity values.
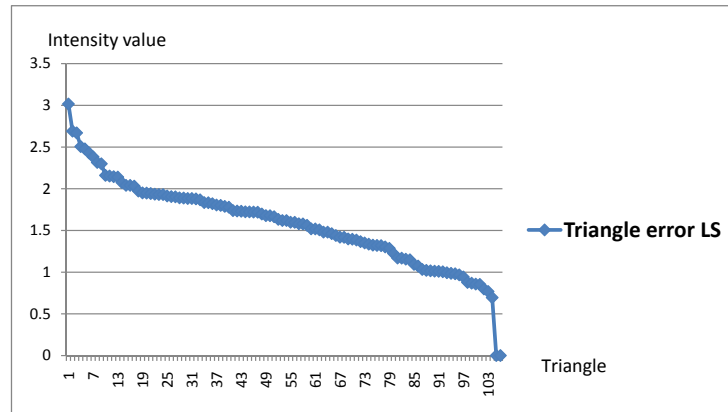
Unknowns for pyramid level 0

Figure 5.17: Plot of intensity deviations for the triangles in the mesh at pyramid level 4 with seven additional 3D points sorted in descending order.

After pyramid level 1, we had obtained a mesh with 309 triangles. No figure for the (average) deviations of the intensity values is shown here because we observed the same situation as for other scene and pyramid levels: 15% triangles have high deviations. On this pyramid level, we have determined and positioned 112 unknowns marked in yellow in Figure 5.18 by splitting the corresponding triangles and obtained a new mesh consisting of 470 triangles.

Figure 5.18: Positions and directions of 112 unknowns in the mesh marked in yellow at pyramid level 0.

Comparing the average deviations of the intensity values for the triangles in the new mesh after least squares adjustment process with the average deviation values before splitting we also found that they have been significantly reduced.

Summary of results for five pyramid levels

To show how unknowns were created and how the average deviations were reduced in the process, we present in Table 5.2 a summary of unknown creation and the average deviations of the triangles on each pyramid level.

Table 5.2: Summary statistics of unknown creation on each pyramid level for advertisement cylinder.

Pyramid level	No. Triangles	No. Vertices	No. Unknowns	Avg. Deviation
Level 4				
- Begin	92	63		9.3
- End	106	70	7	1.6
Level 3				
- Begin	106	70		9.2
- End	149	93	22	1.5
Level 2				
- Begin	149	93		9.3
- End	206	125	33	1.8
Level 1				
- Begin	206	125		9.6
- End	309	182	71	1.4
Level 0				
- Begin	309	182		9.7
- End	470	269	112	1.5

The table again shows that the creation of unknowns helped to reduce the average deviations of the intensity values in the mesh. After all five pyramid levels, a total of 245 unknowns were determined and positioned. Using these 245 additional points together with the 63 given 3D points leads to the reconstruction of the 3D surface of the cylinder shown in Figure 5.19. In the next part, we will compare our reconstructed 3D surface with the original (input) 3D surface (from given 3D points) for different camera views.

Discussion of results

First, when comparing the original 3D surface created from the given 3D points (cf. Figure 5.14) and the surface constructed from the given 3D points together with the unknowns (cf. Figure 5.19), it is evident, that the latter represents the cylinder substantially better than the former. For the original surface, the middle part of the cylinder seems to be flat. However, the curvature of this part can clearly be observed for the reconstructed surface. For direct comparison, we display in Figure 5.20 the original surface on the right and the reconstructed surface on the left for two different views.

The surfaces in the first view indicate that there were some areas with large intensity error in the mesh of the original surface, which have been split into smaller triangles to obtain lower intensity errors and a better matching. Additionally, the figure shows that the additional points also result in a more plausible curvature for the surface.
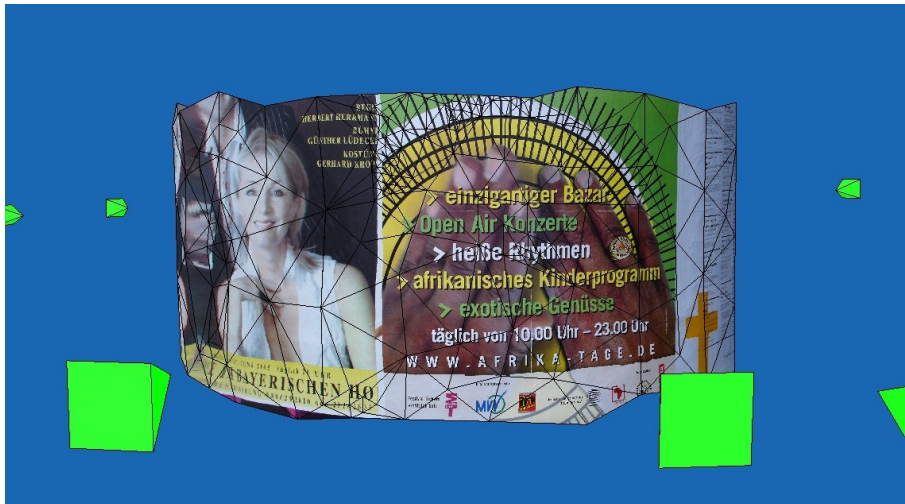
Figure 5.19: Reconstructed 3D surface of the advertisement cylinder.

(a) Original view 1

(b) Reconstructed view 1

(c) Original view 2
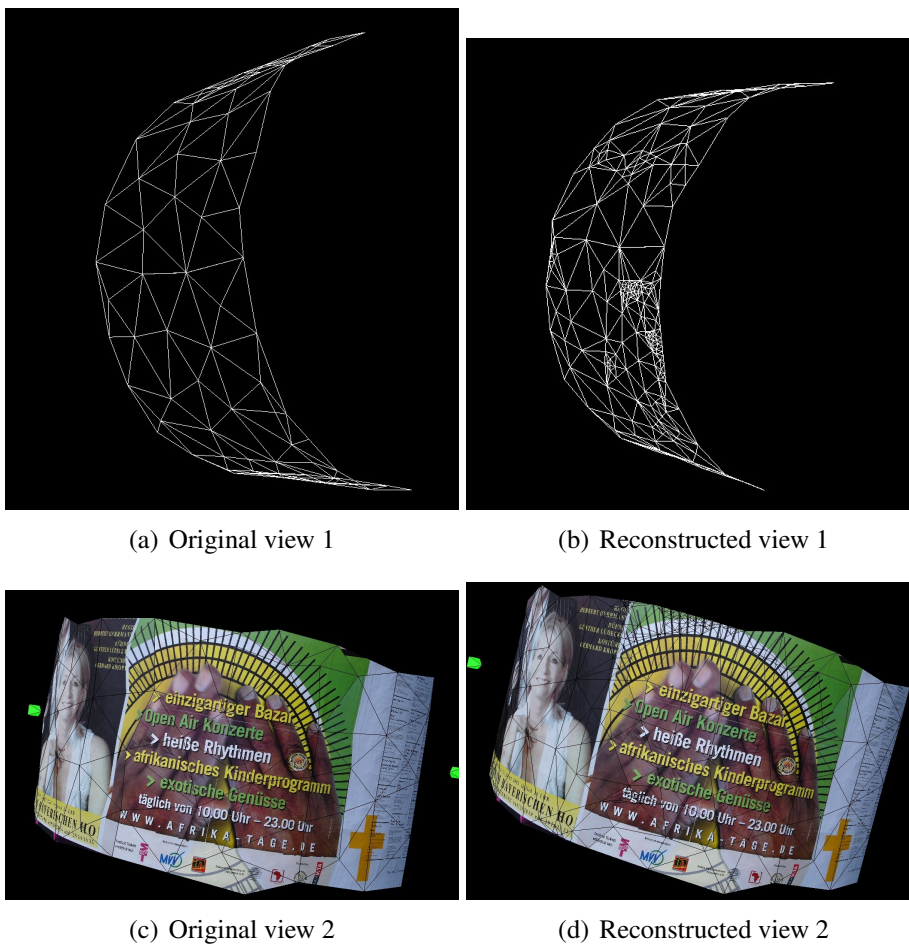

(d) Reconstructed view 2

Figure 5.20: Comparison between the original surface (from given 3D points) and the reconstructed surface (from 3D points and unknowns) for two different views (first view as wire frame).

In the second view of the original surface, particularly two issues are visible: First, the middle part of the surface seems to be flat. Second, the border lines between the posters are not straight. Yet, the view on the reconstructed surface shows that these two problems have been solved. Thus the comparison again confirms the effectiveness of our algorithm in reconstructing the 3D surface for such a type of scene.

5.2.3 Schappenstraat Corner

Experiment

Our third scene consists of four images of Schappenstraat corner in Leuven, Belgium (cf. Figure 5.21). Close to or at the bottom of all images a small pillar to lock out cars from the street is visible. This pillar creates a static occlusion in the images. In addition, there are also people, i.e., moving occlusions in image (a). From the four images we extracted a sparse set of 1606 points (cf. Figure 5.22). However, we only use 786 points on the corner of the building leading to a triangulated mesh of 1505 triangles (cf. Figure 5.23).

(a)   (b)

(c)   (d)

Figure 5.21: Four images of Schappenstraat corner.

For this scene, four pyramid levels (0 to 3) have been used. However, at pyramid level 3, the triangles are too small and thus no unknowns are created. Therefore, we will show how unknowns were chosen for pyramid levels 0 and 2 to give further insight into our algorithm.

Unknowns for pyramid level 2

Projecting the mesh into pyramid level 2, we obtained an ordered list of the (average) intensity value deviations for the 1505 triangles in the mesh. Figure 5.24 clearly shows that the deviations of the intensity values drop significantly for the first 15% of triangles. When
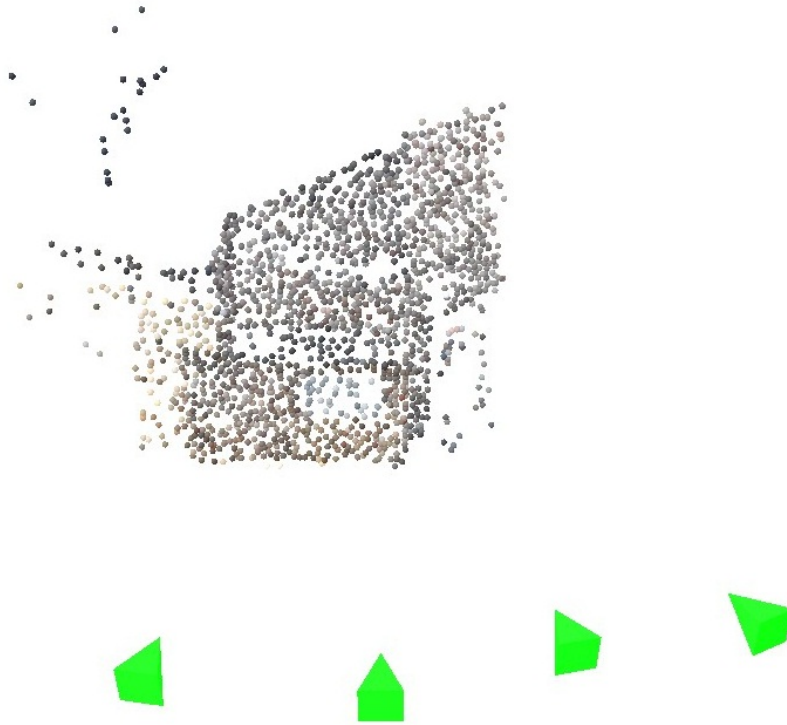
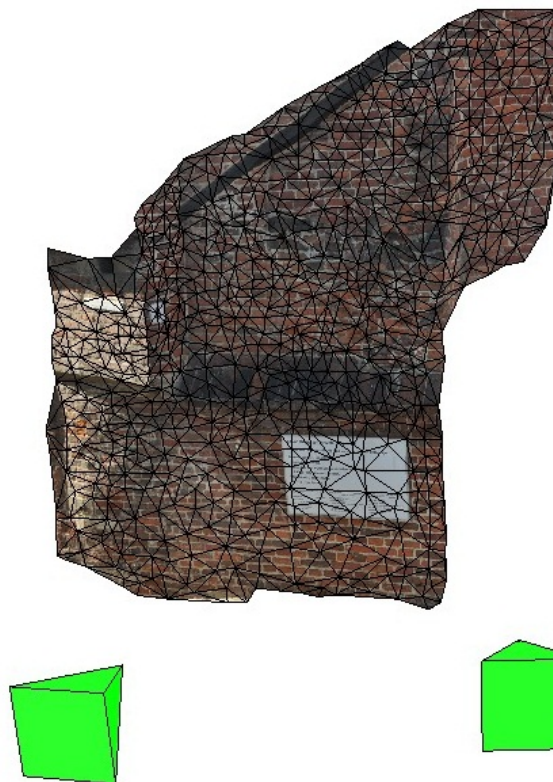Figure 5.22: The four camera positions (green pyramids) and the 1606 sparse 3D points.

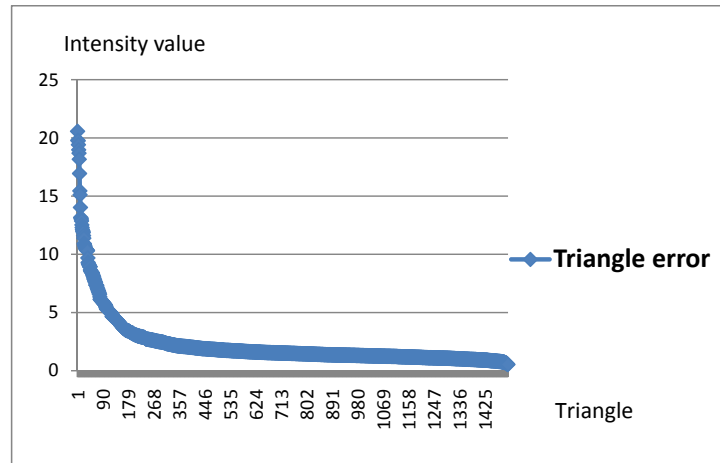Figure 5.23: Triangulated mesh created from 786 points.

Figure 5.24: Plot of intensity deviations for the triangles in the mesh at pyramid level 2 sorted in descending order.

Figure 5.25: Positions and directions of eleven unknowns in the mesh marked in yellow.

trying to split these 15% triangles, we only obtained eleven unknowns marked in yellow in Figure 5.25.

Figure 5.26 shows the (average) deviations of the intensity values for the triangles in the new mesh with eleven additional 3D points after least squares adjustment. Comparing Figures 5.24 and 5.26 we again find that there is a significant improvement in the deviations of the intensity values.
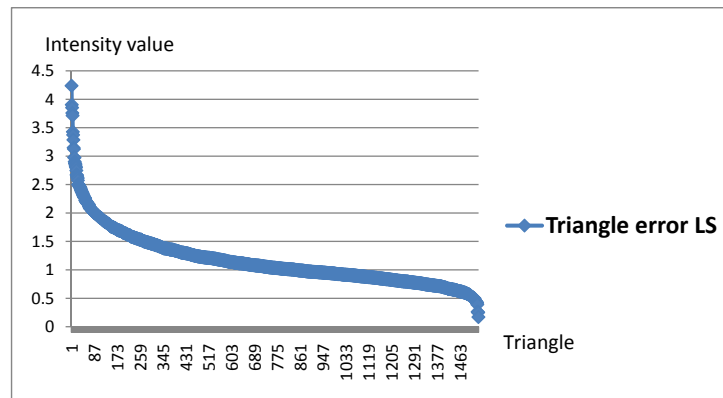
Unknowns for pyramid level 0

Figure 5.26: Plot of intensity deviations for the triangles in the mesh at pyramid level 2 with eleven additional 3D points sorted in descending order.

After pyramid level 1, we had obtained a mesh with 2050 triangles. The ordered list of the (average) intensity value deviations given in Figure 5.27 clearly demonstrates that the deviations of the intensity values are largely descending for the first 15% of the triangles. By splitting these 15% triangles, 1117 unknowns marked in yellow in Figure 5.28 were selected and positioned and a new mesh consisting of 3870 triangles was obtained.
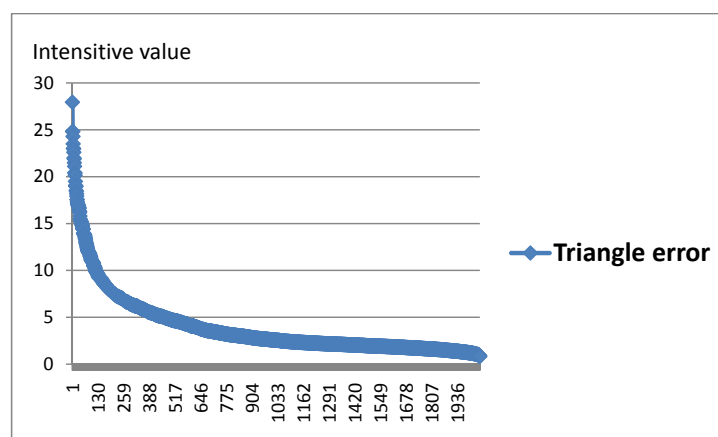

Figure 5.27: Plot of intensity deviations for the triangles in the mesh at pyramid level 0 sorted in descending order.

The (average) deviations of the intensity values for the triangles in the new mesh after least squares adjustment is displayed in Figure 5.29. The comparison of Figures 5.27 and 5.29 clearly demonstrates that the (average) deviations of the intensity values have been reduced significantly.

Summary of results for four pyramid levels

To show how unknowns were created and how the average deviations were reduced in the process, we present in Table 5.3 a summary of unknown creation and the average deviations of the triangles on each pyramid level.

The table shows again that the creation of unknowns helped to reduce the average deviations of the intensity values in the mesh. After all four pyramid levels, a total of 1387 unknowns

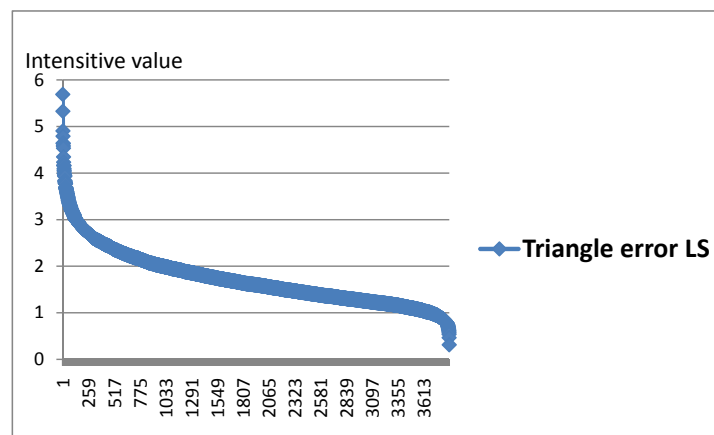Figure 5.28: Positions and directions of 1117 unknowns in the mesh marked in yellow at pyramid level 0.

Figure 5.29: Plot of intensity deviations for the triangles in the mesh at pyramid level 0 with 1117 additional 3D points sorted in descending order.

are selected and positioned. Using these 1387 additional points together with the 786 given 3D points leads to the reconstruction of the 3D surface of the corner presented in Figure 5.30. In the next part, we will compare our reconstructed 3D surface with the original (input) 3D surface (from given 3D points) for different camera views.

Table 5.3: Summary statistics of unknown creation on each pyramid level for Schappenstraat corner.

Pyramid level	No. Triangles	No. Vertices	No. Unknowns	Avg. Deviation
Level 3				
- Begin	1505	786		9.4
- End	1505	786	0	9.4
Level 2				
- Begin	1505	786		9.4
- End	1532	802	11	1.6
Level 1				
- Begin	1532	802		10.1
- End	2050	1072	259	1.5
Level 0				
- Begin	2050	1072		10.5
- End	3870	1997	1117	1.6

Figure 5.30: 3D reconstructed surface of Shappenstraat corner.

Discussion of results

Comparing the original 3D surface created from the given 3D points (cf. Figure 5.23) and the surface constructed from the given 3D points together with the unknowns (cf. Figure 5.30), we again find that the latter represents the shape of the corner much better than the former. This is true particularly for the occlusions part. For direct comparison, we display in

Figure 5.31 the original surface on the right and the reconstructed surface on the left for two different views.

(a) Original view 1

(b) Reconstructed view 1

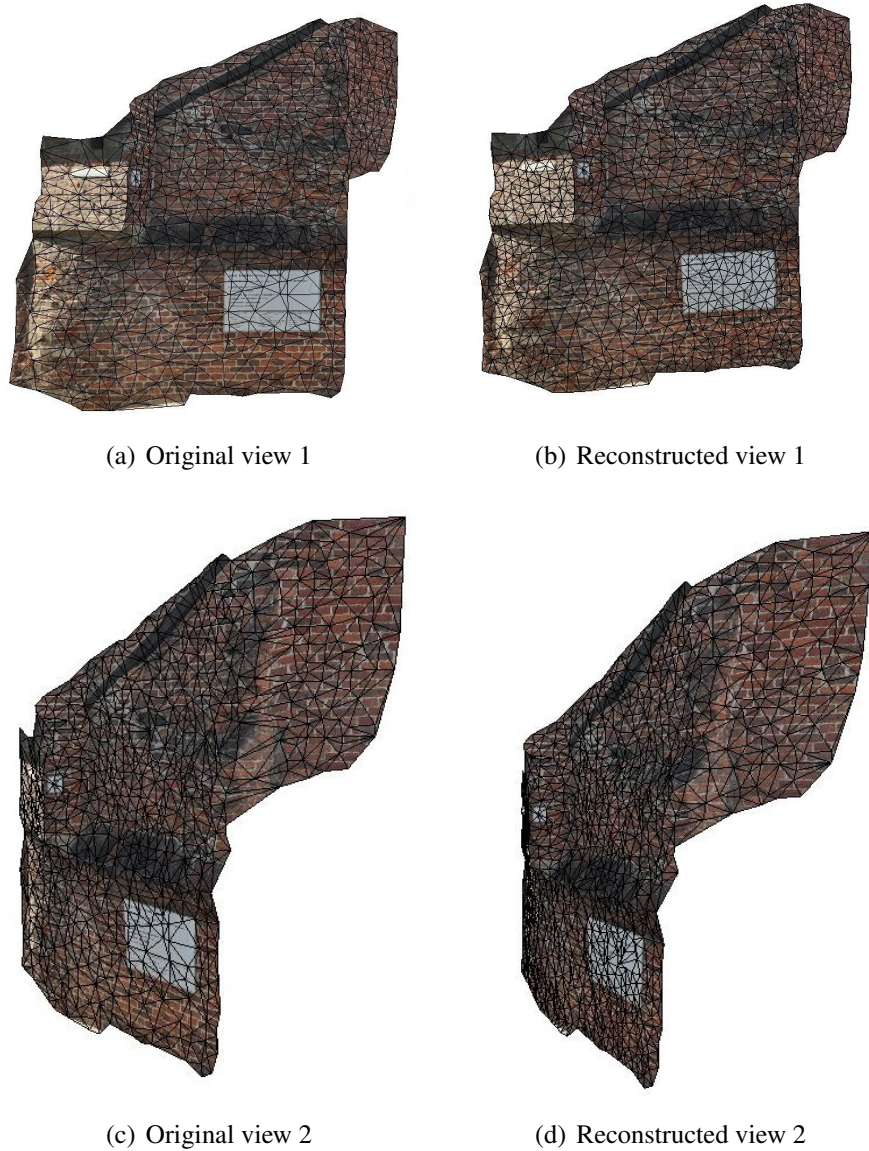(c) Original view 2

(d) Reconstructed view 2

Figure 5.31: Comparison between the original surface (from given 3D points) and the reconstructed surface (from 3D points and unknowns) for two different views.

The surfaces in the first view indicate that there were some areas with large intensity error in the mesh of original surface particularly for the occlusions which have been split into smaller triangles to obtain lower intensity errors and thus a better matching. After splitting, the mesh becomes significantly smoother, especially in the occlusion areas. This indicates that our algorithm has a certain robustness concerning occlusions. The second view shows that the upper part has been split to get a smoother reconstructed surface.

5.2.4   Main Gate of Cologne Cathedral

Experiment

Our fourth scene consists of six images of the main gate of the Cathedral of Cologne, Germany. Four images are presented in Figure 5.32 as examples. From the six images we extracted 3286 sparse 3D points (cf. Figure 5.33). We only use 878 points leading to a triangulated mesh of 1706 triangles (cf. Figure 5.34).

(a)   (b)

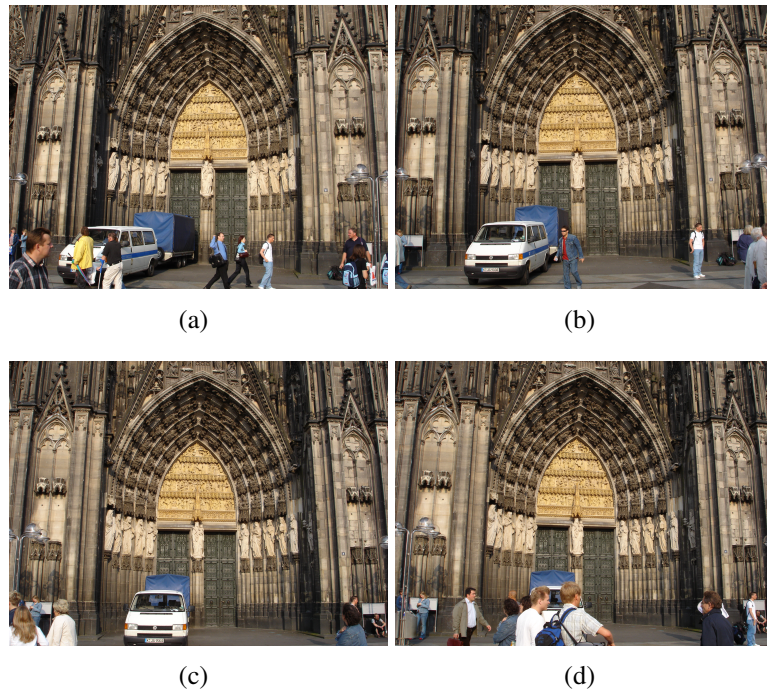(c)   (d)

Figure 5.32: Four images of the main gate of the Cathedral of Cologne.

For this scene, five pyramid levels (0 to 4) have been used. However, at pyramid level 4, the triangles are too small and thus no unknowns are created. Therefore, we will show how unknowns were chosen for pyramid levels 0 and 3 to demonstrate our algorithm.

Unknowns for pyramid level 3

Projecting the mesh into pyramid level 3, we obtained an ordered list of the (average) deviations of the intensity value for the 1706 triangles in the mesh. Figure 5.35 again clearly shows that the deviations of the intensity values drop significantly for the first 15% of triangles. When trying to split these 15% triangles, we obtained only nineteen unknowns marked in yellow in Figure 5.36. Figure 5.37 shows the (average) deviations of the intensity values for the triangles in the new mesh with the nineteen additional 3D points after least squares adjustment. Comparing Figures 5.35 and 5.37 we again notice that there is a significant improvement in the deviations of the intensity values.

Unknowns for pyramid level 0

After pyramid level 1, we had obtained a mesh with 3137 triangles. We observed the same situation as for pyramid level 3: The (average) deviations of the intensity values for the

Figure 5.33: The six camera positions (green pyramids) and the 3286 sparse 3D points.
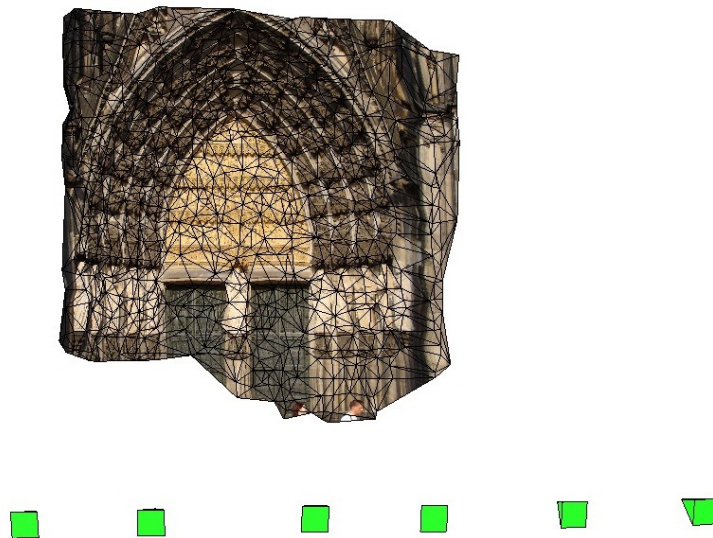

Figure 5.34: Triangulated mesh created from 878 points.

triangles in the new mesh after least squares adjustment have been reduced significantly. By splitting the 15% corresponding triangles, 1442 unknowns marked in yellow in Figure 5.38 were selected and positioned. From these 3D points, we obtained a new mesh consisting of 5007 triangles.

Summary of results for four pyramid levels

To show how unknowns were created and how the average deviations were reduced in the process, we present in Table 5.4 a summary of unknown creation and the average deviations of the triangles on each pyramid level.

The table shows that the creation of unknowns helped to reduce the average deviations of
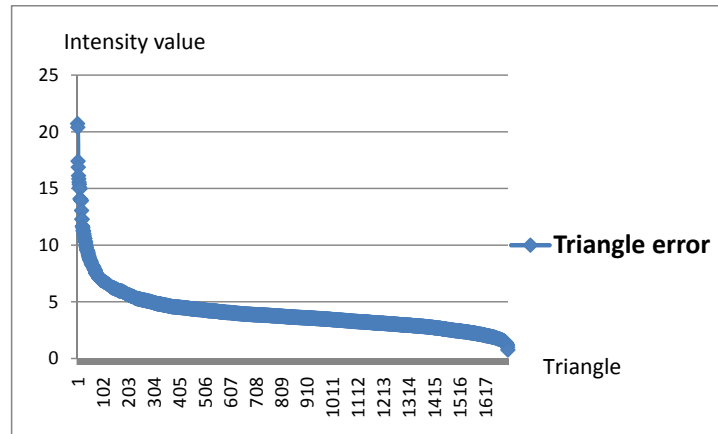
Figure 5.35: Plot of intensity deviations for the triangles in the mesh at pyramid level 3 sorted in descending order.

Figure 5.36: Positions and directions of nineteen unknowns in the mesh marked in yellow.
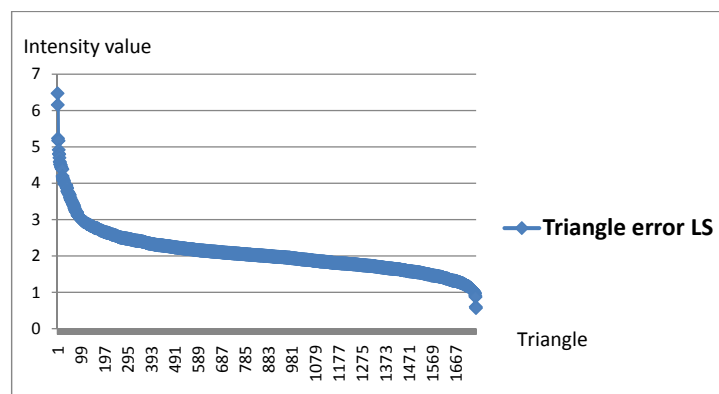

Figure 5.37: Plot of intensity deviations for the triangles in the mesh at pyramid level 3 with nineteen additional 3D points sorted in descending order.

Figure 5.38: Positions and directions of 1442 unknowns in the mesh marked in yellow at pyramid level 0.

Table 5.4: Summary statistics of unknown creation on each pyramid level for Cologne Cathedral.

Pyramid level	No. Triangles	No. Vertices	No. Unknowns	Avg. Deviation
Level 4				
- Begin	1706	878		5.4
- End	1706	878	0	5.4
Level 3				
- Begin	1706	878		5.8
- End	1752	905	19	3.0
Level 2				
- Begin	1752	905		6.6
- End	2078	1076	174	2.6
Level 1				
- Begin	2078	1076		7.7
- End	3137	1611	678	2.8
Level 0				
- Begin	3137	1611		9.0
- End	5007	2549	1442	3.2

the intensity values in the mesh. After all four pyramid levels, a total of 2313 unknowns are selected and positioned. Using these 2313 additional points together with the 878 given 3D points leads to the reconstruction of the 3D surface of the gate presented in Figure 5.39. In the next part, we compare our reconstructed 3D surface with the original (input) 3D surface (from given 3D points) for different camera views.
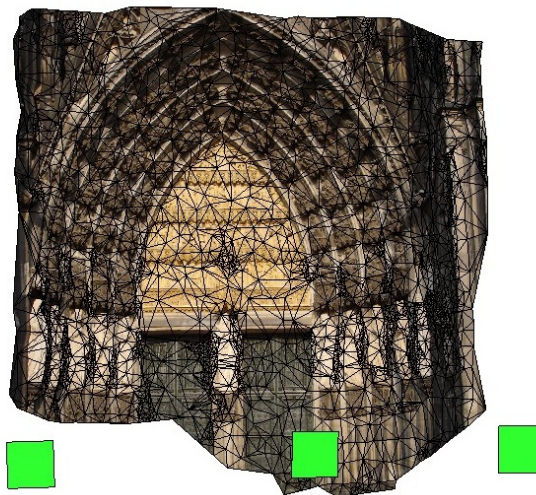
Figure 5.39: 3D reconstructed surface of the main gate of Cologne Cathedral.

Discussion of results

Comparing the original 3D surface created from the given 3D points (cf. Figure 5.34) and the surface constructed from the given 3D points together with the unknowns (cf. Figure 5.39), we find that the latter represents the complicated surface of the gate much better than the former. For direct comparison, we display in Figure 5.40 the original surface on the right and the reconstructed surface on the left for two different views.

The surfaces in the first view indicate that there were some areas with large intensity error in the mesh of the original surface particularly for the areas of the statues which have been split into smaller triangles to obtain lower intensity errors and thus a better matching. After splitting, the mesh becomes significantly more detailed, especially in areas with statues. This indicates that our algorithm can deal also with complicated surfaces. To make it even clearer, in the second view we show the wireframes which highlight the much more detailed triangulation around the statues for the reconstructed grid.

In summary, we have demonstrated that our algorithm is suitable for different types of scenes and can deal with problems due to non-Lambertian reflection and occlusions. To conclude this Chapter, in Table 5.5 we present the runtime when applying our algorithm to reconstruct these four scenes.

(a) Original view 1   (b) Reconstructed view 1

(c) Original view 2   (d) Reconstructed view 2
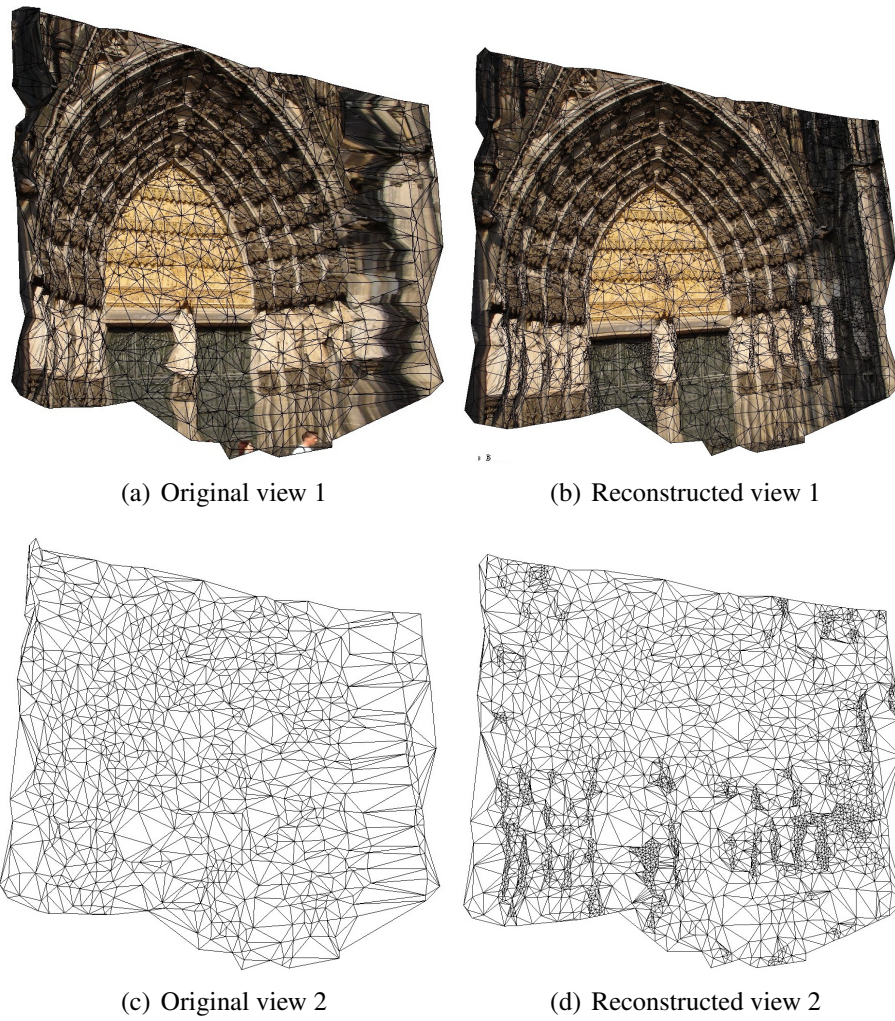
Figure 5.40: Comparison between the original surface (from given 3D points) and the reconstructed surface (from 3D points and unknowns) for two different views.

Table 5.5: Summary of the runtime to reconstruct the four scenes.

Scene	No. images	Size per image *(pixels)*	Runtime *(minutes)*
Trinity Corner	3	640 x 512	25
Advertisement Cylinder	12	2592 x 1944	1560
Schappenstraat Corner	4	2144 x 1424	780
Cologne Cathedral	6	2592 x 1944	1080

Chapter 6

Conclusions and Outlook

This thesis deals with the reconstruction of fully 3D surfaces. The research result is a 3D extension of the classical least-squares matching approach of HEIPKE (1990) which is confined to 2.5D surfaces by employing the normals of a triangulation similarly to SCHLÜTER (1998). Opposed to the latter, our approach focuses on wide-baseline settings and we employ robust estimation to deal with occlusions and non-Lambertian reflection.

The thesis makes two contributions to the field of research: First, advantages of current interesting algorithms are combined and successfully extended to full 3D reconstructions. A mesh representation is the core characteristic of the algorithm. We link triangulations for image triplets to obtain triangulations for a larger number of images. Based on the mesh representation the method can combine different information sources (image- and geometry-based) for the reconstruction, to obtain both the shape and the reflectance properties of complicated surfaces. Second, by making use of robust least squares adjustment, the approach overcomes limitations of other algorithms concerning reflectance properties and the topology of the reconstructed surfaces, particularly non-Lambertian reflection and occlusion.

Our approach first introduces a novel algorithm to select and position additional 3D points in the triangle mesh constructed from given sparse 3D points extracted from different images. Additional 3D points are selected in the mesh, to split it so that the (average) variance of the intensity values in the triangles is reduced. By this way, triangles become more homogeneous and thus the reconstructed surfaces reflect the real shape and texture more precisely. Second, as occlusion and non-Lambertian reflection are two main issues negatively affecting the reconstruction result and the reliability of the algorithm, we insert bias parameters and use robust weights. Finally, we found that we can reduce runtime significantly by applying an image pyramid hierarchy and by approximating the global solution by a regional optimization by dividing the mesh into layers for robust least squares estimation.

Applying the method, we can highly accurately reconstruct full 3D surfaces and their texture. The method was proven to work for a wide range of scene types from a simple to complicated structures, from strong to weak texture, and from static to moving occlusions.

Our approach, however, still has several limitations which need to be improved in future work. The first limitation is that it only works with one surface. I.e., it has a problem with

non-connected or discontinuous surfaces. First of all, a way to move vertices towards edges in the image should be devised, as the latter tend to give hints on break-lines of the surface. However, this problem is usually not too pronounced, as the initial points correspond to corners by definition. Additionally, the current model treats parts of the main surface which are occluded by other parts of the surface as occlusions. It cannot reconstruct the surfaces of two or more objects. Extending our model for a full 3D reconstruction of multiple surfaces is a future research topic.

Second, the model has a limitation in reconstructing surfaces with mirror reflections, e.g., glass windows. To overcome this, one needs to find a way to detect the glass areas in the images and then apply a special algorithm which can deal with mirror reflections.

Finally, even though the approach does not require too much runtime, there is still room for improvement concerning the speed of the overall surface generation: First, at the moment the triangle meshes are manually constructed from the given 3D points. Thus, the overall time for surface generation could be substantially reduced by a technique which would automatically create the triangle meshes. Second, recently (Pons et al. 2005) have presented an approach which is similar to ours, though they link surface reconstruction with scene flow estimation. They employ graphics hardware to speed up processing. This idea could also help to speed up our algorithm, because the determination of the observations entails large numbers of projections from 3D space into the images, which could very well be solved by graphics hardware.

Bibliography

AITKEN, A. C. (1935): On Least Squares and Linear Combinations of Observations, *Proceedings of the Royal Society of Edinburgh* **55**: 42–48.

AKBARZADEH, A., FRAHM, J.-M., MORDOHAI, P., CLIPP, B., ENGELS, C., GALLUP, D., MERRELL, P., SINHA, S., TALTON, B., WANG, L., YANG, Q., STEWENIUS, H., YANG, R., WELCH, G., TOWLES, H., NISTÉR, D. and POLLEFEYS, M. (2008): Detailed Real - Time Urban 3D Reconstruction from Video, *International Journal of Computer Vision* **78**(2–3): 143–167.

BARTELSEN, J. and MAYER, H. (2010): Orientation of Image Sequences Acquired from UAVs and with GPS Cameras, *Surveying and Land Information Science* **70**: 151–159.

BOYKOV, Y. and KOLMOGOROV, V. (2003): Computing Geodesics and Minimal Surfaces via Graph Cuts, **1**: 26–33.

DEMPSTER, A. P., LAIRD, N. M. and RUBIN, D. B. (1977): Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society, Series B (Methodological)* **39 (1)**: 1–38.

EBNER, H. and HEIPKE, C. (1988): Integration of Digital Image Matching and Object Surface Reconstruction, *International Archives of Photogrammetry and Remote Sensing*, Volume (27) B11/III, 534–545.

FAUGERAS, O. (1998): From Geometry to Variational Calculus: Theory and Applications of Three-Dimensional Vision, *Workshop on Computer Vision for Virtual Reality Based Human Communications (CVVRHC)*, 52–71.

FAUGERAS, O. and KERIVEN, R. (1998): Variational Principles, Surface Evolution, PDE's, Level Set Methods, and the Stereo Problem, *IEEE Transactions On Image Processing* **7**(3): 336–344.

FISCHLER, M. and BOLLES, R. (1981): Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Communications of the ACM* **24**(6): 381–395.

FÖRSTNER, W. and GÜLCH, E. (1987): A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features, *ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, Interlaken, Switzerland, 281–305.

FOX, J. (Jan 2002): Robust Regression, *Technical report*, Thousand Oaks/London/New Delhi: Sage.

FUA, P. (1997): From Multiple Stereo Views to Multiple 3-D Surfaces, *International Journal of Computer Vision* **24**(1): 19–35.

FUA, P. and LECLERC, Y. (1994): Using 3-Dimensional Meshes to Combine Image Based and Geometry-Based Constraints, *Third European Conference on Computer Vision*, Volume II, 281–291.

FUA, P. and LECLERC, Y. (1995): Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading, *International Journal of Computer Vision* **16**(1): 35–56.

GARGALLO, P. and STURM, P. (2005): Bayesian 3D Modeling From Images Using Multiple Depth Maps, *Computer Vision Pattern Recognition*, Volume II, 885–891.

HAMPEL, R., RONCHETTI, E., ROUSSEEUW, P. and STAHEL, W. (1986): *Robust Statistics. The Approach Based on Influence Function*, Wiley.

HARTLEY, R. and ZISSERMAN, A. (2003): *Multiple View Geometry in Computer Vision – Second Edition*, Cambridge University Press, Cambridge, UK.

HEIPKE, C. (1990): *Integration von Bildzuordnung, Punktbestimmung, Oberflächenrekonstruktion und Orthoprojektion innerhalb der digitalen Photogrammetrie*, Deutsche Geodätische Kommission (C) 366, München, Germany.

HIRSCHMÜLLER, H. (2008): Stereo Processing by Semiglobal Matching and Mutual Information, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(2): 328–341.

HOUGH, P. (1962): *Method and Means for Recognizing Complex Patterns*, US Patent Number 3,069,654.

HUBER, P. (1996): *Robust Statistical Procedures*, SIAM.

ISHIKAWA, H. and GEIGER, D. (1998): Occlusions, Discontinuities, and Epipolar Lines in Stereo, *Lecture Notes in Computer Science* **1406**: 232–248.

KOLMOGOROV, V. and ZABIH, R. (2002): Multi-Camera Scene Reconstruction via Graph Cuts, *Seventh European Conference on Computer Vision*, Volume III, 82–96.

KUTULAKOS, K. and SEITZ, S. (1999): A Theory of Shape by Space Carving, *Seventh International Conference on Computer Vision*, 307–314.

LAURENTINI, A. (1994): The Visual Hull Concept for Silhouette-Based Image Understanding, *IEEE Transactions on Pattern Analysis and Machine Intelligence.* **16**(2): 150–162.

MARTIN, R., YOHAI, V. and ZAMAR, R. (1989): Min-max Bias Robust Regression, *Annals of Statistics* **17**: 1608–1630.

McGLONE, J., BETHEL, J. and MIKHAIL, E. (2004): *Manual of Photogrammetry*, American Society of Photogrammetry and Remote Sensing, Bethesda, USA.

MEER, P. (2004): Robust Techniques For Computer Vision, *Emerging Topics in Computer Vision,* Prentice Hall, Upper Saddle River, NJ, USA, 109–189.

MIKHAIL, E., BETHEL, J. and McGLONE, J. (2001): *Introduction to Modern Photogrammetry*, John Wiley & Sons, Inc, New York, USA.

NIST/SEMATECH (2010): *e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/, (February 10, 2011).

PARIS, S., SILLION, F. and QUAN, L. (2003): A Volumetric Reconstruction Method from Multiple Calibrated Views Using Global Graph Cut Optimization, *Technical Report 4843*, INRIA.

PONS, J.-P., KERIVEN, R. and FAUGERAS, O. (2005): Modeling Dynamic Scene by Registering Multi-View Image Sequences, *Computer Vision and Pattern Recognition* **II**: 822–827.

ROUSSEEUW, P. (1984): Least Median of Squares Regression, *Journal of American Statistics Association* **79**: 871–880.

ROY, S. (1999): Stereo without Epipolar Lines : A Maximum Flow Formulation, *International Journal of Computer Vision* **34**(1): 147–162.

SCHARSTEIN, D. and BLASIAK, A. (2011): Middebury Stereo Valuation - Version 2, *Technical Report*, Middlebury College, Microsoft Research, vision.middlebury.edu/stereo/.

SCHLÜTER, M. (1998): 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, *International Archives of Photogrammetry and Remote Sensing*, Volume (32) 4/1, 545–552.

SEITZ, S. and DYER, C. (1997): Photorealistic Scene Reconstruction by Voxel Coloring, *Computer Vision and Pattern Recognition*, 1067– 1073.

SEITZ, S., CURLESS, B., DIEBEL, J., SCHARSTEIN, D. and SZELISKI, R. (2006): A Comparison and Evaluation of Multi-View Reconstruction Algorithms, *Computer Vision and Pattern Recognition* 519–528.

STRECHA, C. (2007): *Multi-View Stereo as an Inverse Inference Problem*, Dissertation, Katholieke Universiteit Leuven, Belgium.

Ton, D. and Mayer, H. (2007): 3D Least-Squares-Based Surface Reconstruction, *ISPRS Journal of Photogrammetry and Remote Sensing*, Volume XXXVI, 69–74.

Wrobel, B. (1987): Digitale Bildzuordnung durch Facetten mit Hilfe von Objektraummodellen, *Bildmessung und Luftbildwesen* **3/87**: 129–140.

Yang, R., Pollefeys, M. and Welch, G. (2003): Dealing with Texture Less Regions and Specular Highlights - A Progressive Space Carving Scheme Using a Novel Consistency Measure, *International Conference on Computer Vision* 576–584.

Zamar, R. (1989): Robust Estimation in the Errors-in-variables Model, *Biometrika* **76**: 149–160.