Institute for Data Processing
Technische Universität München
Univ.-Prof. Dr.-Ing. K. Diepold

Stereo-Based 3D Scene Reconstruction for Telepresence and Teleaction

Michel Sarkis

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. J. Eberspächer
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. K. Diepold
	2. Univ.-Prof. N. Navab, Ph.D.

Die Dissertation wurde am 24.09.2008 bei der Technischen Universität München eingerichtet und durch die Fakultät für Elektrotechnik und Informationstechnik am 03.12.2008 angenommen.

Preface

This thesis was written during the time I was a research assistant at the Institute for Data Processing (LDV) at Technische Universität München (TUM). Throughout this period, I had the opportunity to meet and collaborate with several persons who have contributed to its accomplishment.

First of all, I deeply appreciate my advisor Prof. Dr.-Ing. Klaus Diepold who has provided me the opportunity to join the staff of LDV in order to conduct this research. He was always a frequent source of support and guidance in all aspects and at all times. I also want to thank Prof. Dr. Nassir Navab for the interest he showed to co-advise my work.

I would like to express my gratitude to all my colleagues at LDV and in the SFB 453 project for the enjoyable working atmosphere. I am also thankful to Prof. Dr.-Ing. Knut Hüper from the National ICT Australia for his support and insightful discussions which lead to some fruitful research cooperation without which the fulfillment of this dissertation was not possible. This work also benefited from my diploma students to whom I would also like to thank.

Since friends are the essence of life especially when living abroad and away from home, I wish to send a big thanks to all my friends who made my stay in Munich enjoyable.

Finally, I want to profoundly appreciate my parents Adib and Maguy, my brothers Mario and Charles and my little sister Dayan for their support, care and encouragement that motivated me to proceed with my work.

Munich, Germany Michel Sarkis
January 2009.

Abstract

This dissertation explores the problem of reconstructing a three dimensional scene from stereo images. It describes several propositions to enhance this process so that it can be applied in scenarios where flexibility, delay and accuracy are required as in Telepresence and Teleaction. The work mainly emphasizes on calibration of automatic zoom cameras, mesh approximation of images, stereo matching and camera motion estimation.

A method is first derived based on the theory of data fitting to calibrate automatic zoom lenses. A zoom camera is calibrated offline at a number of focus and zoom settings to obtain some measurement points of the intrinsic parameters. The unmeasured values are then interpolated by approximating each variable with local polynomial functions. A technique is then found to reduce the number of points in the image based on the concept of content adaptive mesh representation of images. There, an image is represented with a mesh where its nodes are the only pixels necessary to recover the structure of the scene. To benefit from the obtained reduced set of pixels, a sparse stereo matching strategy based on dynamic programming is developed to compute the depth only at the nodes of the mesh. This reduces the computational time to the one needed for their evaluation. At the end, a technique is established to estimate the motion of a camera by parameterizing the latter on the manifold of the rigid motion. A suitable cost function is defined. Then, a projective Newton-type minimization scheme on the manifold is derived to optimize it.

Contents

Chapter 1

Introduction

1.1 Problem Statement

Estimating the depth information of the environment from planar images is one of the major research areas in computer vision. The objective of this field is to let the computer visualize its surroundings in 3D from a pair of cameras as the human vision system can achieve with the eyes. This aim, however, faces diverse issues that have to be dealt with simultaneously and that make the goal very challenging for researchers. Recovering the geometry of a 3D scene has a variety of applications where it can be used in. One of them is telepresence where a person has to feel fully present at a location different from his own true location. This thesis explores several known issues in the 3D reconstruction process of static scenes from stereo images. The emphasis is to design and develop efficient algorithms in order to enhance the stereo-based 3D reconstruction and make it more feasible for telepresence applications. This dissertation focuses on the problems of camera calibration, stereo matching, mesh generation and camera motion estimation.

1.2 Motivation

Telepresence and teleaction (TPTA) is the experience of being present and active at a location distant from one's real physical location. In a typical scenario, a mobile robot equipped with a stereo camera is placed at a remote scene. This unit is referred to as the teleoperator. The human operator, situated at another location, can interact in the remote scene of the teleoperator by means of a generated virtual 3D environment. The environment should be transparent to the operator. It has to provide him with all the feedbacks necessary to feel "fully present" at the remote location. With the recent advances in computational resources, research in telepresence is becoming more intensive due to the wide range of its applications, e.g. teleconferencing, telemedicine, telemaintenance and collaborative virtual environments.

In order to have the virtual feeling of being present, the human operator should perceive the environment of the teleoperator using realistic 3D views at a very low latency. In a high fidelity telepresence and teleaction scenario, this setup is a part of a closed control loop which suffers from severe stability problems. One cause is the time required to construct the visual 3D environment. Another cause lies in the signal transmission round-trip delays. These delays are significant since there are too many processes that occur in TPTA and have to access the network simultaneously, see Figure 1.1. Among them are the audio, video, haptic, controls between operators and teleoperators [1].

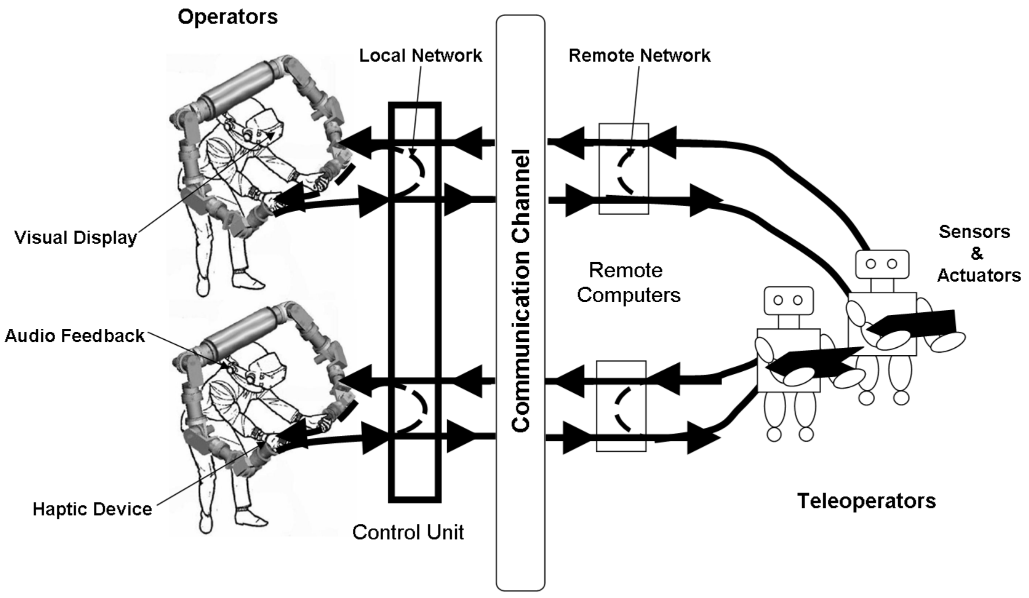


Figure 1.1: Overview of a TPTA scenario.

Research in TPTA has been pursued by several scientists and engineers due to the flexibility that can be achieved by its application. Most of the developed systems are built to support specific tasks. These tasks are mainly emphasized in the current research with 3D video teleconferencing and collaborative virtual environments. Both of these applications are somehow similar. 3D teleconferencing aims at creating long distance calls where the participants are visualized in 3D as if they were present in the same room while in collaborative visual environments, the users have to perform some tasks together, e.g. virtual gaming. In both scenarios, the environment is assumed to be known in advance and can be subtracted from the computations. The main issue is to extract the local participant from the images and display him with an immersive stereoscopic representation to the other user. The classical way of performing this task is by applying stereo reconstruction methods as was done in [2, 3, 4, 5, 6]. Another strategy is to construct the convex hull of the participant from silhouette images [7,8,9,10,11]. A third methodology is to employ image-based rendering techniques where some functions that describe the flow of light in all different directions and positions, i.e. plenoptic functions, are computed. These functions are then used to synthesize novel views of the participant [12, 13, 14, 15].

In a general TPTA scenario, however, the issue is more complex. An environment has to be first acquired by the operator in order to later perform some required tasks. One example is telemaintenance where the scene has to be scanned to locate and repair the malfunction in a remote system. Another example is telenavigation where the operator needs to scan the entourage of the teleoperator before navigating. The application of 3D reconstruction methods based on image silhouettes is not straightforward in this case. Although these methods are computationally efficient, they are focused on constructing the foreground of a segmented scene by estimating its convex hull [7,8,9,10,11]. This scheme is not straightforward in a general TPTA scenario since the structure of the scenery is not known and has to be acquired on the fly. The employment of image based rendering techniques is also not feasible for they require a large array of cameras mounted on the teleoperator to compute the flow of light in all directions [12,13,14,15]. This leaves us with the stereo-based reconstruction techniques. Recovering the 3D geometry of a scene using a stereo scheme requires several operations to be performed. The captured images of the teleoperator have to be calibrated and rectified. A stereo matching algorithm is then applied to calculate the depth map of the scene upon which the 3D points are computed. The obtained points are then registered, meshed and finally textured so that the complete 3D model can be rendered on the virtual display.

Acquiring the 3D environment has a major role in TPTA since it provides the teleoperator with the feeling of being *immersed* in the virtual world. The sense of immersion is mainly affected by the latency in the acquired geometry, the visual errors in the reconstructed scene and the restrictions imposed on the operator. Each of these problems has one or more sources that affect it. The latency of the 3D model generation is dependent on the network delays of a TPTA system as was previously mentioned, see Figure 1.1. It is also dependent on the complexity of the stereo matching algorithm used and the size of the stereo images [2,12,15,16,17]. Note that the size of the generated 3D model is also related to the size of the image. If its size is large, it will introduce further delay to the network and the rendering task since it has to be transmitted, textured and rendered on the virtual display to be visualized later by the teleoperator. The errors in the generated model are due to the inaccuracy induced from the camera calibration algorithm used, the type of the stereo matching algorithm employed and the technique applied to register the consecutive 3D meshes [2,4,17]. In addition, restrictions are imposed on the movements of the operator if variable zoom lenses are installed on the teleoperator since they require continuous calibration to determine the camera parameters as the zoom changes. Using auto or self-calibration algorithms to compute these parameters make the movement of the teleoperator subject to some critical motion where these methods might fail, see [18,19,20] for more information on camera auto-calibration algorithms.

Beside that, the work of this dissertation is related to the subproject M3 of the SFB 453 telepresence and teleaction project. Its goal is to construct a photo-realistic predictive display for a TPTA scenario. The display should generate for the operator 3D model of the scene viewed by the teleoperator in the local network, see Figure 1.1. The model is used to compensate for the delay caused by the signals being transmitted over the communication channel. It should let the operator depict the scene from the positions viewed by the teleoperator and predict the views of the positions that were not covered. In order to accomplish this objective, the local model should be frequently updated at a very low latency. In addition, the reconstruction errors should be as minimum as possible to lower the artifacts so that the model remains visually acceptable. Both of these criteria are important to make the operator have the virtual feeling of being present at the remote scene. The part of the subproject M3 that concerns this thesis is to construct the 3D model. This will be fulfilled using a stereo-based reconstruction scheme due to the reasons that were mentioned earlier. Before we get into details of the contributions made in this work, we will first show how to integrate such a reconstruction scheme in a general TPTA system. A schematic of the scenario is illustrated in Figure 1.2. At the teleoperator site, the stereo images are captured with firewire cameras, passed to a MPEG4 video encoder and then transmitted over a channel. At the other side of the channel where more computational power is available, the video stream is received, decoded and processed to reconstruct the 3D scene. The generated model is placed in a server and can be accessed by the operator via the MPEG4 BIFS standard [21]. This standard possesses a streaming architecture which provides the operator with the ability to connect from anywhere without any restrictions on his location. At the operator site, the 3D scene is received and rendered. The control line feeds the teleoperator with the controls of the stereo cameras, i.e. head tracker of the operator and camera controls. In addition, it provides the acquisition algorithm with the controls of the zoom lens needed for calibration.

The stereo-based 3D reconstruction scheme lies in the middle of the depicted system. It is composed of several independent blocks which have to be all accomplished in order to obtain a complete 3D model of a scene. The images have to be first calibrated, especially if variable zoom lenses are used, then rectified. A stereo matching algorithm should run to estimate the depth of the scene. The mesh approximation of the generated depth map should be constructed. The motion of the camera should be computed to register the generated mesh with the previous ones and update the overall model accordingly. In order for the latter to occur, feature matches have to be tracked all over the image sequence. Moving objects should be detected and segmented from the scene so that they can be tracked and updated separately . . .

This dissertation will consider four challenging problems arising in this process. It will deal with the calibration of an automatic zoom lens camera. It will also discuss the topic of constructing the mesh representation of stereo images. Another important topic that will be covered is stereo matching since it is required to recover the 3D geometry of the scene. Finally, the thesis will address the problem of estimating the global motion of a stereo camera which is necessary to track the movement of the teleoperator.

1.3 Original Contributions

There are several points in which the author believes that the thesis presents an original contribution and will be explained in the rest of this manuscript:

- An algorithm that allows an automatic zoom camera to be calibrated and integrated in a TPTA scenario without incurring any constraints or restrictions on the teleoperator.

- A method that reduces the number of pixels to be processed in an image based on non-uniform sampling. The approach is also able to represent the image with a content adaptive mesh which can be used to render the scene on a virtual display.

- A stereo matching technique based on dynamic programming that handles sparse images.

- The idea of generating the mesh and reducing the number of pixels to be processed in stereo reconstruction. This accelerates the overall acquisition process along with the rendering of the 3D model on the virtual display due to the reduced number of points. This methodology becomes more benefitable when the size of the used images gets larger.

- A framework to optimize the motion of a stereo camera (teleoperator) by parameterizing the motion variables using the manifold of the rigid motion, defining a cost function to be minimized and deriving a projective Newton-type scheme to optimize it accordingly.

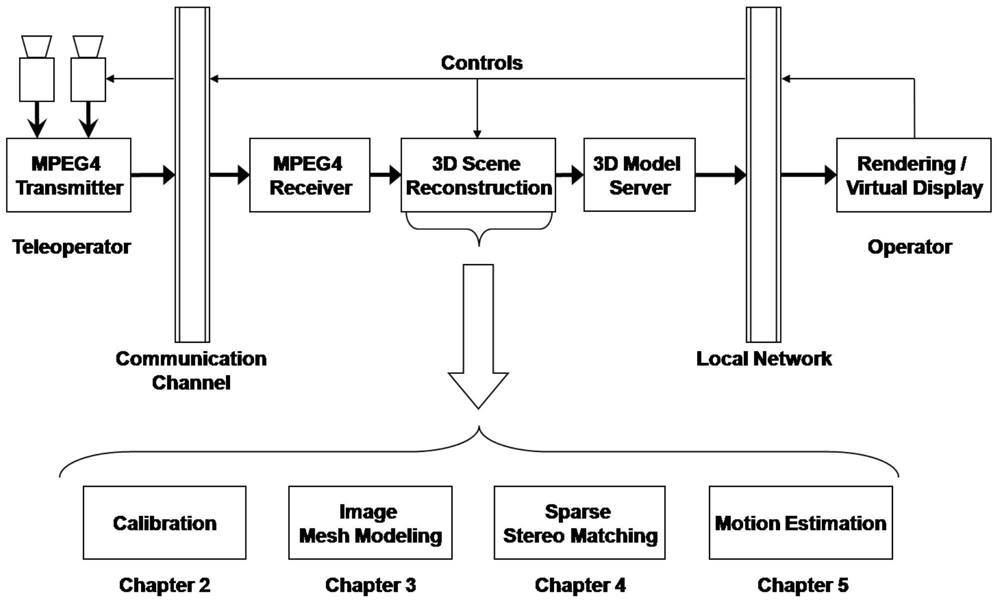


Figure 1.2: A stereo-based 3D scene reconstruction scheme integrated in a TPTA system.

1.4 Organization of the Dissertation

This thesis addresses some of the problems that present a challenge in stereo reconstruction in telepresence scenarios like camera calibration, mesh generation, stereo matching and camera motion estimation. A review on existing telepresence systems was conducted in this chapter. Each specific part, however, has its own research field where intensive studies have been conducted. We will therefore present the literature review of each topic that will be dealt with in this work in the corresponding chapter where it is discussed. The work is structured as follows:

Chapter 2 proposes an approach to determine the intrinsic parameters of an automatic zoom camera by estimating continuous functions of their variations as the focus and the zoom change. The method is based on the moving least-squares (MLS) multiple regression scheme which determines from a predefined number of samples, the complete function of the intrinsic parameters. MLS fits locally a polynomial function at each focus and zoom setting by using the measured neighboring points. In order to reduce the computational complexity of MLS, we propose another algorithm in which the MLS generated curves are clustered. Then, each cluster is approximated with a single polynomial function.

Chapter 3 suggests a method based on binary space partitions to approximate an image with a content adaptive mesh. The algorithm is able to simultaneously reduce the number of pixels and generate the mesh approximation of the scene represented by an image. The idea is to assume that each triangle of the mesh is a plane formed by its three vertices. Thus, it will be possible to reconstruct the inlying pixels inside each triangle with its parametric equation. If a triangle does not have the ability to reconstruct the pixels lying within up to a predefined error, it will be recursively subdivided into smaller triangles until the error is satisfied across the image. The resulting mesh can then be used to construct the 3D mesh of the scene using the depth information estimated from stereo matching.

Chapter 4 derives a framework to generate a sparse depth map in order to benefit from the reduced set of pixels obtained in the previous chapter. A dynamic programming based stereo matching algorithm is formulated which computes the depth only at the sparse samples and hence accelerating the stereo reconstruction process. The smoothness constraint is modified to take into account the distance between the sparse samples. It is then shown by setting up some tests that the non-uniform samples are sufficient to recover the dense depth map of the scene by interpolating the samples using the mesh with an acceptable error.

Chapter 5 presents a projective Newton-type approach on the manifold to enhance the accuracy of the estimated motion parameters while not significantly increasing the accompanying costs in the computations. The key issue is to parameterize the motion variables of a camera on the manifold of the Euclidean motion by using the underlying Lie group structure of the motion representation. A cost function is formulated. The gradient and the Hessian formulation on the manifold are derived to minimize the cost function such that the minimum is the sought estimate of the rigid camera motion.

Chapter 6 applies the developed schemes in this dissertation in a real telepresence and teleaction scenario and assesses their performance.

Chapter 7 summarizes the thesis and reviews the main contributions. Possible directions of research are also highlighted.

Appendix A sums up all the abbreviations, mathematical symbols and notations that are used throughout this work.

Parts of this thesis have been published in [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32].

Chapter 2

Calibration of Automatic Zoom Cameras

The application of automatic zoom lenses in camera systems has become significantly important in the field of machine vision. By varying the focal length, the focus and the aperture values, a zoom camera system can be adjusted to different fields of view, depth of fields and lighting conditions. These advantages present some of the main reasons why zoom lenses are increasingly being adopted in applications like 3D scene reconstruction, visual tracking, robot navigation [33, 34, 35]. Such benefits are also important in a TPTA scenario since they provide the operator the ability to interact with the virtual environment by zooming on some specific regions of interest.

2.1 Overview of Camera Calibration

The challenge of applying automatic zoom lenses in the vision systems lies in modeling the image formation process as the lens parameters, i.e. focus, zoom and aperture, are varied continuously. The image formation process describes the relationship between $\boldsymbol{P} = (X, Y, Z)^T$, an existing point in the real world coordinate system $\mathbb{R}^3$, and its corresponding point $\boldsymbol{p} = (x, y)^T$ in the image coordinate system $\mathbb{R}^2$. This relationship is expressed as

$$l \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.1}$$

using the perspective projection camera model [20], see Figure 2.1 for an illustration. The matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix representing the

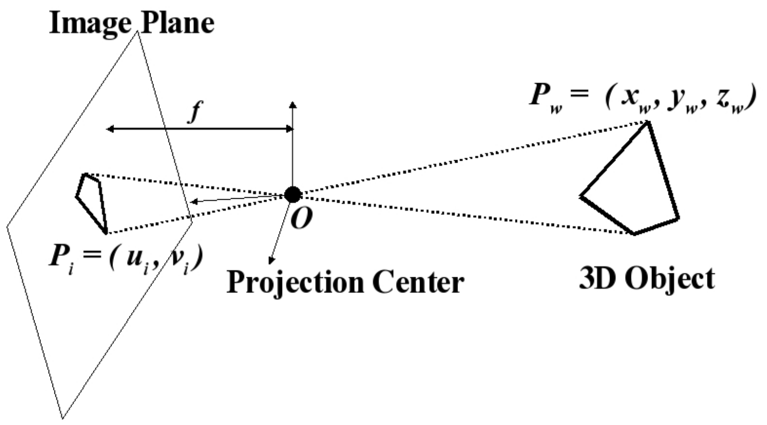


Figure 2.1: The camera perspective projection model. An object in 3D is projected on the image plane through the camera projection center. The distance of the projection center to the image plane along the optical axis is the focal length f. The projection of the center along the optical axis on the image plane is the principal point (x_0, y_0).

camera's orientation in the world coordinate system, $\mathbf{t} \in \mathbb{R}^3$ is the translation vector between the origin of the world and the origin of the camera coordinate system, l is a scalar and $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the matrix of the intrinsic parameters defined by

$$\mathbf{K} = \begin{pmatrix} f_x & \sigma & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.2}$$

where (x_0, y_0) are the coordinates of the principal point, (f_x, f_y) are the components of the focal length and σ is the skewness of the two image axes. Due to manufacturing imperfections in a zoom lens, errors are introduced and have to be taken into account when calibrating a camera [36, 37, 38]. These faults are mainly known as the radial and tangential distortions. They tend to add some non-linear effects to the perspective projection camera model and consequently to the image formed on the sensor [39, 40]. An example of such erroneous effects is illustrated in Figure 2.2.

In order to take these variables into account, the simplest approach is to calibrate the zoom camera at a number of lens settings, i.e. determine the intrinsic parameters and the distortions at a number of zoom, focus and aperture settings and then save the obtained values in a look-up table [41]. This setup, however, requires numerous measurements and the resulting data requires large memory resources. A different and well known strategy is to apply auto-calibration algorithms. These methods are either based on finding the absolute conic, which will be used to compute a transformation from the

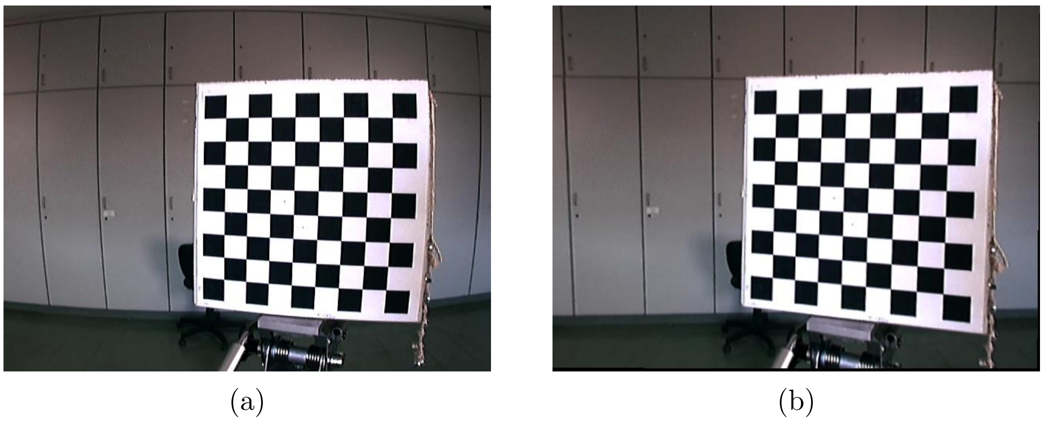


Figure 2.2: Example of a distorted image in (a) and its undistorted version in (b) after undistorting the image.

projective frame to the (calibrated) metric frame, or on solving the Kruppa equations which represent an algebraic representation of the correspondence of the epipolar lines tangent to the absolute conic [42, 43, 20]. These techniques estimate the parameters directly from a given sequence of uncalibrated images, e.g. by means of rotating the visual system [44, 45], by zooming [46] or by pan-tilt movements [47, 48]. The importance of finding the absolute conic in both groups resides in the fact that it is invariant under Euclidean transformations, i.e. it's relative position to a moving camera is constant, see Figure 2.3. Once the absolute conic is found in the projective form, a calibration transformation can then be uniquely determined. However, the usage of these methods is limited since they are highly susceptible to errors mainly due to the inaccuracy in finding the image of the conic [49, 50, 20]. Another limitation resides in the constraints imposed on the movement of the camera system. These constraints are due to the degenerate configurations where the auto-calibration algorithms might fail [18, 19, 20]. This makes the application of such methods in TPTA limited.

Another approach is to treat the zoom camera system as an input/output function as shown in Figure 2.4. The inputs in this case are the focus, the zoom and the aperture, while the outputs are the intrinsic parameters of the camera. With a predictive model of the zoom camera system, it is possible to derive a function that is able to return the intrinsic parameters of the camera at some measured points of focus, zoom and aperture. In addition, this function will be able to interpolate the intrinsic parameters at the unmeasured points. The drawback of such techniques, as compared to the auto-calibration methods, is the need to perform offline calibration at some predefined values of focus, zoom and aperture settings. Once the model of the intrinsic parameters is computed, i.e. the vision system is fully

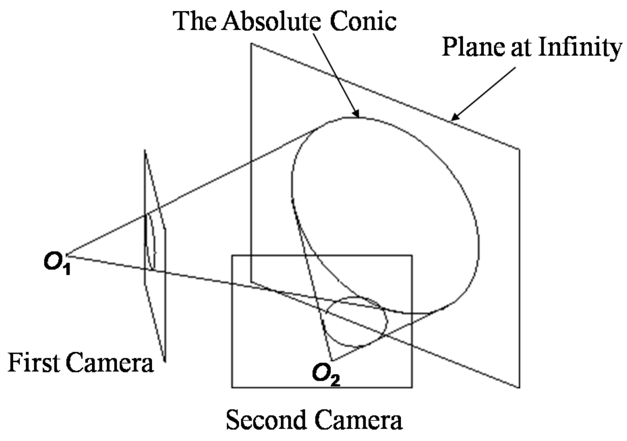


Figure 2.3: The absolute conic which lies on the plane at infinity and it's projections on the images, i.e. the images of the absolute conic. Most of the auto-calibration algorithms are based on finding the absolute conic since it's relative position to a camera is constant.

calibrated, the metric reconstruction of the scene is possible with the first pair of images. No sequence of images is needed to calibrate the system at each zoom setting. In addition, no "critical motion" can appear as with the classical auto-calibration techniques. Hence, there will be no constraints imposed on the movement of the teleoperator in TPTA. For more information on camera calibration, the interested reader may refer to [20, 51, 52]. Due to these reasons, the developed method that will be described in the rest of this chapter will be adopting the last strategy.

2.2 General Assumptions

The intrinsic parameters of a camera are expressed by the focal length, the coordinates of the principal point and the skew parameter. The image distortion terms which are incurred from the imperfectness of the optical lenses are described by the coefficients of their Taylor series expansion [41, 53, 20]. To facilitate the estimation of the underlying function, some assumptions have to be made regarding some of these variables.

Skew Parameter: The skew parameter reflects the angle between the axes of a pixel. In modern camera systems, the angle can be safely assumed to be $90°$ due to the improvement of the manufacturing process [20]. Thus, the skew parameter σ will be neglected in our analysis and will be set to 0.

Aperture Setting: The aperture of a camera controls the amount of light that reaches the sensor. By varying the width of the aperture, a camera system can be brought into focus by extending its depth of field. In this work,

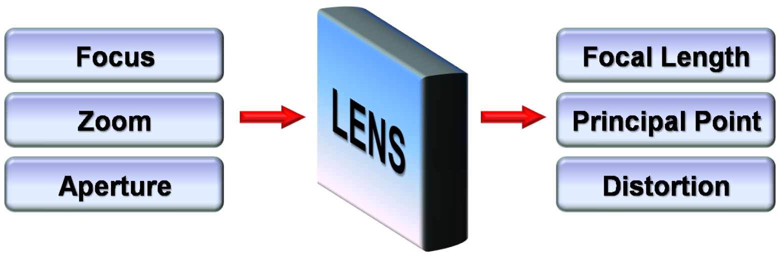


Figure 2.4: Schematic of the zoom camera system. The focus, zoom and aperture are the inputs to the function. The camera intrinsic parameters, i.e. focal length and principal point, and the radial distortion represent the output.

the aperture will be set to a constant value suitable for all lighting conditions since it has been shown in [41, 54] that it has a very negligible influence on the intrinsic parameters. Another reason is to limit the input because the combination of focus, zoom and aperture has a huge number of possibilities. Consequently, it will require a large amount of measurements to be done in order to estimate the underlying model of the camera parameters.

Lens Distortion Coefficients: The usage of lenses leads to non-linear distortions in the image. These distortions have to be taken into account in the perspective projections camera model in order to rectify them. In [39], an expression that models the distortion of a mono-focal lens was proposed

$$\begin{aligned} x_d &= x_n\left(1+\kappa_1 r^2+\kappa_2 r^4+\kappa_3 r^6+\cdots\right) \\ &\quad +\left[2\tau_1 x_n y_n+\tau_2\left(r^2+2x_n^2\right)\right]\left(1+\tau_3 r^2+\cdots\right), \\ y_d &= y_n\left(1+\kappa_1 r^2+\kappa_2 r^4+\kappa_3 r^6+\cdots\right) \\ &\quad +\left[\tau_1\left(r^2+2y_n^2\right)+2\tau_2 x_n y_n\right]\left(1+\tau_3 r^2+\cdots\right), \end{aligned} \tag{2.3}$$

where

$$x_n = X/Z, y_n = Y/Z, r = \sqrt{{x_n}^2 + {y_n}^2}. \tag{2.4}$$

x_d and y_d represent the distorted pixel coordinates of the object point $\boldsymbol{P}$ in the image; x_n and y_n are the normalized coordinates of $\boldsymbol{P}$ in the image, $\kappa_1, \kappa_2, \ldots$ are the coefficients of the radial distortion and $\tau_1, \tau_2, \ldots$ are the coefficients of the tangential distortion.

The effect of the tangential distortion is insignificant in today's lens systems. The radial distortion is mostly dominated by the leading term of the power series expansion $\kappa_1 r^2$; whereas, higher order terms $\kappa_2 r^4, \kappa_3 r^6, \ldots$ are barely significant. It can be also shown, that the higher order terms would result in a numerical instability in Equation (2.3) [38, 39, 41]. Hence, only the first radial distortion coefficient κ_1 which will be referred to as κ in the sequel is considered.

The matrix **K** of the intrinsic parameters can be thought of as a transformation that maps a pixel from the camera coordinates to its true position on the sensor plane. Taking into account the previous assumptions, the intrinsic parameters and the radial distortion form three consecutive transformations which can be written in the form

$$\begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1+\kappa r^2 & 0 & 0 \\ 0 & 1+\kappa r^2 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \tag{2.5}$$

The matrix to the right transforms the undistorted camera pixels $\left(x_n, y_n\right)$ to the distorted form $\left(x_d, y_d\right)$. The middle matrix scales the pixels with the focal length and the left matrix shifts the pixels to their true positions with the coordinates of the principal point.

2.3 Related Work

The objective of this work is to find the underlying model of the intrinsic parameters of a zoom lens camera. This objective intersects with the goal of the manifold learning methods. There, some multidimensional measured data assumed to be uniformly sampled from a smooth manifold is given. The main issue is to obtain a meaningful representation of the data in order to perform subsequent operations such as dimension reduction and clustering.

Principal component analysis (PCA) and independent component analysis (ICA) present some state of the art methods. They are able to find such representations by estimating a linear combination of some basis functions [55]. When the parameters vary in a non-linear manner as in the case of the intrinsic parameters of a camera [53], however, they cannot be well represented using such a linear basis. The limitation induced by the usage of PCA and ICA motivated the development of other methods like local linear embedding (LLE) [56,57,58], diffusion maps [59,60] and isomap [61]. These algorithms parameterize the data using lower dimensional manifolds. Nevertheless, the methods do not have the capability to back project the computed model into the original data set as can be done using PCA and ICA. This is of major importance in our case since what is sought are the real values of the intrinsic parameters in order to calibrate a camera. Despite the fact that some limited techniques have been recently developed to overcome this issue [62], this topic remains an active area of research.

To find the underlying model of camera parameters, the simplest developed method is to measure each intrinsic parameter using a standard offline calibration technique at several focus and zoom settings. Then, the data is stored in a look-up table [63,41,54]. This leads unfortunately to a large memory requirement especially when a precise model is needed. This is justified since for a single focus and zoom setting, it is necessary to store the values of the focus, zoom, focal length in x- and y-direction of the pixel axes, the two coordinates of the principal point and the coefficient of the radial distortion.

Another methodology is to consider a zoom camera as an input/output function. The focus and zoom settings of the camera are considered to be the inputs while the intrinsic parameters are considered to be the outputs, see Figure 2.4. Consequently, by taking some samples of the input and the output of the system, it is possible to find the functions that relate these two entities using curve fitting techniques. The goal of curve fitting is to find a curve that approximates the variation of the scattered data points and which will also allow the interpolation of the unmeasured points. Many curve fitting techniques can be found in the literature such as regression analysis, polynomial fitting and radial basis functions. Good reviews on such methods can be found in [64,65,66].

In [53], a camera parameter is fitted with a bivariate polynomial. The polynomial is empirically estimated by minimizing an error function over the measured data points. This method has a very efficient memory consumption since only the coefficients of the polynomial functions need to be stored. The general equation that defines a bivariate polynomial ϖ is

$$\varpi\left(s,o\right)=\sum_{i=0}^{m}\sum_{j=0}^{m-i}\gamma_{ij}s^{i}o^{j}, \tag{2.6}$$

where s and o represent a single focus and zoom setting, m is the degree of the polynomial and γ_{ij} are the polynomial coefficients. The number of coefficients ξ of the polynomial function is given by

$$\xi=\frac{\left(m+1\right)\left(m+2\right)}{2}. \tag{2.7}$$

A polynomial of order m can be generally considered as a Taylor's series expansion of the underlying function truncated after m coefficients. The higher the order m of the polynomial, the more close can the Taylor series approximate the underlying behavior of the measured points.

However, a high order m leads to a large number ξ of coefficients, as shown in Equation (2.7). The increase in ξ leads in its turn to an increase in both the computational costs and the number of points needed to estimate the polynomial. In addition, a high order polynomial can also overfit the data. To overcome these issues, the term m_{max} was derived. It reflects the maximum order of the polynomial that can fit a set of N data points

$$m_{max} = \left\lfloor \frac{\sqrt{8N+1}-3}{2} \right\rfloor , \tag{2.8}$$

with $\lfloor \cdot \rfloor$ representing the floor operator [53]. To approximate the intrinsic parameters, a polynomial of a specific degree is chosen to represent each of the variables, i.e. polynomials of degree 5 for each of the focal length and principal point components and one of degree 2 for the radial distortion coefficient. Then, the solution polynomial for each variable is computed globally using the whole measured points in a least-squares (LS) sense.

2.4 Calibration with Moving Least-Squares

The LS fitting technique of [53] fits the camera parameters using a global scheme. This will cause the approximation not to be accurate especially for highly scattered variables, e.g. see Figure 2.10. If the global LS solution can be combined from several local solutions instead, the accuracy of the approximation will be enhanced. This is the basic idea behind deriving moving least-squares (MLS) in this section to approximate the intrinsic parameters of a zoom lens camera. This idea is similar in spirit to that of the LLE method of [56] used to embed the data to a lower dimensional manifold. The objective here is to approximate the 3D surface of an intrinsic parameter in order to interpolate its values at the unmeasured focus/zoom settings. In addition, an extension is proposed to reduce the computational complexity of MLS and make it more suitable to be utilized in TPTA applications.

2.4.1 Problem Formulation

The basic idea of the MLS approximation is to start with an arbitrary fixed point and then move it over the entire domain where the variable is defined. At each point, an approximation function is locally minimized and evaluated using weighted least-squares by fitting the neighboring data points. The approximation function must be chosen while taking into account that it is continuously differentiable. This guarantees that the computed MLS surface is smooth [67,68]. A comparison between the LS and MLS approximation in the two dimensional case is shown in Figure 2.5.

Given is a set of measured focus/zoom inputs and the corresponding intrinsic parameter, e.g. focal length, at these points. Let $\chi_i = \{s_i, o_i\}_{i \in S}$ be the set of these distinct data points in $\mathbb{R}^2$, and let $\{\rho(s_i, o_i)\}_{i \in S}$ be the intrinsic parameter values at the focus and the zoom settings, i.e. $\rho(s_i, o_i) = \theta_i$. The objective of MLS is to find the coefficients vector γ of the polynomial that minimizes a local error function of the intrinsic parameter at an unmeasured point χ_j using its (measured) neighboring points from χ_i. Note that the vector γ is formed by the coefficients γ_{ij} of Equation (2.6). The length of γ is given by Equation (2.7) which depends on the order of the chosen polynomial. With a bivariate polynomial of degree m, the basis vector $\psi(s_i, o_i)$ of the polynomial at each of its neighboring points is

$$\psi(s_i, o_i) = (1, s_i, o_i, s_i \cdot o_i, \ldots, s_i^m, o_i^m)^{\mathrm{T}} \in \mathbf{\Pi}_m^2, \tag{2.9}$$

where $\mathbf{\Pi}_m^2$ presents the space of the polynomial vectors of degree m in $\mathbb{R}^2$ of the intrinsic parameter. The error function of MLS should be able to estimate the value of the intrinsic parameter at χ_j from its neighbors in a least-squares sense. The suggested error function that performs this task is given in the sequel.

Definition 2.1 *The moving least-squares approximation of degree m at a point $\chi_j \in \mathbb{R}^2$ of the focus/zoom settings is the vector γ that minimizes the weighted least-squares error function E*

$$E = \sum_{i \in I} \vartheta\left(\mathrm{d}\left(\chi_i, \chi_j\right)\right) \cdot \left[\psi^T\left(s_i, o_i\right)\gamma - \rho\left(s_i, o_i\right)\right]^2, \tag{2.10}$$

where $\vartheta(\cdot)$ is a non-negative weighting function and $\mathrm{d}(\chi_i, \chi_j)$ is a distance in $\mathbb{R}^2$.

As can be noticed, the MLS approximation leads to a weighted LS solution. It decides using the weighting function $\vartheta(\cdot)$ how each of the neighbors of χ_j can contribute to the result depending on its distance from χ_j. In the selection of a suitable weighting function for our problem, it is important to keep the following definition into consideration.

Definition 2.2 *The MLS weighting function must satisfy two conditions. First, it should allow the approximation of the intrinsic parameter to be local. Second, it should ensure that the approximation allows for interpolation.*

An approximation is local if it is rapidly decreasing or of finite support. The first condition suggests that the weighting function $\vartheta(\cdot)$ of MLS must force the weights of the points that are distant from χ_j to vanish. Consequently, the solution will be local in the sense that only the neighboring points to χ_j will be affecting the solution. An approximation allows for interpolation if $\lim\limits_{\mathrm{d}(\chi_i,\chi_j)\to 0} \vartheta\left(\mathrm{d}\left(\chi_i, \chi_j\right)\right) = \infty$. This means that the closer is χ_j to a measurement point, the more probable that it has a similar value of the

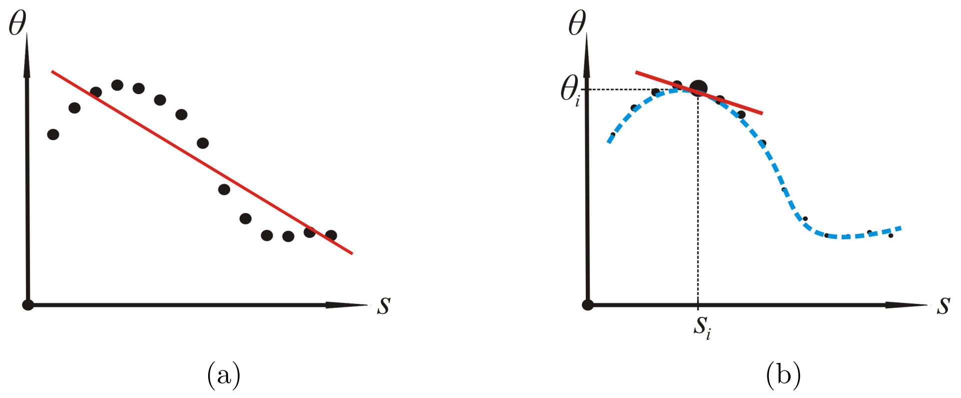


Figure 2.5: Comparison between LS in (a) and MLS in (b) approximations using a polynomial of degree 1. The global nature of LS treats all data points equally, i.e. equal weights. Using MLS, local weights are computed at the neighboring data points of a certain point (s_i, θ_i) (solid line). By moving this particular point over the whole domain, the resulting approximation surface is closer to the data points (dashed line) than that of the LS.

intrinsic parameter. The importance of this condition is that it obliges the close points to have similar values and, hence, enforcing the MLS approximation to be smooth.

Proposition 2.1 *A weighting function that satisfies the conditions of Definition 2.2 is*

$$\vartheta\left(\mathrm{d}\left(\chi_i, \chi_j\right)\right) = \frac{1}{\mathrm{d}^2\left(\chi_i, \chi_j\right)}. \tag{2.11}$$

Proof To prove that Equation (2.11) satisfies the first condition of Definition 2.2, it is necessary to show that the $\vartheta(\mathrm{d}\left(\chi_i, \chi_j\right))$ is rapidly decreasing as $\mathrm{d}\left(\chi_i, \chi_j\right) \rightarrow \infty$. The proof of this condition is quiet straightforward since the first derivative of Equation (2.11) is always negative. To prove that the proposed function satisfies the second condition, it is sufficient to show that $\lim\limits_{\mathrm{d}(\chi_i,\chi_j)\rightarrow 0} \vartheta\left(\mathrm{d}\left(\chi_i, \chi_j\right)\right) = \infty$ which is true in our case since $\lim\limits_{\mathrm{d}(\chi_i,\chi_j)\rightarrow 0} \frac{1}{\mathrm{d}^2(\chi_i,\chi_j)} = \frac{1}{0} = \infty$. ■

2.4.2 Minimization of the Error Function

Before deriving the solution, let us rewrite Equation (2.10) as

$$\left\|\mathbf{W}\mathbf{\Psi}\gamma - \mathbf{W}\theta\right\|^2, \tag{2.12}$$

where each row of $\mathbf{\Psi}$ represents the polynomial basis vectors $\psi(s_i, o_i)$ at every point χ_i, $\mathbf{W}$ is the diagonal weight matrix, γ reflects the vector coefficients of the MLS approximation at χ_j and θ is the vector of the measured values of the intrinsic parameters at each χ_i, i.e. θ_i, and $\left\|\cdot\right\|$ is the 2 norm.

To solve for γ, it is necessary to calculate the Euclidean distances d_{ij} of each point χ_j with its neighbors. The weights are then computed using Equation (2.11) and rearranged into the diagonal matrix $\mathbf{W}$. The estimated vector of MLS coefficients γ, the solution of Equation (2.12), is computed as

$$\gamma = \left(\mathbf{\Psi}^T \mathbf{W} \mathbf{\Psi}\right)^{-1} \mathbf{\Psi}^T \mathbf{W} \theta. \tag{2.13}$$

Once γ is estimated, the MLS approximated value of the intrinsic parameter $\hat{\theta}_j$ at the focus/zoom setting χ_j is then obtained as

$$\hat{\theta}_j = \psi\left(s_i, o_i\right) \cdot \gamma. \tag{2.14}$$

The MLS computations have to be carried out at every camera setting χ_j that needs to be interpolated and for every intrinsic parameter. Fortunately, Equation (2.12) can be formulated to estimate simultaneously all the intrinsic parameters at χ_j by rewriting it in the form

$$\left\|\mathbf{W}\mathbf{\Psi}\mathbf{\Gamma} - \mathbf{W}\mathbf{\Theta}\right\|^2, \tag{2.15}$$

where every column of $\mathbf{\Gamma}$ represents the vector of coefficients γ corresponding to each intrinsic parameter. Similarly, each column of $\mathbf{\Theta}$ holds the measured values of an intrinsic parameter at each χ_i.

To ensure that only the points in a certain distance around χ_j contribute to the MLS approximation, the computation of the distances can be extended in a way that only the points within a certain predefined radius are considered in the calculations. This leads, however, to the problem of the proper choice of the radius since if the distance between the data points is too large, it is possible that no neighboring values will be chosen.

To overcome this limitation, the k nearest neighbors with $k \geqslant \xi$ are chosen instead in this work. For a bivariate polynomial function of degree m, the minimum number of neighbors ξ is computed using Equation (2.7). In order to retain the stability of the computations, it is preferable to always choose more than ξ points so that Equation (2.13) remains well conditioned.

Searching for the k nearest neighbors is a computationally expensive operation which has to be performed for every focus/zoom setting. The complexity of the search is in the order of $O\left(ND + k\right)$ where N is total number of measured focus and zoom settings and D in our case is 2 (focus and zoom).

In order to accelerate this process, it is preferable to build a kD-tree of the measured points. Then, the search for the nearest neighbors is performed along this tree. This will reduce the complexity of the computations to $O\left(N^{1-\frac{1}{D}}+k\right)$ which is in the case of this work $O\left(N^{\frac{1}{2}}+k\right)$ [69]. The MLS based modeling algorithm is illustrated in Table 2.1.

MLS is considered to deliver the best approximation of scattered data points since it bounds the local error with the error of the best local approximation [67,68]. This explains the wide employment of MLS to approximate the surfaces of 3D scattered data points from range images [70,71,72,73].

The MLS algorithm uses the data values within an open ball around χ_j since the weighting function suggested in Equation (2.11) forces the weights of the points that are outside of the ball to vanish even if the k nearest neighbors are used in the computations. In other words, only the close neighbors contribute to the reconstruction of the intrinsic parameter at χ_j by minimizing the linear error function in Equation (2.10). However, the nearest neighbor search introduces a non-linearity in the MLS solution since it can be considered as a non-linear thresholding function. These points are similar to what is performed in the LLE method, see [56]. Both techniques use the nearest neighbor search to construct a group of local least-squares solutions which when concatenated forms a global solution. The main difference resides in the fact that the MLS technique approximates the data with polynomials and thus allowing us to interpolate the unmeasured values of the intrinsic parameters while the LLE method embeds the data to a lower dimensional manifold. If the embedded data is interpolated, it cannot be back projected to recover the sought values of the intrinsic parameters.

The MLS technique approximates each intrinsic parameter independently. It interpolates each variable without taking into account its interdependency with the other ones. Despite the fact that the focus/zoom measurements can be obtained using a global optimization method like [74] which considers this issue, it might be more convenient to model the intrinsic parameters using a scheme that counts this dependency. This motivates the investigation of a suitable parametric representation of the manifold structure of the intrinsic parameters. The representation must allow us to have a two way mapping between the parameters and the manifold structure and not just a one way mapping as in LLE or any other manifold learning algorithm, see [56,57,58,59,60,61]. Such a feature will provide us with the possibility to embed the seven dimensional space, formed by the focus and zoom inputs along with the intrinsic parameters, into a lower dimensional space where the data can be fitted easier than the original space. Then, by back projecting the computed model into the higher space, it will be possible to recover the real values of the intrinsic parameters required for camera calibration. Whether it can be achieved or not, this point presents an interesting topic for future research.

Table 2.1: The MLS based zoom lens modeling algorithm.

```
Construct a $kD$-tree using the measured focus and zoom settings

FOR every focus and zoom setting $(s_j, o_j)$, do the following:

1   Determine at least $k$ nearest neighbors of $(s_j, o_j)$ using
    the tree where $k$ is computed with Equation (2.7).
2   Calculate the Euclidean distances between the chosen
    setting $(s_j, o_j)$ and all of its neighbors $(s_i, o_i)$.
3   Weight the distances with Equation (2.11) and solve
    Equation (2.12) using a bivariate polynomial of order $m$.
4   Calculate the local approximation $\hat{\theta}_j$ with Equation (2.14).
```

2.4.3 Clustered Moving Least-Squares

The MLS based approach requires a lot of computations to determine the intrinsic parameters of a zoom lens camera. If it has to be applied to an online system like telepresence, numerous computation steps has to be made; while in [53], it is only necessary to evaluate a simple polynomial function. To reduce the computational power, a clustering technique will be employed to subdivide the points of each intrinsic parameter into several clusters. A MLS surface is then generated for each cluster as in Section 2.4.2; hence, the naming clustered MLS (CMLS). At the end, each cluster will be modeled with a bivariate polynomial function. Consequently, each intrinsic parameter will be approximated by several bivariate polynomial functions. This will reduce the required online computations by CMLS to the evaluation of these functions as in [53]. The method is summarized in Table 2.2.

Clustering

In the current problem, some of the lens parameters suffer from irregularities in their densities, i.e. they are very scattered, as in the case of the coordinates of the principal point and the radial distortion coefficient. The main idea in CMLS is to cluster the highly scattered data into similar regions. Then, each one will be approximated in a similar manner to [53]. So instead of obtaining a single representative polynomial for an intrinsic parameter, it will be described by several polynomials. In general, two groups of methods can be applied to cluster a data set [70]. The first, known as clustering by region growing, is based on choosing a random point and then building a cluster by successively adding neighboring points. An example from this group is the kmeans clustering method [75]. The second, hierarchical clustering or binary space partitions [76], splits the data set into different clusters recursively using some predefined criteria, see Figure 2.6.

A hierarchical clustering scheme is used in this work since it has the advantage of organizing a data set. In order to split the data, two criteria are used: The *scatterness* of the points and the minimum number of points within a cluster. The interpretation of the second criterion is simple, a cluster is formed if the number of its points is more than a predefined number. To fulfill the first criterion, the coefficient of determination is applied [77].

Definition 2.3 *The coefficient of determination R^2 reflects the goodness of the fit, i.e. it describes how precise the cluster is approximated with a plane. A value $R^2 = 1$ denotes that all the points of the cluster are located on the regression plane while a value of 0 means that the data is highly scattered.*

Given a cluster of n zoom and focus setting points, a multiple linear regression is carried out to fit a plane. Suppose that (s_j, o_j) is a point of this cluster where θ_j is the measured value of the intrinsic parameter. The point can be approximated using a planar equation as

$$\hat{\theta}_j = \gamma_{00} + \gamma_{01} \cdot o_j + \gamma_{10} \cdot s_j, \tag{2.16}$$

where γ_{00}, γ_{01} and γ_{10} are the coefficients of the plane using Equation (2.6) with $m = 1$ and $\hat{\theta}_j$ is the approximated (fitted) value of θ_j. The coefficient of determination R^2 is defined as [77]

$$R^2 = \frac{\text{SSR}}{\text{SST}}. \tag{2.17}$$

SSR describes the sum of the squared residuals given by

$$\text{SSR} = \sum_{j=1}^{n} \left(\hat{\theta}_j - \bar{\theta}_j\right)^2, \tag{2.18}$$

where $\bar{\theta}_j$ is the mean of the samples in the cluster, i.e. $\bar{\theta} = \frac{1}{n}\sum_{j=1}^{n}\theta_j$ and SST reflects the total sum of squares errors

$$\text{SST} = \sum_{j=1}^{n} \left(\theta_j - \bar{\theta}_j\right)^2. \tag{2.19}$$

As a result, it is possible to decide whether or not to subdivide the data in the cluster by setting an appropriate threshold value for R^2.

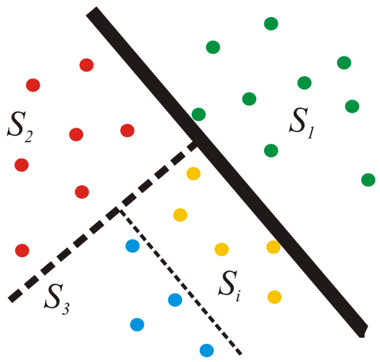


Figure 2.6: Hierarchical clustering. The original region S of the data is recursively split into a set of clusters S_i by subdividing the space using certain criteria.

Approximation of the Clustered Moving Least-Squares Surfaces

In order to save the online computations, the LS technique of [53] will be used to generate a polynomial function like Equation (2.6) for each cluster. So instead of obtaining a global bivariate function for each intrinsic parameter, several bivariate functions will be defined.

Suppose that S_i is the cluster to be modeled. In order to approximate S_i with a bivariate polynomial of degree m, it is required to obtain the regression coefficients vector using an equation similar to Equation (2.13)

$$\gamma_S = \left(\mathbf{\Psi}^T \mathbf{\Psi}\right)^{-1} \mathbf{\Psi}^T \hat{\theta}. \tag{2.20}$$

Here, the approximated points $\hat{\theta}_j$ obtained from Equation (2.14) are grouped into one vector $\hat{\theta}$ and are used to compute the regression coefficients vector γ_S of the cluster.

To ensure that the transition between S_i ant its neighboring clusters is smooth, a constraint must be enforced. Suppose that S_j is a neighboring cluster of S_i. The points that lie on the border between S_i and S_j must have the same values to avoid abrupt jumps in the model. This can be ensured by considering that the border points belong to both S_i and S_j. Thus, $\mathbf{\Psi}$ and $\hat{\theta}$ in Equation (2.20) have to be appended by the corresponding entries of the border points of the neighboring clusters. What remains is to find the degree m of each polynomial that will be used to fit S_i. In [53], a bivariate polynomial of degree 2 was used to fit the coefficient of the radial distortion and others of degree 5 to fit each of the coordinates of the principal point and the focal length components. The same thing is done here. To prevent overfitting of the data, however, the order of the fitting polynomial is reduced following Equation (2.8) depending on the number of points in S_i.

Table 2.2: The CMLS zoom lens modeling algorithm.

```
FOR all intrinsic parameters ($f_x$, $f_y$, $x_0$, $y_0$, $\kappa$) do the following:

1      Apply the clustering procedure described in Section 2.4.3.
2      Generate the surfaces (interpolate new setting points) within
       each cluster by applying the MLS algorithm in Table 2.1.
3      FOR each cluster
3.1        Calculate the LS approximation of the MLS
           surface by using polynomial functions of
           order $m = 5$ for $f_x$, $f_y$, $x_0$ and $y_0$ and a
           polynomial of order $m = 2$ for $\kappa$. Details
           are in Section 2.4.3.
3.2        Save the polynomial function of this cluster.
4      Combine all the saved polynomials into the model of the
       intrinsic parameter.
```

2.5 Results and Discussion

The first camera system used in the experiments is a PROSILICA (EC 1280C) CCD camera with a motorized PENTAX zoom lens (C6Z1218M3). The pixel size, as specified in the manual, is 6.7 μm in both directions of the pixel. As discussed in Section 2.2, the skew parameter σ is set to 0. The aperture setting is assumed to be constant and is fixed in this experiment to a value of $F = 5.6$ which corresponds to a proper opening for the lightning conditions of the conducted experiments.

The lens system's zoom range is between 12.5 mm - 75 mm and that of the focus is between ∞ - 1.2m. The zoom setting in motor units is varied between $0 - 700$ in steps of 100. This is equivalent to a change in focal length between 12.5 mm - 65 mm. For the focus, a range of 150 - 600 motor units with a step size of 50 is used, which roughly corresponds to the focused distance of the lens system. To avoid a hysteresis problem due to backlash, as mentioned in [53], the lens motor is driven to the desired setting by starting with smaller values for both, focus and zoom.

The measured set of data points, consisting of a total of 80 focus/zoom settings, is used to generate the models. For each lens setting, 40 images of a calibration grid with 64 points for calibration on it are taken from different fields of view. Due to the wide range of focal length, 6 checkerboards, different in size, are used for the calibration procedure. The calibration is done separately for each focus and zoom setting by treating each one as a single mono-focal lens.

To determine the intrinsic lens parameters at the chosen settings of focus and zoom, the Camera Calibration Toolbox for MATLAB is used [74]. This toolbox is mainly inspired from the classical calibration approach described in [78]. A slight difference resides in the closed-form estimation of the intrinsic parameters from the homographies and in the initialization of the distortion coefficients. These parameters are considered the ground truth data of the modeling process.

As a measure of accuracy, the pixel reprojection error or the Undistorted Image Plane Error (UIPE), as was called in [53], is used

$$\text{UIPE} = \sqrt{(x_i - \hat{x}_i)^2 + (x_i - \hat{x}_i)^2}, \tag{2.21}$$

where (x_i, y_i) are the measured coordinates the image point $\boldsymbol{p}_i$ and $(\hat{x}_i, \hat{y}_i)$ are the computed coordinates of the image point with the camera's intrinsic parameter model. A value UIPE $= 0$ means that the camera model has totally corrected the reprojection error of the pixel.

To make the proposed measure invariant to the number of data points involved, it is also suggested to use the Mean UIPE expressed as

$$\text{Mean UIPE} = \frac{1}{G}\sum_{i=1}^{G} \text{UIPE}, \tag{2.22}$$

where G is the number of calibration points multiplied by that of the calibration grids, i.e. $G = 64 \times 40 = 2560$ in our case. In addition, it is possible to define the Root Mean Squared Error (RMSE) of the pixels' reprojection error of the total number of focus/zoom settings N as

$$\text{RMSE} = \sqrt{\frac{1}{N \cdot G}\sum_{i=1}^{N \cdot G} \left((x_i - \hat{x}_i)^2 + (x_i - \hat{x}_i)^2\right)}. \tag{2.23}$$

To assess the correctness of the measured ground truth points, the plot of the Mean UIPE of the pixel reprojection error is given in Figure 2.7. The mean value of this curve, or the Mean Mean UIPE (MM_UIPE), is 0.08 pixels which reflects that the ground truth data is accurate. This value is of the same order of magnitude of that of the ground truth data used in [53].

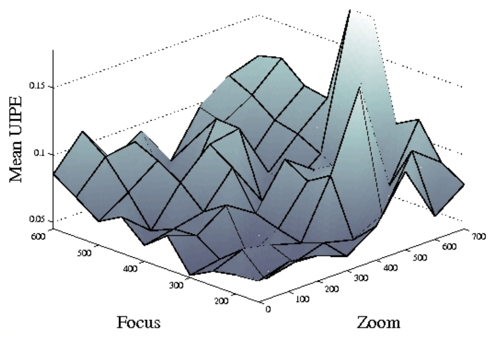
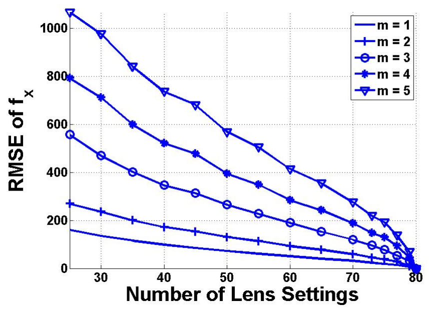


Figure 2.7: Mean UIPE of the measured ground truth data set of the PENTAX PROSILICA zoom camera system used in this work.

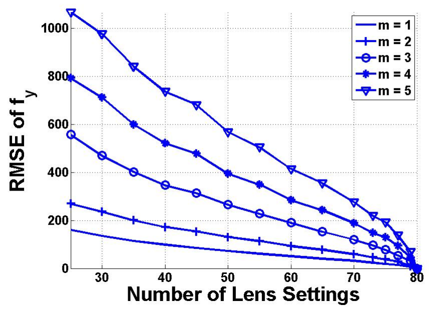
The first thing to be inquired is the best order m of each bivariate polynomial function that has to be used to model the intrinsic parameters of the PENTAX PROSILICA using MLS. For this, the *hold-out* cross validation test was performed [79]. A number of camera point settings varying between 1 and 55 are randomly chosen from the total of 80 points as the validation data. The remaining ones are retained as the training data. Each time, the training data is used to generate the models of the intrinsic parameters which are then tested by predicting the validation data. The limit 55 was chosen since a bivariate polynomial function of order 5 requires at least 21 neighboring points to be estimated since it is formed of 21 coefficients, see Equation (2.7). As a measure of significance, the RMSE of the pixel reprojection error is employed. Furthermore, the RMSE of the estimation of each intrinsic parameter is also computed using similar equations as in Equation (2.23); i.e. by replacing the pixels variables with the corresponding intrinsic parameters.

Figure 2.8 illustrates the outcome of the cross validation test for the polynomial orders 1 to 5. What can be noticed on the one hand is the sufficiency of a polynomial of order $m = 1$ to model each point setting of the focal length and the coordinates of the principal point. On the other hand, the coefficient of the radial distortion resulted in almost the same dependency with the five orders. When looking at the RMSE of the reprojection error in Figure 2.8f, the polynomial of degree 1 shows the lowest error. Thus, bivariate functions of order 1 will be utilized in the following to approximate the intrinsic parameters.

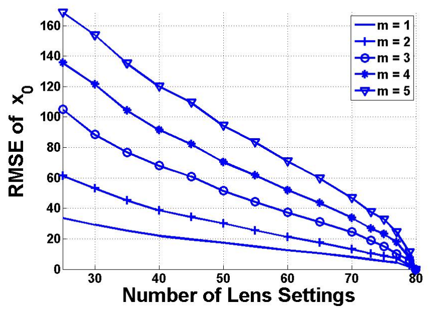
Figures 2.9 to 2.12 show the model of the intrinsic parameters of the used zoom lens camera system by employing the LS technique of [53] and the proposed MLS and CMLS techniques. The LS algorithm is applied by taking



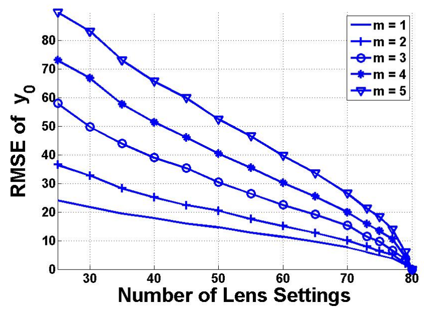
(a)



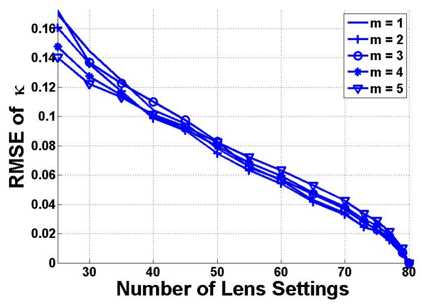
(b)



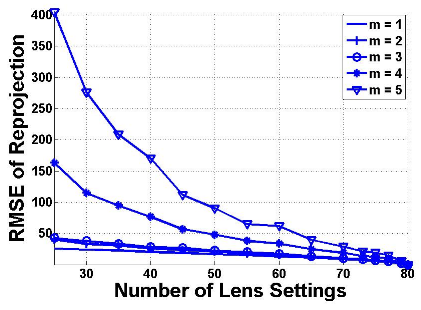
(c)



(d)



(e)

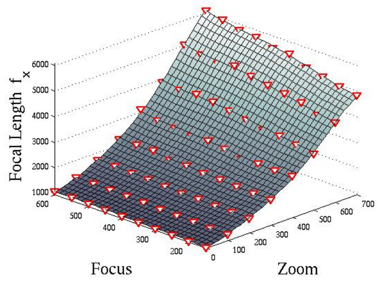


(f)

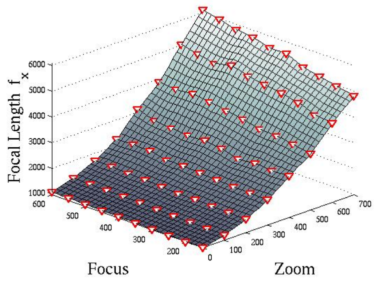
Figure 2.8: Evaluation of the MLS modeling algorithm of Section 2.4.2 with different orders of the bivariate polynomial functions. The polynomials of order 1 result clearly in the best modeling ability of MLS.

bivariate functions of order 2 to model the coefficient of the radial distortion and of order 5 to model each of the coordinates of the principal point and the components of the focal length as specified in [53]. The threshold coefficient of determination upon which the data in CMLS is clustered is set to 0.9.

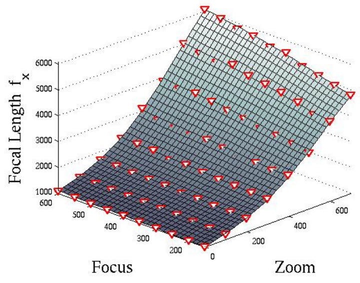
By examining the models of the focal length in Figure 2.9, it can be seen that the variation of this parameter is smooth along its domain. This is why all of the techniques have the ability to model its variation. Note that only the first component f_x is presented here since the second one f_y undergoes



(a) f_x with LS.



(b) f_x with MLS.

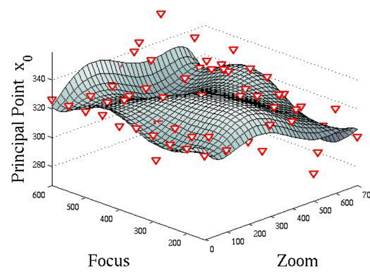


(c) f_x with CMLS.

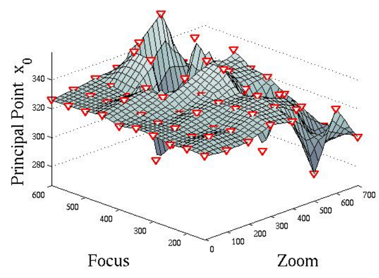
Figure 2.9: The focal length component f_x of the PENTAX PROSILICA camera system obtained with the different modeling algorithms. The measured points are marked with triangles.

almost the same behavior. In the obtained models of the coordinates of the principal point illustrated in Figures 2.10 and 2.11, however, the LS technique of [53] fails in capturing their variations when compared to both versions of the proposed method. The main reason is that the variable is very scattered by nature. Hence, a global bivariate polynomial of order 5 is not enough to reflect the model. To get a feeling of the scatterness of the data, the coefficient of determination R^2 defined in Equation (2.17) is computed for each of the intrinsic parameters while assuming the whole measurements as a single cluster. The outcome is illustrated in Table 2.3. The coefficient of determination of both components of the focal length is found to be 0.95 since they are smooth while those of the coordinates of the principal point are very close to zero due to their scatterness. Such a result justifies the employment of R^2 as a measure to cluster the data in CMLS. A similar conclusion is obtained with the curve approximation of the coefficient of radial distortion in Figure 2.12. In this case, the coefficient of determination R^2 is 0.62 which means it is in the middle region between being scattered and smooth.

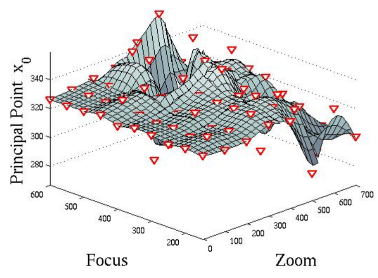
Compared to MLS, the CMLS approach leads to almost the same results. Nevertheless, some of the ground truth points do not lie on the approximation curve. This is because CMLS approximates each cluster with a single polynomial while MLS computes a polynomial for every point.



(a) x_0 with LS.



(b) x_0 with MLS.



(c) x_0 with CMLS.

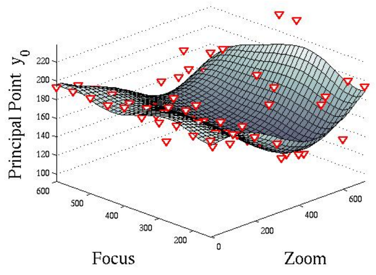
Figure 2.10: The coordinate of the principal point x_0 of the PENTAX PROSILICA camera system obtained with the different modeling algorithms. The measured points are marked with triangles.

Table 2.3: Variation of the coefficient of determination R^2.

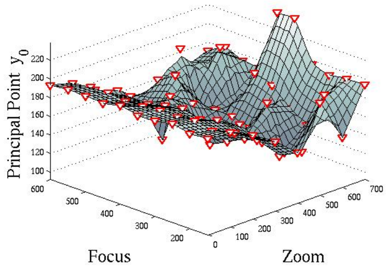
	f_x	f_y	x_0	y_0	κ
R^2	0.95	0.95	0.13	0.06	0.62

What is left to be appraised is the robustness of the algorithms against the number of measured focus and zoom sample points. The importance of this test is to inquire the minimal number of measurement points required to obtain an accurate model of the intrinsic parameters, i.e. low pixel reprojection errors. A powerful test that provides this measure is again the *hold-out* cross validation using the RMSE of the pixel reprojection as a significance measure. The length of the validation data is varied between 1 and 55 in the LS method since the coordinates of the principal point are modeled with a function of order 5 [53]. That of the MLS and CMLS algorithms are varied between 1 and 70 since here only polynomials of order 1 will be used by MLS to generate complete surfaces of the intrinsic parameters. This will provide CMLS enough number of points to approximate the models even with bivariate functions of order 5.

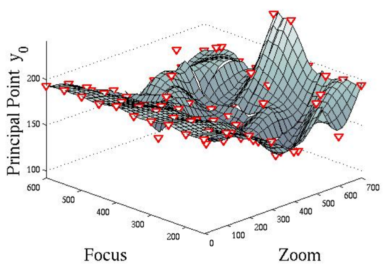
Figure 2.13 shows the outcome of the experiment. When compared to the MLS based methods, the LS has the worst performance due to the global type



(a) y_0 with LS.



(b) y_0 with MLS.

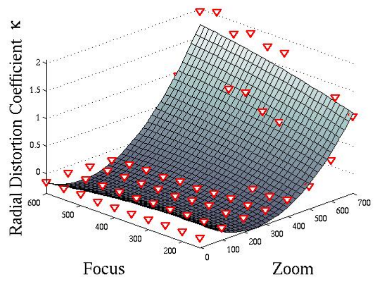


(c) y_0 with CMLS.

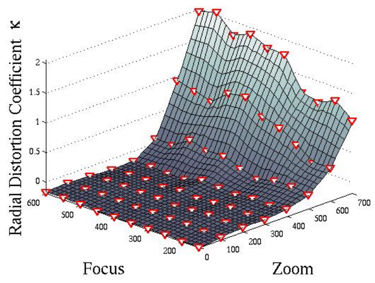
Figure 2.11: The coordinate of the principal point y_0 of the PENTAX PROSILICA camera system obtained with the different modeling algorithms. The measured points are marked with triangles.

of the solution. Using only 10 measurements of the focus/zoom settings, the RMSE of both MLS methods is 40, which is attained with 38 measurement points using LS. This means by applying the MLS based techniques, it is possible now to save 70% of the measurement points needed to estimate the model, i.e. 10 measurement points equally distributed along the zoom range to get a rough estimate of the zoom camera model of the intrinsic parameters. This shows that the proposed method has a very good interpolation capability which makes it require less points. This capability is obtained because MLS finds a global fit of the intrinsic parameters by concatenating several local fits. Hence, the computed surface will be more adapted to the variation of the data. This result is of major importance since each focus/zoom setting point requires numerous measurements of the calibration grid followed by an offline calibration algorithm.

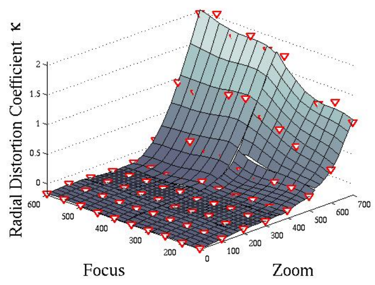
MLS leads to the most accurate model of the intrinsic of a zoom camera system. After a certain number of samples, the RMSE of MLS drops a lot faster than that of CMLS. This is due to the same reason that was earlier described. Its only disadvantage is its prerequisite for more computational power than the CMLS and LS approaches. To determine the intrinsic parameters at each focus/zoom setting, MLS needs to search for the nearest



(a) κ with LS.



(b) κ with MLS.



(c) κ with CMLS.

Figure 2.12: The coefficient of the radial distortion κ of the PENTAX PROSILICA camera system obtained with the different modeling algorithms. The measured points are marked with triangles.

neighbors, an operation which has a complexity of $O\left(N^{\frac{1}{2}} + k\right)$ followed by the evaluation of the distances, and then the minimization of Equation (2.10).

By employing CMLS, the amount of computations required in an online system is nothing but the time needed to evaluate 5 bivariate functions corresponding to the 5 intrinsic parameters as in LS. The only difference between the two is the number of polynomial functions that has to be stored. In the case of the PENTAX PROSILICA zoom camera system used in these tests, the number of these functions is shown in Table 2.4. The number of functions to be saved for the focal length is similar to that of LS since this parameter has smooth variation, i.e. $R^2 = 0.95$. In the case of the coordinates of the principal point, this number goes to 16 due to the high variation of the parameter while that of the radial distortion is 9. This result suggests that CMLS adapts to the variation of the parameters. In case of a smooth variable, it leads to a similar result as LS while it results in many LS solutions if the variable is scattered, e.g. principal point and radial distortion.

In order to validate our results, the proposed algorithms are applied to another camera system. A PENTAX motorized zoom lens, of the same model as the one previously used, is integrated to a DOLPHIN F145C IRF camera. The PENTAX DOLPHIN system was calibrated and then the modeling algorithms were applied in the same manner as was done to the previous system. The visual results reflected in Figures 2.14 to 2.17 emphasize once again on the accuracy of the derived methods.

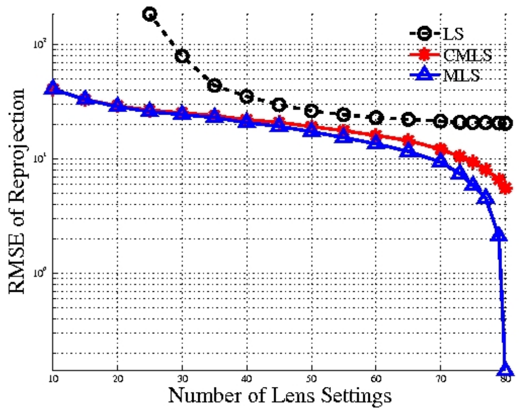


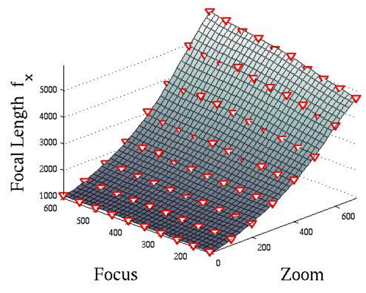
Figure 2.13: Dependency of the RMSE of the pixel reprojection on the number of used lens settings for the different modeling techniques.

Table 2.4: The number of bivariate functions to be stored for the LS algorithm of [53] and the proposed CMLS technique.

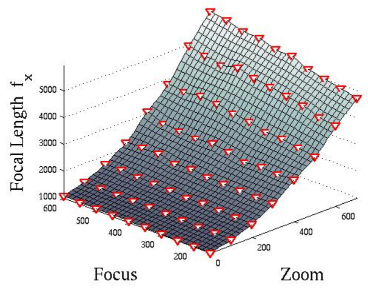
	f_x	f_y	x_0	y_0	κ	TOTAL
LS	1	1	1	1	1	5
CMLS	1	1	16	16	9	43

2.6 Summary

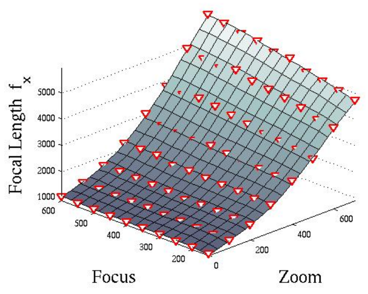
A new technique was proposed in this chapter, MLS, to calibrate an automatic zoom camera. MLS locally fits the intrinsic parameters with some polynomial functions. Compared to previous methods, MLS was able to increase the fitting accuracy of the variables. The cause of this improvement is the ability of the proposed technique to better fit the scattered data due to the concatenation of several local regression models. Moreover, the proposed method was less sensitive to the number of measured focus and zoom settings since it has a better ability to interpolate the missing data. To reduce the computational complexity of MLS, the CMLS algorithm was derived which clusters and approximates the MLS curves with several polynomial functions. Compared to MLS, CMLS requires very simple computation in a TPTA scenario; however, the latter results in more accurate values of the intrinsic parameters. The obtained results motivate the application of the MLS based methods in TPTA without imposing any constraints on the teleoperator.



(a) f_x with LS.

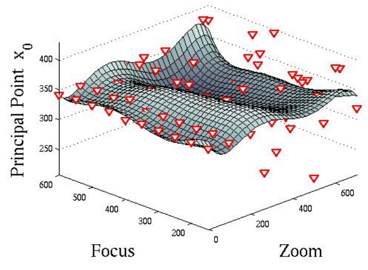


(b) f_x with MLS.

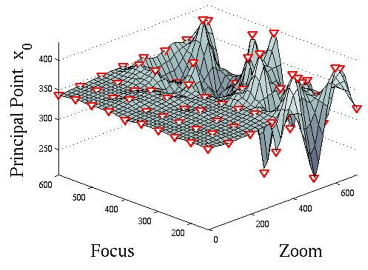


(c) f_x with CMLS.

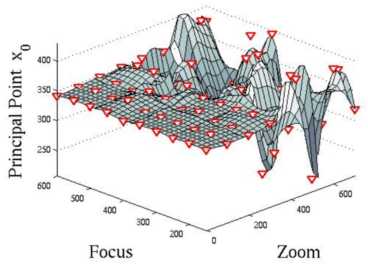
Figure 2.14: The focal length component f_x of the PENTAX DOLPHIN camera system obtained with the different modeling algorithms. The measured points are marked with triangles.



(a) x_0 with LS.

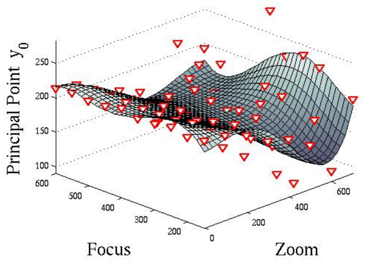


(b) x_0 with MLS.

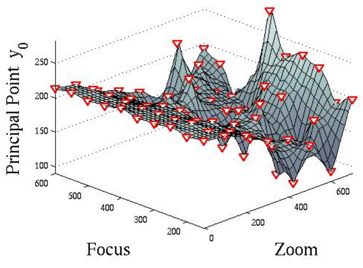


(c) x_0 with CMLS.

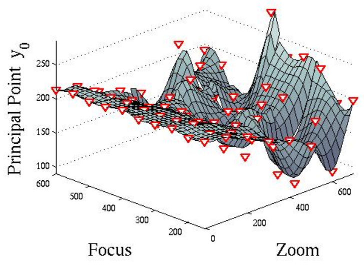
Figure 2.15: The coordinate of the principal point x_0 of the PENTAX DOLPHIN camera system obtained with the different modeling algorithms. The measured points are marked with triangles.



(a) y_0 with LS.

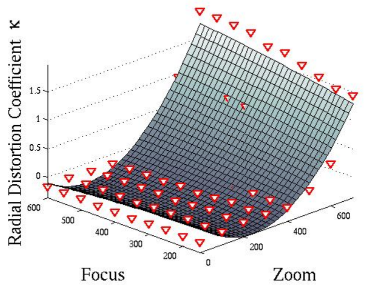


(b) y_0 with MLS.

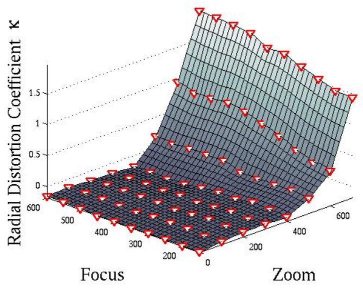


(c) y_0 with CMLS.

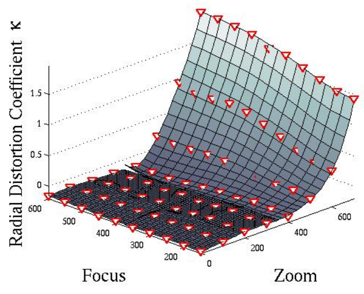
Figure 2.16: The coordinate of the principal point y_0 of the PENTAX DOLPHIN camera system obtained with the different modeling algorithms. The measured points are marked with triangles.



(a) κ with LS.



(b) κ with MLS.



(c) κ with CMLS.

Figure 2.17: The coefficient of the radial distortion κ of the PENTAX DOLPHIN camera system obtained with the different modeling algorithms. The measured points are marked with triangles.

Chapter 3

Meshing the Stereo Images

Rendering the scene on the virtual display is one of the most important steps in a TPTA system. The rendering process should be performed with a very minimal delay to let the operator incur the feeling of being *immersed* in the virtual world. The speed of this process is affected by the size of the generated 3D model. The larger the model is, the more are the delays introduced to both the rendering task and the network transmissions. The size of the 3D model is measured using the number of the triangles in the mesh approximation of the scene. It is also related to the size of the images of the stereo cameras integrated to the teleoperator. Taking a 640×480 VGA camera for example, there are 307200 pixels that have to be processed and which lead each frame to a mesh of more than 600000 triangles. These figures will naturally increment with cameras of higher resolution. To reduce the size of the mesh and accelerate the stereo based scene acquisition, surface simplification algorithms must be applied. A simplification scheme can be applied in a TPTA scenario, however, as long as it does not significantly increase the latency of scene reconstruction. This dilemma is the main motivation behind addressing the topic in this thesis.

3.1 Overview of Surface Simplification

The aim of a surface simplification algorithm is to reduce the size of a mesh without deteriorating the details of the corresponding scene. Some of these methods reduce the number of the 3D points before constructing the mesh as in [82, 73, 80, 83]. In such a case, the points are first clustered into several groups by employing methods like k-nearest neighbors, binary space partitions or quadtrees. Representative points of each cluster are then computed upon which the approximation mesh of the scene is constructed. The flow

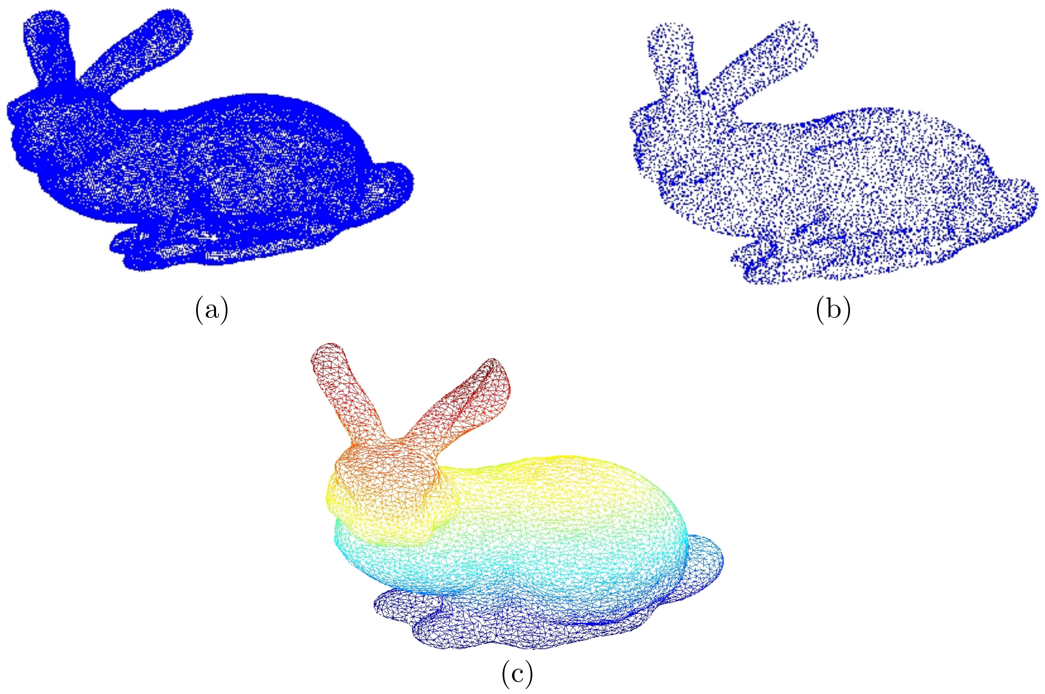


Figure 3.1: Example of a 3D meshing scheme by applying the method of [80] to the Stanford Bunny [81]. (a): The dense 3D points. (b): The reduced set of the points. (c): The 3D mesh approximation of the Stanford Bunny using the reduced set of points.

of such algorithms is perhaps better understood by looking at the example shown in Figure 3.1 using the Stanford Bunny [81]. Other approaches store a primary mesh of the scene in the memory which is formed by interconnecting all the pixels in the image. The mesh is then reduced with the help of the depth information using decimation algorithms as the ones proposed in [4, 84, 85, 86]. The objective of such techniques is to reduce the number of triangles in the areas where there are no significant variations in the surface of the scene and preserve the ones where high variations occur.

An alternative methodology is to reduce the number of points directly in the image by choosing the pixels that are necessary to recover its content. These pixels are then used with their depth values to construct the 2.5D mesh of the scene. The pixels, which are also the nodes of the mesh, should be selected in such a way that they are able to recover all the other pixels in the image with minimal error. Such methods are called the content adaptive mesh approximation techniques of images, the goal of which is to obtain a realistic visualization of an image using a mesh, e.g. see [28, 87, 88, 89, 90]. The nodes of the mesh form *the non-uniform samples* of the image since they are sampled in an irregular manner. An example is illustrated in Figure 3.2.

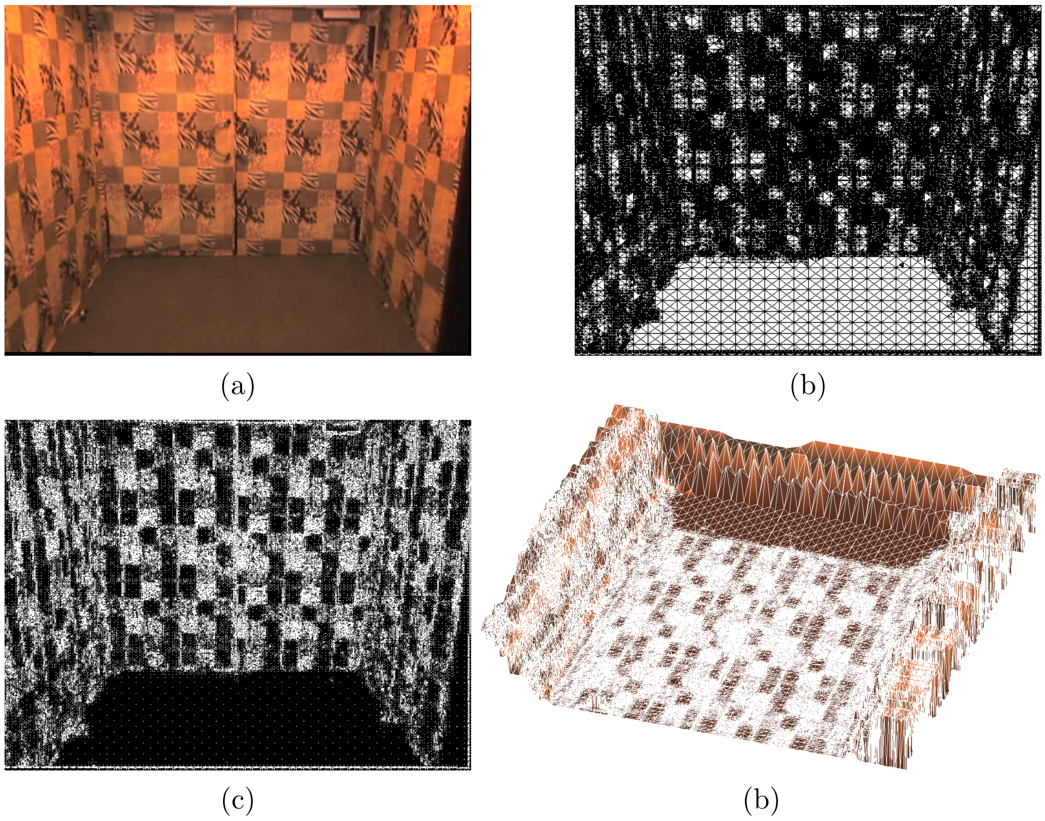


Figure 3.2: Sample result using the proposed 2.5D meshing algorithm. (a): The sample image. (b): The 2D mesh obtained by the tritree meshing scheme of Section 3.4.2. (c): The non-uniform samples of the sample image (nodes of the mesh). (d): The 2.5D mesh of the scene; the depth values are obtained via the sparse dynamic programming stereo matching technique described in Chapter 4.

Any algorithm from these groups can be used in a TPTA scenario as long as it satisfies the low delay constraint. In this chapter, the interest lies in the direction of the third set of methods. This is justified from the fact that these techniques perform the reduction directly in the image. This will allow us to employ in the next chapter the sparse stereo matching strategy to estimate the depth of the pixels for only the non-uniform samples of the image are required to generate the mesh approximation of the scene. Therefore, stereo matching will be accelerated because the number of pixels that has to be computed in 3D is reduced. The combination of content adaptive mesh representation of images along with sparse stereo to construct the 3D scenes is one of the new ideas presented in this thesis since the surface simplification problem and the stereo matching problem are usually treated independently. This point will be clarified in the next chapter. In this chapter, the focus will be on representing an image with a mesh that preserves its content.

3.2 Related Work

Representing an image with a triangular mesh is a major research area due to the wide range of its applications in image processing, computer vision and computer graphics. The aim of this field is to allow a realistic visualization of an image by using an approximating mesh. This is usually achieved by removing the redundant pixels from the image and retaining only the ones with the most of the information, see Figure 3.2. Then, a mesh is generated from the remaining pixels using a triangulation scheme.

The advantage of such approximation is that it allows the image to have a more compact representation. The approximating or representing mesh, usually possesses a lot less number of pixels than the original image and which are designated as the significant pixels or the mesh nodes. Getting back to Figure 3.2, the number of pixels used to construct the mesh forms around 40% of the total pixels in the image. These pixels are used in combination with the generated mesh to recover the intensity of the other pixels using some interpolation schemes [25, 90, 87]. Such a representation has proved to be useful in various applications as in image compression and coding [91,88,92, 93,87], image processing for medical application [94,95], super resolution [96], image authentification [97,98] and computer vision [25,31,99,100].

The major issue in this research is in the methodology that should be used to find the lowest possible number of significant pixels in an image while preserving its content. As a consequence, the aim becomes equivalent to finding the *non-uniform samples* of the image [89,90,101,102]. This makes the methods developed for progressive image coding and transmission, as in [103,101,104,105], relevant since non-uniform sampling is in the core of the implemented schemes. These can then be easily followed by a triangulation method to generate the adaptive mesh as done in [89,90] for example. Other techniques that can also be equivalently used in determining the non-uniform samples are based on finding the verge points of an image [106, 107].

The work of [90] derives a content adaptive mesh approximation technique for an image by finding its non-uniform samples. The developed method has shown in the analytical comparisons with several other approaches as the ones proposed in [108, 88, 89] to have a better performance in terms of quality and speed. The work derived in [101], which concentrates on the adaptive non-uniform sampling for image coding and transmission, presents also a good candidate for mesh approximation since it can be followed by a triangulation method to generate the mesh. This scheme was also the basis of other developed techniques as the one suggested in [105]. These two algorithms will be briefly introduced in the sequel since they will be later used in the comparisons with the approach that will be developed.

- **Yang's algorithm:** In [90], a pixel is chosen to be a non-uniform sample if its second directional derivative is significant. Let $\Xi(x, y)$ denote the magnitude of the second derivative of the intensity $I(x, y)$ of each pixel

$$\Xi(x, y) =_{\varphi \in [0, 2\pi]}^{\max} \left| I''_{\varphi}(x, y) \right|, \tag{3.1}$$

where $\varphi \in [0, 2\pi]$ is the direction of the second derivative of the image function. Each pixel that possesses a value $\Xi(x, y)$ above a predefined threshold is put into a feature image. Then a modified version of the classical Floyd-Steinberg algorithm is used to diffuse the error of a feature pixel to its neighbors. As experiments showed, this technique works better if the serpentine raster order was used instead of the standard raster order diffusion method. The reason for the improvement is that the errors are propagated in a more balanced manner between a pixel and its neighbors. After that, Delaunay triangulation follows to generate the content adaptive mesh of the image.

- **Ramponi's algorithm:** In [101], the main issue is to derive a non-uniform sampling technique that can be used for progressive image coding. Given an image, the initial set of the non-uniform samples is determined by first computing the skewness of the pixels. The skewness sk of a pixel evaluated on a $w \times h$ mask is defined as

$$sk(x, y) = \frac{1}{w \cdot h} \sum_{i=1}^{w} \sum_{j=1}^{h} \left(I(i, j) - \upsilon(i, j) \right)^3, \tag{3.2}$$

where $\upsilon(\cdot)$ is the mean of the intensity values of the pixels in the mask. To make this measure insensitive to the dynamic range of the image, it is better to normalize Equation (3.2) as

$$\bar{sk}(x, y) = \frac{|sk(x, y)|}{\max(|sk(x, y)|)}, \tag{3.3}$$

where $\bar{sk}$ is the normalized skewness and the denominator is the maximum skewness obtained in the image. After that, a threshold is defined and each pixel that has a normalized skewness $\bar{sk}(x, y)$ higher than the threshold is considered as a non-uniform sample. To further reduce the number of the obtained samples and get a more compact representation of the image, the initial set is decimated by defining a forbidden circular area around each sample. Therefore, if a sample belongs to the area of another one, it is removed from the set.

3.3 Problem Formulation

The goal of content adaptive mesh representation is to represent an image with an approximating mesh. The nodes of the mesh are the non-uniform samples and they can be used to recover the original image with minimal error. Non-uniform, irregular or adaptive sampling is the process of sampling the signal at different rates. Uniform or regular sampling is the process of sampling the signal at a uniform rate. Both sampling techniques have to obey the Nyquist sampling theorem, but the non-uniform sampling takes in addition the statistics of the signal into account. The sampling rate is adapted to the variation of the intensities in the image, i.e. it is high in the regions of the image where there are a lot of intensity variations; otherwise, it is low. The difference between non-uniform and uniform sampling can be visualized by looking at the non-uniform samples of Figure 3.2a in Figures 3.2c and the uniform samples in Figure 3.3. Figure 3.2c preserves the structure of the image while Figure 3.3 simply leads to a smaller version of the image.

In most of the developed methods, the content adaptive mesh is constructed in two steps. The non-uniform samples are first extracted then the mesh is formed upon them. This makes the mesh relatively dependent on the samples found since reconstructing the missing pixels using the mesh highly depends on the ability of each triangle of the mesh to represent and recover the pixels lying within. A more convenient way would be by finding the non-uniform samples that best describe the nodes of the approximating mesh; hence, making the samples found depend on the structure of the mesh and not vice versa. Consequently, the main objective becomes equivalent to finding the vertices of all the triangles in an image which can interpolate the intensities of the pixels lying within up to a predefined error.

In order to that, it is necessary to have a quantity that measures the error between the intensity values $I(x, y)$ of the original image and those of the approximated image $\hat{I}(x, y)$ interpolated using the nodes of the mesh[1]. A well known measure is the peak signal to noise ratio (PSNR), see [109], which is defined as

$$\mathrm{PSNR} = 10 \log \left(\frac{255^2}{\mathrm{MSE}} \right). \tag{3.4}$$

MSE is the mean squared error in the approximated pixels. It can be defined over the area of a triangle T by

$$\mathrm{MSE} = \frac{1}{\lfloor \triangle T \rfloor} \sum_{i=0}^{\lfloor \triangle T \rfloor - 1} \Big(I\left(x_i, y_i\right) - \hat{I}\left(x_i, y_i\right) \Big)^2, \tag{3.5}$$

[1]The derivations made in this work assume gray-scale intensity images. In the case of color images, similar derivations apply but have to be done using all the color planes.

Figure 3.3: The uniform sampling grid of Figure 3.2a taken at every third sample.

where $\triangle$ is a symbol denoting the area of a triangle, i.e. $\triangle T$ is the area of T, and (x_i, y_i) are the coordinates of a pixel in T. It is also possible to use the MSE as a quality measure instead since it is equivalent to PSNR.

Equation (3.4) presents the cost function that has to be maximized over all the triangles of the mesh. What remains is to find a methodology that will allow us to reconstruct the intensity values of the pixels inside each triangle. To derive it, it is necessary to describe the geometry of the problem in hand and which is illustrated in Figure 3.4. The points of an image describe a 3D space represented by the 2D coordinates of the pixel in the image and the corresponding intensity. Each triangle is formed by three points which are parts of the nodes or the vertices of the mesh. In a similar manner, a plane can also be defined by the three vertices of the triangle. Let $\boldsymbol{v}_i(x_i, y_i, I_i)$ with $i = 1, 2, 3$ be the three vertices of a triangle T under consideration. The plane describing these vertices is defined using the normal equation as

$$\vec{\eta} \cdot \boldsymbol{p}_I + \eta_4 = 0, \tag{3.6}$$

where $\boldsymbol{p}_I$ denotes a pixel with coordinates (x, y) and intensity value I lying on the plane, $\vec{\eta} = \left[\begin{smallmatrix}\eta_1 \\ \eta_3 \\ \eta_3\end{smallmatrix}\right]$ is the vector normal to the plane and η_4 is the distance from the origin such that $\eta_4 = -\vec{\eta} \cdot \boldsymbol{v}_i$. The vector $\vec{\eta}$ can be computed with the three vertices $\boldsymbol{v}_i$ as the cross product of any of the two edges of the triangle as

$$\vec{\eta} = [\boldsymbol{v}_1 - \boldsymbol{v}_2] \times [\boldsymbol{v}_3 - \boldsymbol{v}_2]\,. \tag{3.7}$$

Consequently, the equation of the plane is directly now obtained as

$$\eta_1 x + \eta_2 y + \eta_3 I + \eta_4 = 0. \tag{3.8}$$

Therefore, it is possible now to recover the intensity value $\hat{I}$ of a pixel lying inside the triangle T by rewriting Equation (3.8) in the form

$$\hat{I} = -\left(\eta_1 x + \eta_2 y + \eta_4\right)/\eta_3. \tag{3.9}$$

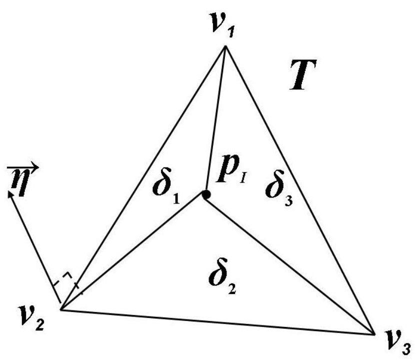


Figure 3.4: Geometrical setting of the problem.

Note that it is also possible to define the reconstructed intensity value $\hat{I}$ of $\boldsymbol{p}_I$ in a different way by considering it as the weighted sum of the intensity values of the vertices of the triangle

$$\hat{I} = \sum_{i=1}^{3} \delta_i \cdot I(\boldsymbol{v}_i), \tag{3.10}$$

where $I(\boldsymbol{v}_i)$ is the intensity value of the vertex $\boldsymbol{v}_i$ and δ_i is the weight assigned to $\boldsymbol{v}_i$. The weights in this case have to reflect the distance of the point $\boldsymbol{p}_I$ from the vertices of the triangle. In the special case where the weights δ_i satisfy the following constraints

$$\boldsymbol{p}_I = \sum_{i=1}^{3} \delta_i \cdot \boldsymbol{v}_i \text{ such that } \sum_{i=1}^{3} \delta_i = 1, \tag{3.11}$$

they are called the barycentric coordinates of $\boldsymbol{p}_I$ in the triangle T [110]. The barycentric coordinates are also known as the areal coordinates since they are also described by the ratio of the *signed* areas of the triangles formed by $\boldsymbol{p}_I\boldsymbol{v}_1\boldsymbol{v}_2$, $\boldsymbol{p}_I\boldsymbol{v}_2\boldsymbol{v}_3$ and $\boldsymbol{p}_I\boldsymbol{v}_3\boldsymbol{v}_1$ to that of the triangle $\boldsymbol{v}_1\boldsymbol{v}_2\boldsymbol{v}_3$, see Figure 3.4 for an illustration of the concept. The barycentric coordinates can be written as

$$\delta_1 = \frac{\triangle \boldsymbol{p}_I\boldsymbol{v}_1\boldsymbol{v}_2}{\triangle \boldsymbol{v}_1\boldsymbol{v}_2\boldsymbol{v}_3}, \quad \delta_2 = \frac{\triangle \boldsymbol{p}_I\boldsymbol{v}_2\boldsymbol{v}_3}{\triangle \boldsymbol{v}_1\boldsymbol{v}_2\boldsymbol{v}_3} \text{ and } \delta_3 = \frac{\triangle \boldsymbol{p}_I\boldsymbol{v}_3\boldsymbol{v}_1}{\triangle \boldsymbol{v}_1\boldsymbol{v}_2\boldsymbol{v}_3}. \tag{3.12}$$

Theorem 3.1 *The weights imposed by Equation (3.9) to reconstruct the intensity of the point $\boldsymbol{p}_I$ in the plane are equivalent to the barycentric coordinates of $\boldsymbol{p}_I$ in the triangle T defined by the vertices $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$.*

Proof To prove the equivalence of the barycentric coordinates of $\boldsymbol{p}_I$ to the weights imposed by Equation (3.9), it is necessary to reformulate the equation of the plane defined by T. Let us define the terms of the cross product in

Equation (3.7) as $\overrightarrow{\boldsymbol{v}_2\boldsymbol{v}_1} = \boldsymbol{v}_1 - \boldsymbol{v}_2$ and $\overrightarrow{\boldsymbol{v}_2\boldsymbol{v}_3} = \boldsymbol{v}_3 - \boldsymbol{v}_2$. The parametric equation of the plane shown in Equation (3.8) can be now written as

$$\boldsymbol{v}_2 + x\overrightarrow{\boldsymbol{v}_2\boldsymbol{v}_1} + y\overrightarrow{\boldsymbol{v}_2\boldsymbol{v}_3} = x\boldsymbol{v}_1 + (1 - x - y)\,\boldsymbol{v}_2 + y\boldsymbol{v}_3. \tag{3.13}$$

By comparing Equations (3.11) and (3.13), it is easy to notice the correspondence between the weights by setting $\delta_1 = x$, $\delta_2 = 1 - x - y$ and $\delta_3 = y$. Hence, the equivalence is proved. ■

As a result of this proof, it is possible now to reconstruct the intensities of the pixels lying inside a triangle by simple computation of the barycentric coordinates of the pixel in the triangle or using the planar equation since they are equivalent. In addition, one can check how does each triangle represent the pixels which lie within by simple computation of the PSNR described in Equation (3.4). If the PSNR of the reconstructed intensities of the pixels lying inside the triangle T is lower than a predefined threshold, then T should be further decomposed into two smaller triangles.

3.4 BSP based Image Mesh Representation

Given an input image, the objective of this section is to construct an adaptive mesh that approximates the original image with the lowest number of non-uniform samples while preserving its contents using the formulas derived in the previous section. The proposed method is based on the binary space partitions (BSP) principle, see [111], which clusters a data set recursively as was formerly depicted in Figure 2.6. In each step, one cluster is divided into two sub-clusters depending if some predefined criteria are not met. The cluster in our case is a triangle in the mesh. As a starting point, an image is first divided into two triangles along one of the diagonals. Then, each of the two triangles is recursively split into smaller triangles if it does not satisfy the criteria. The criteria that will be used are the maximum number of points that the triangle can hold and the minimum acceptable threshold ϵ of the reconstruction error of the pixel intensities, i.e. PSNR. The flowchart of the proposed BSP based content adaptive meshing algorithm is illustrated in Figure 3.5 and the rest of this section is dedicated to explain it.

3.4.1 Determining the Inlying Points of a Triangle

The first thing to assess is to find the pixels that belong to the triangle T under consideration. All the pixels inside the triangle have in common that they lie on the same side of each of the triangle's edges in a similar manner

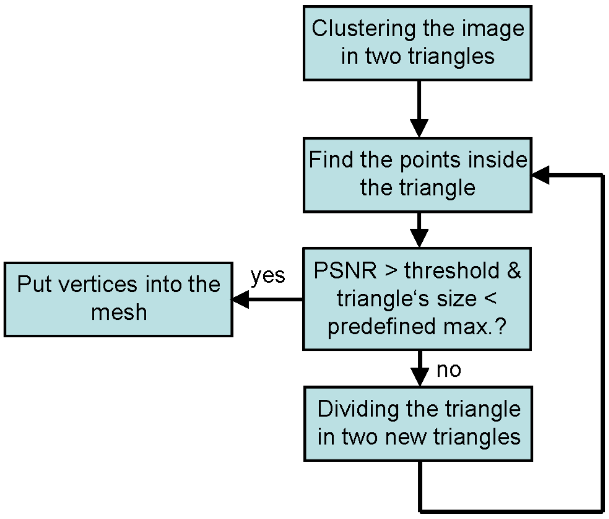

Clustering the image in two triangles
Find the points inside the triangle
PSNR > threshold & triangle's size < predefined max.?
yes
Put vertices into the mesh
no
Dividing the triangle in two new triangles

Figure 3.5: Flow chart of the BSP based content adaptive meshing technique.

to the barycenter of T. In order to determine if a point lies inside T, it is sufficient to verify this property at each point using the following theorem.

Theorem 3.2 *A point* $\boldsymbol{p}_I$ *is said to be lying inside a triangle* T *of the mesh if all the barycentric coordinates of* $\boldsymbol{p}_I$ *with respect to* T *are positive.*

Proof Theorem 3.2 is a direct consequence to the definition of the barycentric coordinates in Equation (3.12). The barycentric coordinates of $\boldsymbol{p}_I$ in the triangle T are proportional to the signed areas of the smaller triangles $\boldsymbol{p}_I\boldsymbol{v}_1\boldsymbol{v}_2$, $\boldsymbol{p}_I\boldsymbol{v}_2\boldsymbol{v}_3$ and $\boldsymbol{p}_I\boldsymbol{v}_3\boldsymbol{v}_1$, see Figure 3.4 for a reminder on the problem setup. To prove the theorem, it is necessary to show that the signed areas of the three smaller triangles of a point $\boldsymbol{p}_I$ lying inside T are positive. Taking the triangle $\boldsymbol{p}_I\boldsymbol{v}_1\boldsymbol{v}_2$ for example and by denoting the angle formed by $\widehat{\boldsymbol{v}_2\boldsymbol{p}_I\boldsymbol{v}_1}$ to be α. The area of the triangle is given by

$$\triangle\boldsymbol{p}_I\boldsymbol{v}_1\boldsymbol{v}_2 = \frac{1}{2}\left\|\boldsymbol{p}_I\boldsymbol{v}_1\right\| \cdot \left\|\boldsymbol{p}_I\boldsymbol{v}_2\right\| \cdot \sin\left(\alpha\right), \tag{3.14}$$

where $\|\cdot\|$ denotes the norm of the vector. From this equation, it is easy to notice that the signed area of the triangle is positive if the sine of the angle α is positive. This is satisfied only if the point $\boldsymbol{p}_I$ lies inside the triangle assuming the positive direction to be counter clockwise. A similar demonstration can be done for the other triangles. ■

Corollary 3.3 *One can infer from the proof of Theorem 3.2 that a point* $\boldsymbol{p}_I$ *lies outside the triangle* T *if any of its barycentric coordinates is negative while it lies on an edge of the triangle if any of the coordinates is zero.*

3.4.2 Computing the Partition Line

If the PSNR of a triangle is less than the predefined threshold or the number of points is larger than the specified maximum size of the triangle, it has to be divided into two new triangles. In order to let BSP decide how a triangle should be divided, three methods will be discussed.

The Tritree Subdivisions

Tritree subdivisions consists of dividing an area into a triangular grid [112]. A triangle is first generated that includes all the area to be triangulated. Then, it is subdivided into smaller triangular regions until all the area of interest is meshed. This technique has been applied to compress images in [92]. In this work, a different strategy is used to allow the formulation of the recursive subdivisions using BSP. An image is first divided into two triangles along one of its longest diagonals. Then, each triangle is recursively split until the criteria that were earlier presented are satisfied. For simplicity, the difference between the original tritree method and the proposed one is illustrated in Figure 3.6. Here, each triangle is divided from its longest edge. The new vertex $\boldsymbol{v}_n$, hence the non-uniform sample, is nothing but the middle point of this edge. Let $\boldsymbol{v}_o$ be the opposite vertex of the triangle's longest edge. The partition line that splits the triangle is given by

$$\boldsymbol{v}_o + \beta_p(\boldsymbol{v}_n - \boldsymbol{v}_o), \tag{3.15}$$

where β_p is the coefficient of the division line that can be determined using the coordinates of $\boldsymbol{v}_n$ and $\boldsymbol{v}_o$.

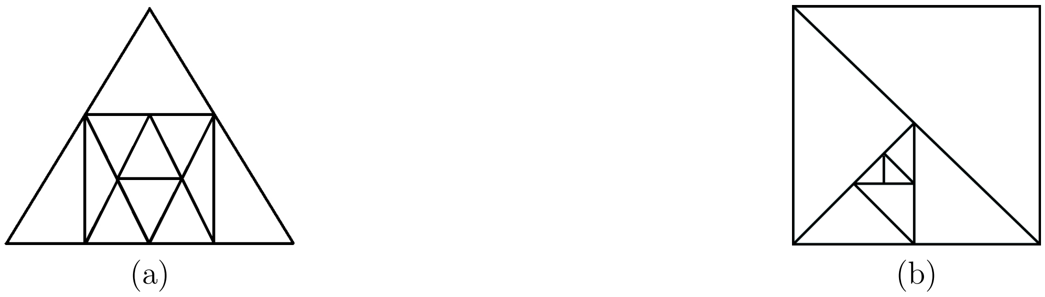


Figure 3.6: Sketch illustrating the tritree subdivisions applied to an image. (a): The algorithm of [112]. (b): The proposed tritree partitioning scheme.

The tritree subdivisioning scheme is very simple to implement. Each triangle is divided into two smaller triangles of the same size. Tritree is similar in spirit to the quadtree subdivisioning method that was used in [113, 88] to represent an image with a mesh. There, an image is subdivided into rectangles which can be then divided into two or more triangles depending on the error criterion, i.e. MSE or PSNR, and hence obtaining the mesh approximation of the image. Thus, it seams more practical to start directly building a tree of triangles instead of generating a tree of rectangles and transforming it after that into a tree of triangles.

Kmeans Clustering

Kmeans is a method that divides a data set into k clusters by finding the center of each of them [75]. In this work, the interest resides in dividing a triangle into two smaller ones and thus $k = 2$. The aim of kmeans is to minimize the variance in each of the sought clusters by minimizing the following error function

$$\min \sum_{i=1}^{2} \sum_{\boldsymbol{p}_{Ij} \in S_i} \left\| \boldsymbol{p}_{Ij} - \boldsymbol{c}_i \right\|^2 , \tag{3.16}$$

where S_i represents a cluster, $\boldsymbol{c}_i$ is the center of each cluster and $\boldsymbol{p}_{Ij}$ is a pixel in S_i. In general, the kmeans algorithm starts by randomly initializing two cluster centers in the triangle and by assigning each point of the triangle to its nearest cluster center. Then, a new center is recomputed for each cluster and the points are reassigned to the new centers. This process is repeated until there is no change in the pixels' reassignment.

The drawback of kmeans is the possibility to get different results on the same image since the initial cluster centers are usually randomly selected [75]. The alternation of the results may lead to a degradation of the quality of the mesh approximation especially if kmeans does not converge to a solution. To overcome this burden, kmeans is modified in this work by initializing the first two cluster centers as the vertices of the longest edge of the triangle. In this case, the first two clusters will be similar to the result of the tritree subdivisions. After several iterations, however, the clusters will adapt to the variation of the pixels inside the triangle.

Suppose that $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the two computed centers. The direction vector of the partitioning line $\vec{\ell}$ is nothing but the normal to the mid-point $\boldsymbol{p}_m$ of the vector $\overrightarrow{\boldsymbol{c}_1\boldsymbol{c}_2}$ connecting to the two centers. This can be illustrated in Figure 3.7. Since $\overrightarrow{\boldsymbol{c}_1\boldsymbol{c}_2}$ is perpendicular to the optimal partition line, it is necessary to find which vertex $\boldsymbol{v}_o$ of the three vertices of the triangle minimizes the inner product $\left\langle \overrightarrow{\boldsymbol{v}_o\boldsymbol{p}_m}, \overrightarrow{\boldsymbol{c}_1\boldsymbol{c}_2} \right\rangle$. Consequently, the direction $\vec{\ell}$ of the partition line is

$$\vec{\ell} = \overrightarrow{\boldsymbol{v}_o\boldsymbol{p}_m}. \tag{3.17}$$

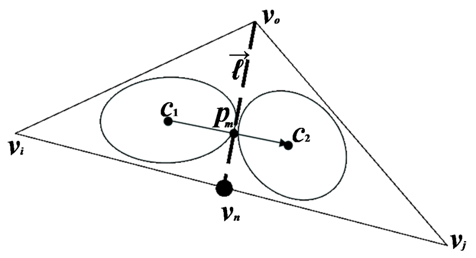


Figure 3.7: Sketch illustrating the method to obtain the direction $\vec{\ell}$ of the partitioning line of a triangle using the centers found by kmeans.

Singular Value Decomposition

The SVD is well known matrix factorization method. It has a wide use in various engineering applications. In order to apply it here, the points $\boldsymbol{p}_I$ of the triangle have to be arranged into a measurement matrix $\boldsymbol{\Upsilon}$ as

$$\boldsymbol{\Upsilon} = \left[\begin{array}{cccc} x_1 & x_2 & x_3 \dots & x_n \\ y_1 & y_2 & y_3 \dots & y_n \\ I_1 & I_2 & I_3 \dots & I_n \end{array} \right], \tag{3.18}$$

where n is the number of pixels in the triangle. The matrix $\boldsymbol{\Upsilon}$ has then to be centered, i.e. the mean is subtracted, to obtain the centered measurement matrix $\tilde{\boldsymbol{\Upsilon}}$. Then, the SVD of $\tilde{\boldsymbol{\Upsilon}}$ is computed as

$$\tilde{\boldsymbol{\Upsilon}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T. \tag{3.19}$$

As a property of the SVD, the columns of $\mathbf{U}$ present the directions of the variances of the measurement matrix $\tilde{\boldsymbol{\Upsilon}}$ in which the first column of $\mathbf{U}$, i.e. $\mathbf{u} = (u_1, u_2)$, is the largest one. It can also be shown that $\mathbf{u}$ is perpendicular to the sought partition line [114]. Hence, it can be written as

$$\vec{\ell} = \left[\begin{array}{c} u_2 \\ -u_1 \end{array} \right]. \tag{3.20}$$

The SVD is an expensive operation and it has to be repeated for every triangle using this partitioning scheme. In order to determine the direction of the subdivision in a triangle, however, only the first singular vector (column) of $\mathbf{U}$ is required. This can be understood since this vector represents the direction of the highest variance in the triangle which in the only thing needed from this factorization to compute the partition line. Therefore, the full SVD computation is not necessary anymore. Instead, the *thin*, the *compact* or the *truncated* SVD algorithms can be applied to determine the first singular vector of $\mathbf{U}$ because they are much cheaper to evaluate. An excellent review on these algorithms can be found in [114].

3.4.3 Dividing a Triangle

The tritree subdivisioning method assumes that the vertex lies in the middle of the longest edge of the triangle from which the partitioning line is then determined using Equation (3.15). In the case of the kmeans and SVD clustering methods, it is the other way around. Using the directional vector of the partitioning line $\vec{\ell}$ obtained from Equation (3.17) or Equation (3.20), it is possible to compute the new non-uniform sample as the intersection of the partitioning line and the opposite triangle's edge. Suppose that the line equation of the opposite edge is given by

$$\boldsymbol{v}_1 + \beta_o \cdot (\boldsymbol{v}_2 - \boldsymbol{v}_1), \tag{3.21}$$

and that of the partitioning line which starts at the opposite vertex $\boldsymbol{v}_3$, i.e. $\boldsymbol{v}_o = \boldsymbol{v}_3$, is

$$\boldsymbol{v}_3 + \beta_p \cdot \vec{l}. \tag{3.22}$$

By combining Equations (3.21) and (3.22), we obtain the following

$$\begin{bmatrix} (\boldsymbol{v}_2 - \boldsymbol{v}_1) & \vec{\ell} \end{bmatrix} \cdot \begin{bmatrix} \beta_o \\ \beta_p \end{bmatrix} = \boldsymbol{v}_3 - \boldsymbol{v}_1, \tag{3.23}$$

where β_o is the coefficient of the line opposite to $\boldsymbol{v}_o$ and β_p is as defined before. The coordinates of the new vertex or the non-uniform sample is then determined by plugging the result in Equation (3.21) or Equation (3.22). In the case that the scalar β_o is equal to 0 or 1, Equation (3.21) becomes ill conditioned and it will be not possible to compute the new vertex anymore. To overcome this concern, the subdivision of the triangle has to be conducted by placing the new vertex in the middle of the triangle's longest edge in a similar manner to what is done in the tritree subdivisions.

Another issue that is important to mention is the possibility that the computed new vertex possesses non-integer coordinates. In such a case, it is necessary to round the coordinates so that the new vertex lies a little bit outside the triangle T to be divided. This is justified since if the point lies inside the triangle, the resulting mesh will contain gaps.

3.4.4 Implementation Issues

Two points must be addressed in order to ameliorate the performance of the BSP based content adaptive meshing algorithm in terms of speed and let it avoid excessive computations.

1. **Minimum Size of a Triangle:** The subdivision of a triangle stops if the stopping criterion is satisfied. However, it might be possible that a

certain triangle subdivision leads to a child triangle with an area less then unity. Such triangles do not contain any inlying pixels and will not lead to any enhancement in the quality of the mesh approximation of the image. Instead, these triangles will only result in increasing both the number of non-uniform samples and that of the triangles. To avoid this issue, such subdivisions have to be stopped and deleted from the triangle tree and the father triangle should be added instead.

2. **Parallelization of the Meshing Process:** The nature of the BSP based meshing algorithm makes it suitable for the computations to be distributed among several processors working in parallel. After each subdivision in the image, the processing of each of the two children triangles is completely independent from the other one. Consequently, each of the new triangles can be treated using a different processor. To clarify the last idea, an example is illustrated in Figure 3.8 using the tritree subdivision scheme. When working with a single central processing unit (CPU), all the image is assigned to it as shown in Figure 3.8a. When using two CPUs working in parallel as in the modern multi-core processors, the image can be divided in advance into two triangles and then each one of the triangles is processed by a CPU. The same strategy can be followed with more than two CPUs. The complete adaptive mesh of the image can be finally combined from every CPU into a single mesh by simply padding all the parts together. A similar strategy can be applied when using the SVD or the kmeans clustering schemes in BSP. The image can be predivided using tritree into several triangles. Then, each triangle can be independently processed using these methods. The parallelism capability introduced by the application of BSP cannot be executed in most of the other content adaptive meshing techniques since the mesh cannot be easily split and merged as can be done in this case. This issue is also valid when comparing the proposed tritree algorithm with the original version proposed in [112] that can be depicted in Figure 3.6.

3.5 Results and Discussion

The experiments performed consist of measuring the performance of the BSP based algorithms and compare them to some of the state of the art methods described in Section 3.2. We will first test the efficiency of the algorithms by measuring the size of the mesh needed to approximate an image and the

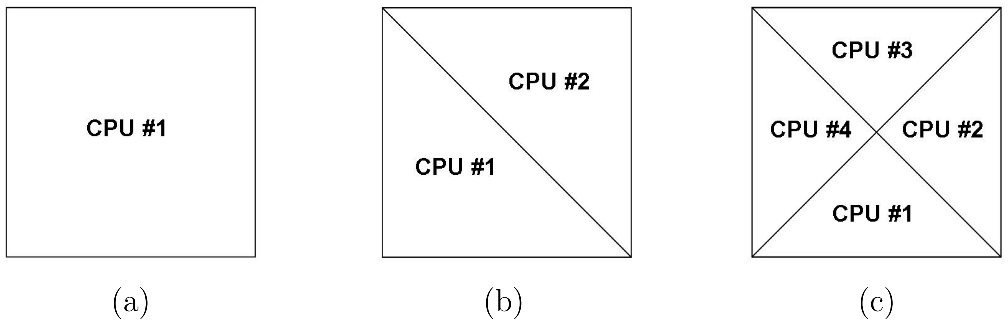


Figure 3.8: Distributing tritree among several processing units working in parallel. (a): The whole image is processed by a single CPU. (b): The image is divided into two equal triangles and then each is processed by a CPU. (c): The image is split into four parts and each is then processed by a different CPU.

compression ratio which is defined by

$$compression\,ratio = \frac{image\ size}{number\ of\ feature\ points}, \tag{3.24}$$

where the number of the feature points denotes that of the nodes of the mesh or the non-uniform samples. In the following, the BSP-Tritree will refer to the BSP with the simple partitioning scheme, the BSP-kmeans refers to the combination of BSP and kmeans clustering while the BSP-SVD refers to BSP with SVD. In all the presented results, the image is reconstructed from the mesh by computing the weighted sum of the intensities shown in Equation (3.8) with the weights taken as the barycentric coordinates of the pixel in the corresponding triangle of the mesh, see Equations (3.11) and (3.12). The maximum area of the triangle was set to 100 pixels. In addition, the methods of Ramponi and Yang are slightly modified from their original versions to allow the variation of the PSNR threshold in the image. The first mesh approximation is constructed as described by the authors and the PSNR of the image is computed. If the latter is lower than a predefined threshold, e.g. 35 dB, new pixels are added to the mesh in the locations where the error is high. A summary of the adjustment is illustrated in Table 3.1.

Figure 3.9 shows the results of the compression ratio and number of triangles obtained with the different algorithms when applied to the Lena image. The results are obtained while varying the PSNR between 25 and 40 dB. As can be seen, the proposed method in all of its versions has a better compression ratio than both of the other techniques especially at low values of the PSNR. As the PSNR increases, i.e 40 dB, the improvement of the proposed methods becomes smaller. However, by looking at the number of the

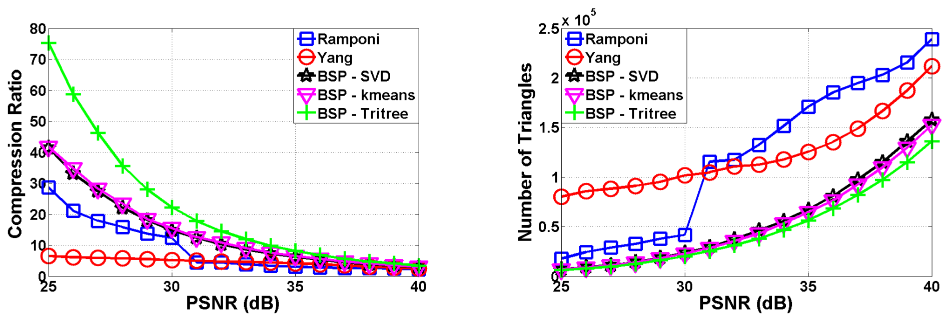
Table 3.1: Incurring the variation of the PSNR threshold in the meshing algorithms of Section 3.2.

```
For a given image and a PSNR threshold $\epsilon$, do the following:

1- Compute the non-uniform samples by applying a technique from
   Section 3.2
2- Apply Delaunay triangulation to the samples to generate the
   adaptive mesh
3- Construct the image using the mesh
4- Compute the PSNR value $\epsilon_c$ of the reconstructed image
   4a- if $\epsilon_c \geq \epsilon$, exit.
   4b- if $\epsilon_c \leq \epsilon$,
       4b1- Add some samples where the error is high
       4b2- Modify the mesh
       4b3- Goto Step 4
```

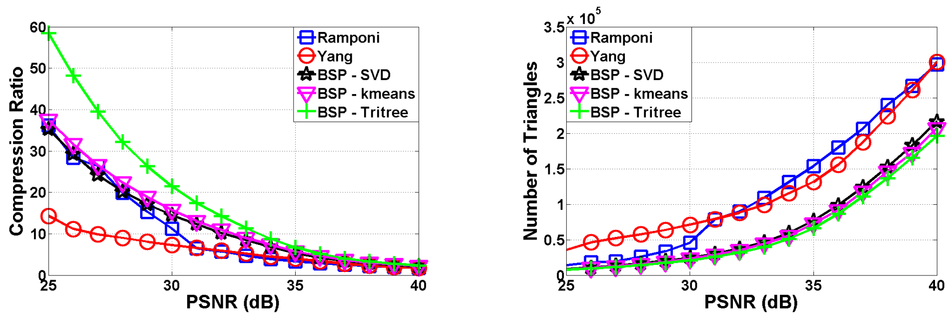
triangles in the generated meshes, the amelioration is still significant. The proposed techniques results in 33% less number of triangles on average than the other techniques even at 40 dB PSNR which corresponds to a high reconstruction quality of the image. Moreover, the proposed algorithm has a stable transition while varying the PSNR. It has no abrupt jumps as in Ramponi's algorithm since it considers the quality of the reconstruction of each pixel when constructing the mesh while Ramponi's method seeks for the non-uniform samples by simple filtering operations. Such stability is also noticed in Yang's technique since it propagates the error using a diffusion algorithm in order to choose the non-uniform samples. A similar conclusion is reached by applying the algorithms to the Peppers image shown in Figure 3.10, the Tsukuba and the Teddy images of the Middlebury data set, available at [115], in Figures 3.11 and 3.12 respectively.

Figure 3.13 shows the original image of the Lena along with its reconstruction using all of the methods and by setting the PSNR at 30 dB. By analyzing the reconstructed images, it is not difficult to notice that the BSP based methods preserve the quality of the image while presenting the image with meshes of smaller sizes than the one obtained by the other algorithms. The number of triangles used to represent the image is around $0.25 \cdot 10^5$ for all the versions of the BSP based method while it is $0.32 \cdot 10^5$ for Ramponi's method and 10^5 using Yang's method, see Figure 3.9. In addition, the proposed methods tend to have a smoother reconstruction of the images and with lower artifacts. This can be visualized in the reflection of Lena's hat



(a) (b)

Figure 3.9: Results of the meshing algorithms on the Lena image with varying PSNR. (a): Compression ratio. (b): Number of triangles in the mesh.

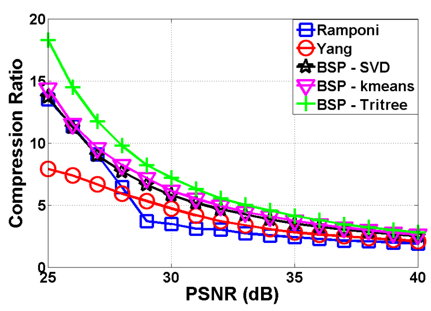


(a) (b)

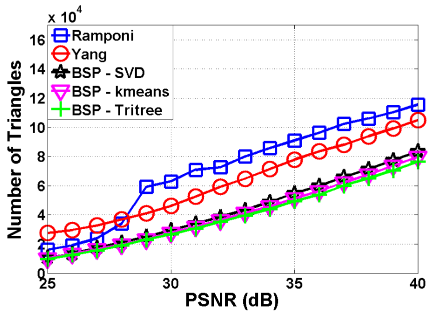
Figure 3.10: Results of the meshing algorithms on the Peppers image with varying PSNR. (a): Compression ratio. (b): Number of triangles in the mesh.

in the mirror and it's shoulder. However, the quality of the BSP-Tritree is slightly less when comparing it to that of BSP-kmeans and BSP-SVD. The main reason for this degradation is that BSP-Tritree follows a simple method to subdivide a triangle which does not take into account the variation of the pixels' intensities as BSP-kmeans and BSP-SVD do. For convenience, the content adaptive mesh representations of the Lena resulting from the all the methods in comparison are presented in Figure 3.14. By examining and comparing the visual outputs of the algorithms on the Teddy image of [115] in Figures 3.15 and 3.16, one can infer a similar judgment.

To further elaborate on the results, the outcome of the algorithms on the Lena image is presented in Figure 3.17 for 35 dB PSNR. Here, it can be directly seen that all the algorithms have resulted in a good output. The artifacts have almost vanished in all of the reconstructed images to the exception of the outcome of Yang's algorithm where the reflection of the hat in the mirror is a little bit erroneous. However, for this approximation, the proposed algorithms resulted in around $0.7 \cdot 10^5$ triangles for the image, Ramponi's

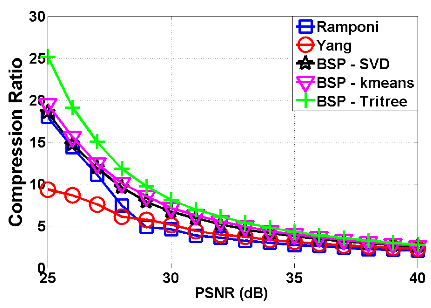


(a)

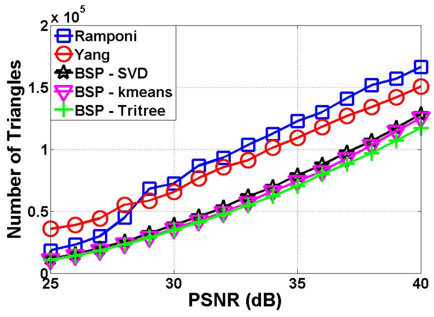


(b)

Figure 3.11: Results of the meshing algorithms on the Tsukuba image of [115] with varying PSNR. (a): Compression ratio. (b): Number of triangles in the mesh.



(a)



(b)

Figure 3.12: Results of the meshing algorithms on the Teddy image of [115] with varying PSNR. (a): Compression ratio. (b): Number of triangles in the mesh.

algorithm required $1.65 \cdot 10^5$ triangles and Yang's method obtained $1.25 \cdot 10^5$. Consequently, better image qualities are achieved with the application of the proposed algorithms with the requirement of smaller mesh approximations.

Besides the quality, the speed of the algorithm is an important factor to assess especially if it has to be applied in TPTA applications. For this purpose, some tests are conducted by applying each meshing algorithms to various sample images while varying the PSNR threshold. We used here the Tsukuba and Teddy images of the Middlebury data set along with a VGA sample image shown in Figure 4.7. The simulations were performed using an AMD Athlon XP 64 bit processor (2.2 Ghz, 2 GB RAM) with a Linux operating system and C++ programming language and their outputs are recorded in Table 3.2. The outcome suggests that the time required by the tritree partitioning scheme is slightly more than what is needed by the methods of Ramponi and Yang while the other BSP variants result in a noticeable increase in time. The reason for the increase is clearly due to the employment of the SVD and the kmeans clustering schemes.

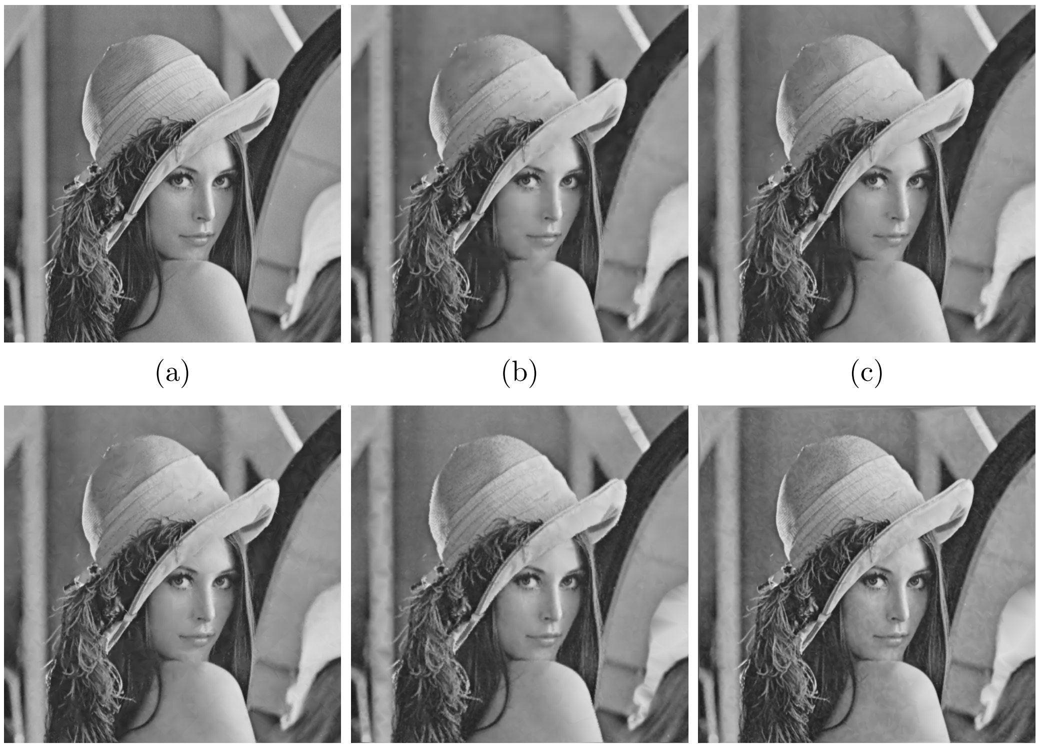


Figure 3.13: Results of the different algorithms on the Lena image at 30 dB PSNR. (a): The original 512×512 image. From (b) to (f), respectively: The reconstructed image with BSP-Tritree, BSP-kmeans, BSP-SVD, Ramponi's algorithm and Yang's algorithm.

Table 3.2: Time Evaluation in seconds of the content adaptive meshing methods at various PSNR thresholds.

	Tsukuba			Teddy			Sample		
Algorithm	30 dB	35 dB	40 dB	30 dB	35 dB	40 dB	30 dB	35 dB	40 dB
BSP-Tritree	0.19	0.24	0.3	0.28	0.36	0.47	0.51	0.61	0.73
BSP-kmeans	0.62	0.76	0.88	0.97	1.16	1.43	1.87	2.14	2.38
BSP-SVD	0.51	0.63	0.79	0.76	0.98	1.26	1.34	1.61	1.92
Yang [90]	0.12	0.16	0.2	0.16	0.24	0.32	0.27	0.33	0.49
Ramponi [101]	0.11	0.13	0.18	0.12	0.19	0.28	0.2	0.31	0.41

By comparing the three BSP based methods, BSP-kmeans and BSP-SVD lead to almost the same results in terms of visual quality and performance. Such an event might seem to be suspicious at the beginning. However, both SVD and kmeans cluster a data set by finding a hyperplane which passes through the centroid of the data which explains the similarity in their results. This was shown in several recent works as in [116,117] for example and which

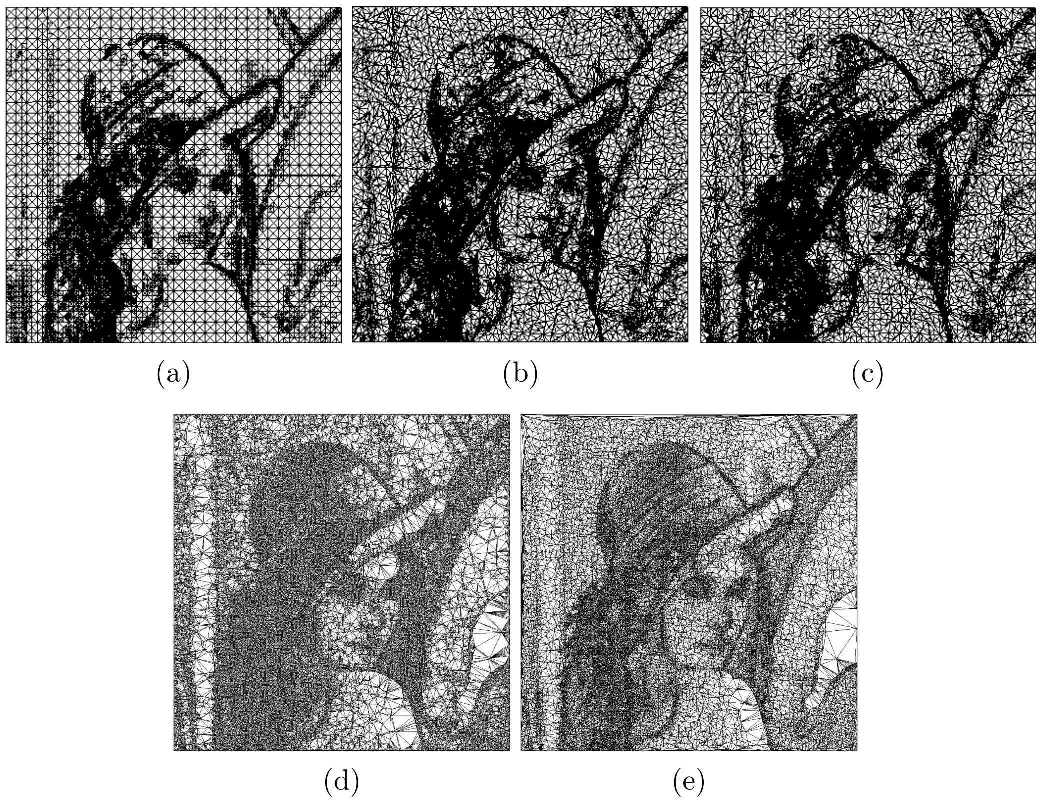


Figure 3.14: The content adaptive mesh representation obtained on the Lena image at 30 dB PSNR. From (a) to (e), respectively: The mesh with BSP-Tritree, BSP-kmeans, BSP-SVD, Ramponi's algorithm and Yang's algorithm.

confirms the validity of the obtained outcome. On one hand, the two schemes lead to an improvement in the visual quality of the reconstructed images when compared to BSP-Tritree as was previously shown. On the other hand, they are accompanied by an increasing complexity in the computations as was illustrated in Table 3.2. For this reason, it makes more sense to apply a simple partitioning scheme, i.e. BSP-Tritree, and avoid the other ones in a TPTA scenario since it is cheaper to evaluate and does not lead to a noticeable degradation in the quality of the results.

Although BSP-Tritree is slightly slower than the methods of Yang and Ramponi, it has an advantage of possessing a parallel structure due to the property incurred from BSP. By exploring this attribute as was suggested in Figure 3.8, it will be possible to benefit from the presence of multiple CPUs and subdivide the computations to each one accordingly. Figure 3.18 shows the average time required by BSP-Tritree applied to several 640×480 images at 40 dB PSNR. The measurements were performed using a cluster of four AMD processors where each is similar to the one used in the previous

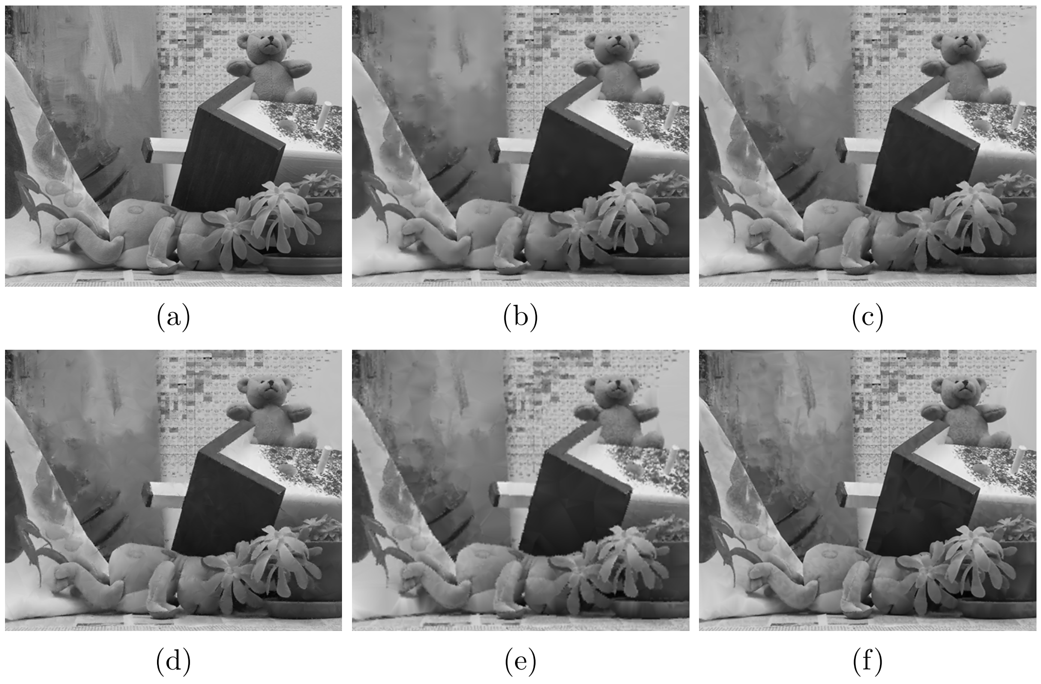


Figure 3.15: Results of the different algorithms on the Teddy image at 30 dB PSNR. (a): The original 450×375 image. From (b) to (f), respectively: The reconstructed image with BSP-Tritree, BSP-kmeans, BSP-SVD, Ramponi's algorithm and Yang's algorithm.

experiments. Each CPU was assigned an equal amount of the image to be processed and the complete mesh of the frame is then gathered in one data set by padding each part. To force such parallelism in the C++ programming language, it is necessary to use the multi-threading library, i.e. "*pthread*", along with "*vfork*" and initiate each part of the image as a single thread. Using a single CPU, the average frame rate is low as was remarked in Table 3.2 for the 640×480 image. But as the number increases to 4 CPUs, i.e. (4 parallel threads), the rate increases to 4 VGA frames per second (fps). Note that applying this scheme requires the usage a power of two number of processors to make the overall processing time decrease. Using three CPUs for example, two CPUs have to process half of the image while the third one must process the other half and thus requiring more time than the others. This obtained result quantifies the main advantage of BSP-Tritree. The meshing scheme can be distributed among several processing units while using other methods, the job is more complicated. The main reason lies in the fact that if the image is split into several CPUs, there will be a difficulty in interconnecting the partial meshes into a single one as can be done using tritree. Such a result motivates the application of BSP-Tritree in TPTA.

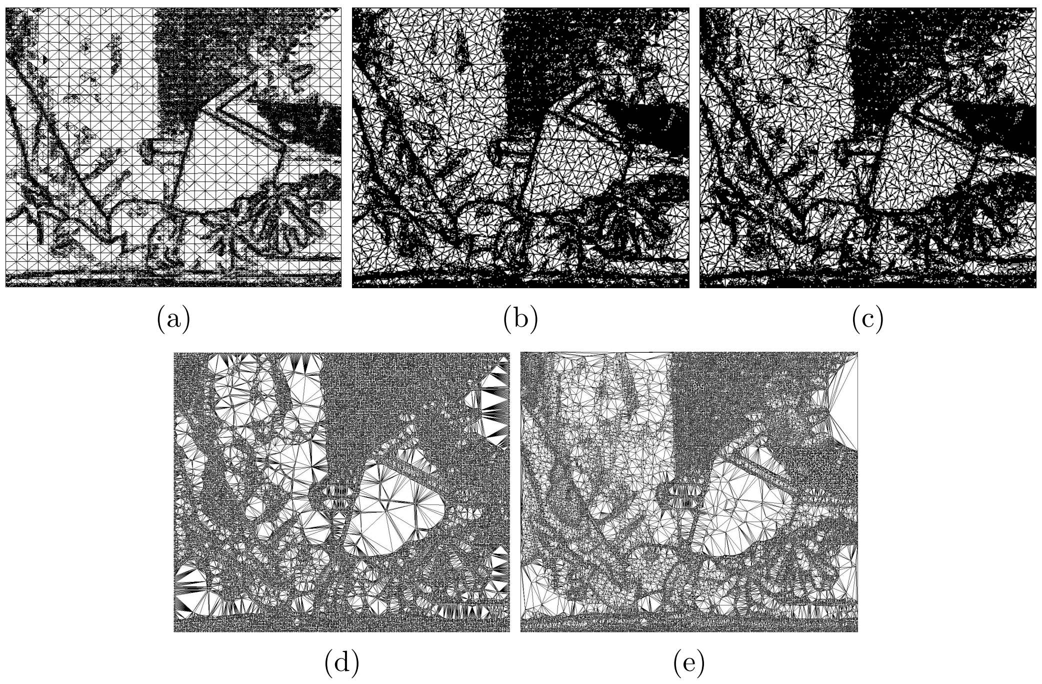


Figure 3.16: The content adaptive mesh representation obtained on the Teddy image at 30 dB PSNR. From (a) to (e), respectively: The mesh with BSP-Tritree, BSP-kmeans, BSP-SVD, Ramponi's algorithm and Yang's algorithm.

3.6 Summary

This chapter presented a new technique based on BSP to approximate an image with content adaptive mesh. The main idea was to model the intensity variation of the pixels located inside a triangle of the mesh using the parametric equation of the plane defined by its three vertices. A cost function using PSNR was then defined and maximized with the objective to locate the triangles that best describe the points lying within. As a starting point for the method, an image is divided into two triangles along one of the diagonals. In order to make BSP decide how to recursively subdivide the triangles, three variants of the algorithm were proposed depending on the clustering method used. The results showed that the size of the meshes obtained using the BSP scheme are smaller than the values given by the state of the art methods. Furthermore, the quality of the reconstructed images was preserved. In terms of speed, the BSP method using the simple clustering scheme, i.e. BSP-Tritree, was found suitable for TPTA applications since it can be easily expanded to real-time. As a consequence, this method will be employed in the next chapter to reconstruct the 3D meshes of a scene in combination with the depth information provided from stereo matching.

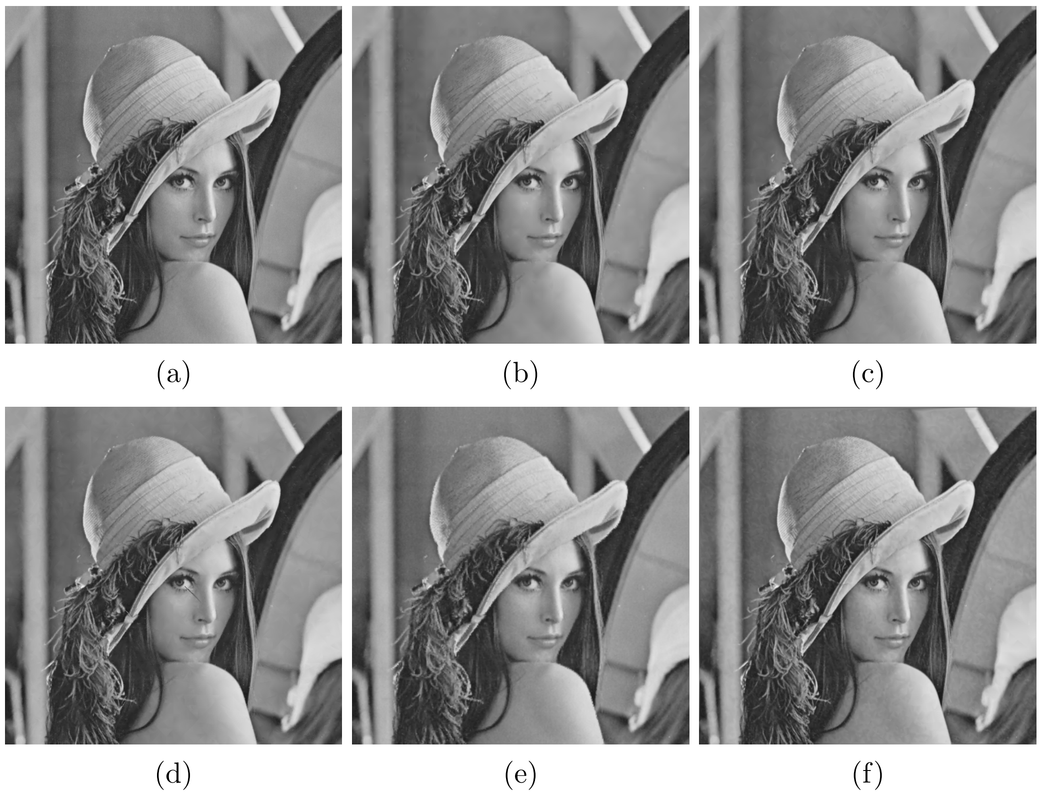


Figure 3.17: Results of the different algorithms on the Lena image at 35 dB PSNR. (a): The original 512×512 image. From (b) to (f), respectively: The reconstructed image with BSP-Tritree, BSP-kmeans, BSP-SVD, Ramponi's algorithm and Yang's algorithm.

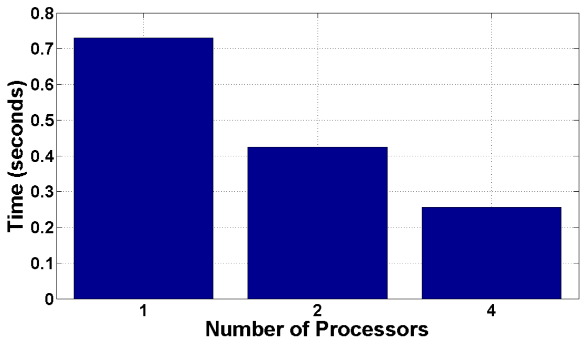


Figure 3.18: Average time needed by BSP-Tritree to generate a mesh versus the number of CPUs used in parallel. The PSNR threshold was set to 40 dB. The images used were of size 640×480.

Chapter 4

Stereo Matching

Matching the stereo images of the teleoperator is the kernel of the scene acquisition system to assess the depth. In a TPTA scenario, it is essential to compute a *confident* depth estimate of the scene by minimizing the errors incurred from stereo matching. On one side, this is necessary to ensure that the operator experiences an accurate depth perception of the view, and hence ameliorating his sense of *immersion* in the virtual 3D environment. On the other side, increasing the fidelity of stereo matching comes with an increment in the required computational resources. This is due to the wide range of possibilities that should be taken into account when formulating the problem. Such a consequence enhances the delays and deteriorates the sense of immersion. To resolve this predicament, the sake of this chapter is to exploit the redundancy in an image and benefit from the derived non-uniform image sampling strategy. A stereo matching framework will be formulated to estimate the depth values at the samples in order to restrict the domain of the computations while taking into account their structure in the image.

4.1 Geometrical Setup

Depth from stereo is one of the oldest problems in computer vision. The aim of this field is to estimate the depth by finding the correct correspondences between two images of a scene taken from different views. The main idea behind this issue can be understood from the fact that the displacement in the position of each pixel is directly related to its depth value in the real world. This phenomenon occurs since when each camera perceives the scene from a different view, a point in the 3D space will be projected to a different position in each image. This setup can be described using the epipolar geometry of a camera system depicted in Figure 4.1. A point $\boldsymbol{P}$ in the 3D space is projected

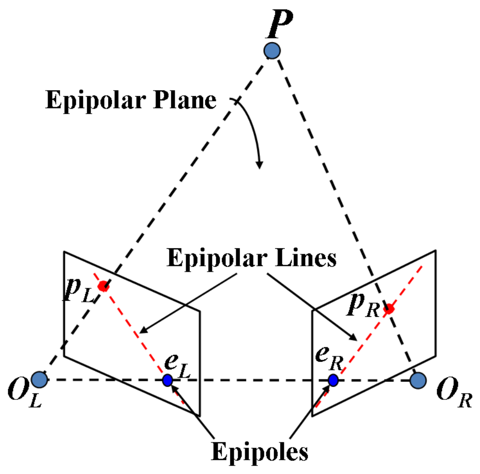


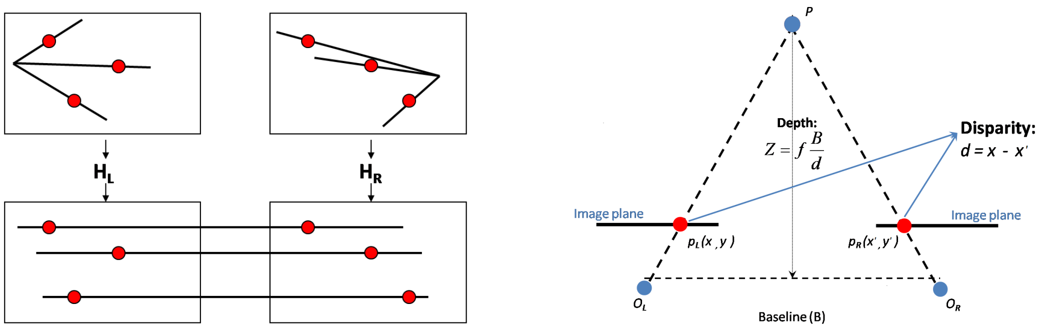
Figure 4.1: The epipolar geometry of a stereo camera system.

using the perspective projection camera model of the left and right cameras into two points $\boldsymbol{p}_L$ and $\boldsymbol{p}_R$ respectively[1]. The projection of $\boldsymbol{P}$ onto each camera passes through the corresponding camera center, see Figure 2.1. The plane formed by $\boldsymbol{P}$ and the two camera centers $\boldsymbol{O}_L$ and $\boldsymbol{O}_R$ is called the epipolar plane. The line that joins the two camera centers, i.e. $\boldsymbol{O}_L\boldsymbol{O}_R$, is called the baseline of the stereo camera. The baseline crosses each image at the epipoles, i.e. $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$, which can be also considered as the projection of the camera center of one camera in the other one. The intersection of the epipolar plane with each image is called the epipolar line. Note that all the possible epipolar lines in an image must pass trough the epipole since each epipolar plane includes the baseline by definition. From this fact, one can deduce that the candidate match of a pixel in the left image must lie on the corresponding epipolar line in the right image and vice versa[2].

The epipolar geometry is very important for stereo matching since it can be used to constrain the domain of search of every pixel in the left image to searching only along the corresponding epipolar line in the right image. To incorporate this knowledge, the stereo images are usually transformed such that the epipolar lines lie along the horizontal scanlines of the images. This process is usually referred to as stereo rectification. It consists of finding a homography for each image that maps its epipole to infinity, see Figure 4.2a for an illustration. Several algorithms can be found in the literature that perform this issue as the ones proposed in [118, 119, 120] for example. By applying any of these techniques, the problem of stereo matching can be

[1]The perspective projection model of the camera was described in Section 2.1. Please refer to it for more information.

[2]A more detailed derivation of the epipolar geometry can be found in [20]



(a) (b)

Figure 4.2: The stereo rectification process. (a): A homography is applied to each image. Each homography maps the epipole of the corresponding image to infinity so that the epipolar lines are parallel to the scanlines. (b): The geometrical setup in Figure 4.1 after rectification.

formulated as to find the corresponding pixels of the images that lie along the same scanline and, hence, reducing the complexity of the process. The search region for each pixel reduces from the product of the size of the image to that of the length of the scanline.

The geometrical setup depicted in Figure 4.1 will become after rectification as shown in Figure 4.2b. Let $\boldsymbol{p}_L(x, y)$ and $\boldsymbol{p}_R(x', y')$ be the corresponding pixels in the left and right images respectively. The disparity value is defined to be as the difference in the positions of the pixels. It can be written since the images are rectified as

$$d(x, y) = x - x'. \tag{4.1}$$

The disparity value is inversely proportional to the depth of a pixel. By looking at Figure 4.2b, it is not difficult to prove that the depth Z is

$$Z = f \cdot \frac{B}{d(x, y)}, \tag{4.2}$$

assuming that the two cameras have a baseline of length B, possess the same focal length f and have parallel optical axes. In case the focal lengths of the cameras are different or their optical axes are not parallel to each other, the depth can be recovered using ray intersection methods as described in [20].

4.2 Problem Formulation

The goal of stereo matching is to estimate the value of the disparity function $d(x, y)$ at each pixel $\boldsymbol{p}_L(x, y)$ that best describes the depth of the scene captured by the reference image[3]. The disparity function depends on the similarities of the pixels in the stereo images. It can be computed when the identical pixels or features, which represent the projections of the same 3D object of the scene in the stereo images, are found. Therefore, it is necessary to establish some costs that describe the similarities among the pixels. The simplest and fastest cost measure between the pixels is the absolute difference, equivalently the squared difference, between the intensities of the pixels [16]. This measure has been extensively used in the literature because it possesses a simple equation that allows it to be easily implemented in practice. In addition, it can be easily parallelized using MMX and SSE2 SIMD (single instruction multiple data) instruction set of the Intel and AMD processors respectively [121]. Another widely used measure in stereo matching is the cross correlation which is usually performed along a patch of an image. To make the latter more robust against lighting conditions, the zero mean normalized cross correlation is used. The cross correlation results in more robust costs when compared to the absolute difference, however, it leads to a significant increase in the computational complexity. To reduce the effort of its evaluation, it is possible to accelerate its calculation by using specific hardware as in [5, 122, 123], by subdividing its computations into several steps where each can be optimized on the assembly level instruction set of the CPU [2, 124] or by applying box filtering techniques [125, 126]. Other than these ones, several measures exist in the literature. These include some that are based on image gradients [127, 128] while others compare linearly interpolated functions of the pixels instead of the pixels themselves [129]. The advantage of the last technique is its insensitivity to image sampling.

Irrespective of the similarity measure used, the cost values of each pixel in the reference image must be computed with all the candidate pixels in the other image. These form the cost volume of the disparity image. The disparity function $d(x, y)$ of the image can be thought of as a surface in the cost volume where each pixel has a single associated disparity value from all the possible disparity values. An example that visualize this configuration is shown in Figure 4.3. The cost volume is also called the disparity space of the image [16]. It is a discrete set since a disparity reflects the amount of displacement between a pixel in the reference image and the corresponding one in the other stereo image. In order to obtain an accurate estimate of the depth value of the scene, it is necessary to compute the optimal disparity surface (function) that reflects the optimal correspondences of the pixels between the images. Suppose that there are a total of N pixels in the image

[3]The left image of the stereo pair is assumed to be the reference image throughout this chapter. Hence, the disparity function will be constructed with respect to it.

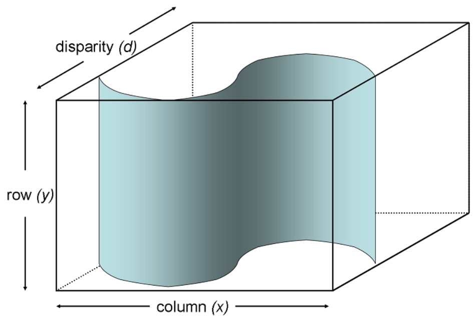


Figure 4.3: Example of the cost volume of an image which has the form of a cuboid. The disparity function of the image is a 3D surface inside the cuboid. Note that each pixel is attributed with a single disparity value on the surface.

where each has k possible disparity values. The total combinations of the possible disparity surfaces amount to N^k which is an extremely large value. To compute the optimal disparity surface, an error or energy function which we denote in this work by $E_{data}(d)$ is usually assigned to each combination of the costs. We will be designating by Δ each of these configurations. In terms of the cost values of the pixels, the energy term can be written for each combination as

$$E_{data}(\Delta) = \sum_{x,y} C(x, y, d(x, y)), \tag{4.3}$$

where $C(\cdot)$ is the cost value of the pixel which depends on the similarity measure applied. The term $E_{data}(\Delta)$ measures the overall cost of the match, or the matching quality, of each pixel with the corresponding one in the second image with respect to the chosen disparity value. The optimal disparity function in this case is nothing but the combination that leads to the least value of this cost function.

The techniques that operate only on the term $E_{data}(\Delta)$ are called the *local* methods. This naming is due from the fact that the optimization process of these algorithms ignores the effect of the neighbors of each pixel in the computations and takes only the data costs into account. To incorporate the effect of the neighboring pixels in the optimization process, three constraints must be imposed on the sought disparity surface.

Definition 4.1 *The projections of some points in 3D space on the stereo images preserve almost always the order of the points in the two images.*

This definition is called the *ordering* constraint. The reason for it is that it is practically very difficult for a projection of a rigid surface on the image planes to change the order of its points. This is why this constraint has been thoroughly exploited in practice in the selection of matches [130, 131, 16].

Since the stereo images are rectified, the ordering constraint boils down to the fact that the order of the points on the two epipolar lines (scanlines) should remain the same. This makes the constraint easy to consider in practice.

Definition 4.2 *A pixel in one image must have at most one corresponding pixel in the other image.*

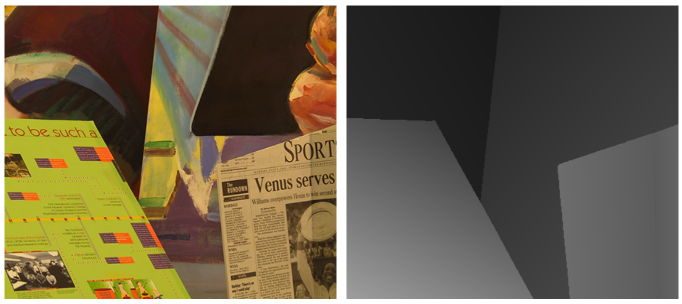
This definition is usually referred to as the *uniqueness* constraint. It states that a pixel cannot have multiple matches (disparity values) which is reasonable to assume and not difficult to reckon in practice.

Definition 4.3 *The disparity surface of an image must be piecewise-smooth.*

The last definition is known as the *smoothness* constraint. It rises from the fact that the structure of the surfaces of real-life objects varies smoothly between the edges. The abrupt jumps or discontinuities can only be found at the edges of the object; therefore, each piece (surface) is smooth and this is why it is designated by piecewise-smooth. This assumption has been dealt with thoroughly in stereo vision [132,133,16,134]. Taking this constraint into account is not as simple as with the previous ones. It is necessary here to define a term $E_{smooth}(\Delta)$ which incorporates the interactions of each pixel of the image with its neighbors. The term should be able to allow the surfaces of the disparity to vary smoothly and it should be simultaneously able to preserve the discontinuities at the edges. See Figure 4.4 of the Venus image of [16] for an example. The smoothness term should associate a pixel with its direct neighbors in such a way that ensures there is no abrupt jumps change during the minimization except at the edges. To do that, it is usually written as a function of the difference between the disparity values of the pixels

$$E_{smooth}(\Delta) = \sum_{x,y} \sum_{x_i,y_i} \iota\left(\left|d(x,y) - d(x_i,y_i)\right|\right), \tag{4.4}$$

where ι is a function that monotonically increases with the magnitude of the disparity difference and (x_i, y_i) are the coordinates of the neighbors of the pixel (x, y) under consideration. There are several ways to specify the function ι. It can be set as an everywhere smooth function that assigns penalties proportional to the difference between the disparities. This makes, however, the edges to be extended since the larger differences are highly penalized which makes the disparity jumps more difficult [132,135]. Another way is to presume smoothness as a linear model of the disparity difference and let it increase up to a certain threshold after which it becomes constant. This is usually referred to as the truncated model of the smoothness cost [134]. Other methods assume the discontinuities to be piecewise constant on the edges.



(a) (b)

Figure 4.4: (a): The Venus image of [16]. (b): Its ground truth disparity map.

This is achieved via the Potts energy model proposed in [136] which penalizes only the disparity differences above a certain threshold with a constant value. The last two functions are widely used in stereo matching algorithms since they have good abilities in preserving the edges.

By combining Equations (4.3) and (4.4), the error function that has to be minimized to obtain the optimal disparity function of the image is

$$E(\Delta) = \sum_{x,y} C(x, y, d(x, y)) + \lambda \cdot \sum_{x,y} \sum_{x_i,y_i} \iota\left(\left|d(x, y) - d(x_i, y_i)\right|\right) \tag{4.5}$$

The scalar λ is introduced to weigh the relative importance of the smoothness term with respect to the data term. It is also possible to add an extra element to the energy function that recognizes if a pixel is occluded or not in the other image. To seek for the optimal method to model pixel occlusions in stereo vision is a research area by its own and it will not be covered in this dissertation.

4.3 Related Work

Stereo matching is one of the most investigated research topics in computer vision. One can find a huge amount of literature describing various methods that try to find the optimal disparity surface of a scene captured by stereo images. An interesting reference is the Middlebury stereo vision website [115] where a benchmark was developed to compare the performance of matching algorithms. A detailed description of the benchmark is described in [16]. Among the earliest algorithms in stereo vision are the cooperative techniques which try to imitate the human vision system in their computations as in [132, 137]. The methods perform the computations locally on each pixel by applying non-linear minimization schemes. Apart from that, there exists three different groups of techniques to compute a disparity map. First, there are local algorithms which are based on the block matching technique. These can usually incorporate all the constraints defined in the previous section to the exception of the smoothness constraint. The resulting disparity value of each pixel is optimal with respect to the data costs only which explains why they are sometimes referred to as the *greedy* methods. The second group are the *scanline* based techniques which minimize the error function defined in Equation (4.5) over each scanline. The key idea here is to partition the error function into several ones which leads to a decrease in the complexity of the problem. Therefore, the obtained solution is global for each scanline which might lead to some artifacts between the different ones. Finally, there are the *global* methods which operate on the entire image to minimize the energy function. The illustrative example depicted in Figure 4.5 shows the resulting disparity maps form the application of one algorithm from each group on the Tsukuba image of [16].

The local approaches are quite fast and efficient to implement. They can be accelerated using graphical processing units (GPUs) [138, 139], by optimization on the assembly level instructions of a CPU [124,2] or by employing DSP hardware [122, 5]. However, their outcome suffer from inconsistencies, as depicted in Figure 4.5, and they usually result in low-quality depth maps.

The scanline based methods represented by dynamic programming (DP) are also fast and produce depth maps with better quality when compared to the local methods. Moreover, there has been a lot of research lately to accelerate these techniques while preserving the quality of the depth maps. In [129], low complexity cost functions were defined and used in addition to the fact that DP was modified to prune the bad search regions. In [125], rectangular subregioning was employed to partition an image into several sub-images and then each one was processed using a two-stage DP technique to obtain an efficient maximum 3D surface in the 3D volume defined by the costs. In [126], a similar principal to the latter was implemented using quadtree subregioning then followed by iterative DP which sweeps between horizontal and vertical optimizations. In [140], a DP scheme on a tree (treeDP) was implemented which minimizes the inconsistencies among the scanlines. In [141], the previous method was ameliorated by segmenting each line into several segments where each is processed using tree DP and then refined. In [142], like the previous method, tree DP was derived and used on segmented image regions instead of line segments which lead to results of better quality. In [143], a four stage DP matching strategy was suggested which handles the occlusions in an efficient manner. Last but not least, the GPU was employed to parallelize the computations in [144, 145, 146, 147].

The global techniques as the ones described in [148, 136, 134, 149] are usually too slow to use in real-time applications unless they are formulated using graphical hardware [150]. These methods, however, result in depth maps which exhibit the best quality. In the rest of this section, the dynamic programming approach will be briefly described for it represents the core of the method that will be developed in this chapter.

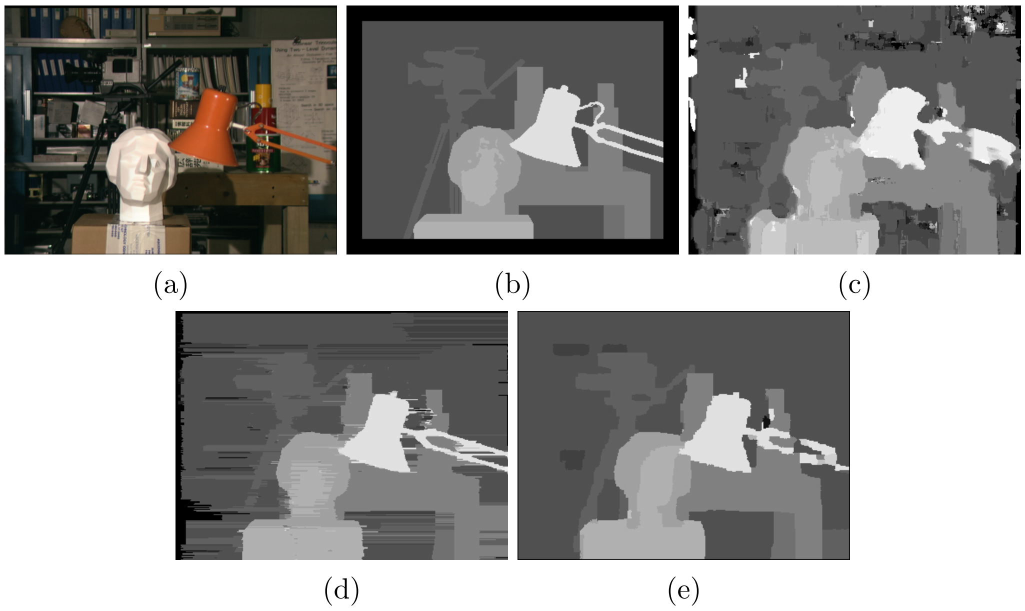
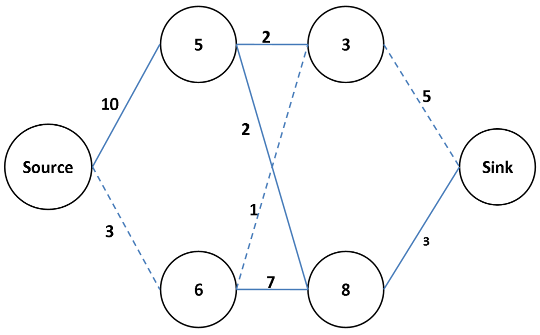


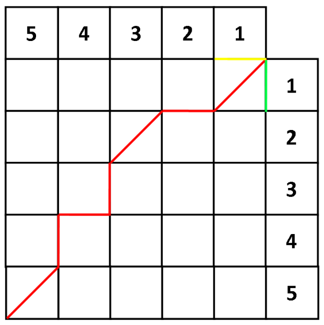
Figure 4.5: The application of the different classes of stereo matching algorithms on the Tsukuba image of [16]. (a): The original image (b): The ground truth disparity map. (c): The result of zero mean normalized cross correlation which is a local approach. (d): The scanline method of [16] based on dynamic programming. (e): The global method of [134] based on belief propagation.

4.3.1 Dynamic Programming

The dynamic programming is an optimization technique aimed to solve large and sequential problems which can be subdivided into overlapping subproblems. DP finds the solution of each subproblem and then use their combination to obtain the optimal solution of the problem. An easy example that illustrates this idea is the common shortest path problem in graph theory. The route with the cheapest cost between a source and a sink has to be computed. The path can be subdivided into nodes (vertices) where each one is attributed with a penalty. In addition, a cost is assigned to the transition between each consecutive two nodes. The setup is shown in Figure 4.6a. The optimal routing path is the combination of vertices that leads to the cheapest cost between the source and the sink. It is highlighted in the figure with a dashed line and amounts to a total cost of 18. Such a methodology has been widely used in engineering problems as in the Viterbi algorithm used to decode the data in the cellular systems [151]. A similar approach can be applied in stereo images to find the optimal disparity path. The nodes of the graph are in this case the corresponding candidate matches. The cost matrix, which designates the graph, is constructed for each scanline in the reference image with the corresponding one in the other image. The entries of this matrix form the costs of the nodes of the graph. An example is shown in Figure 4.6b for a scanline of length 5 pixels. To move from node (1,1) for example, it is necessary to choose the destination with the minimal cost between the three nodes (2,2), (2,1) and (1,2) marked in different colors. This process is repeated until the shortest path is determined. DP was first introduced in the domain of stereo vision by Baker and Binford in 1981 [130]. Its employment has then rapidly spread among researchers, as we showed earlier in this section, because it is one of the fastest optimization methods and leads to good quality disparity maps.



(a)



(b)

Figure 4.6: (a): The shortest path is the cheapest path between the source and the sink. It is highlighted in a dashed line. (b): To find the shortest disparity path using DP in stereo images, a matrix of costs is computed between the corresponding scanlines. The numbers designate the pixel positions in the scanline. Each node, e.g. (1,1), reflects a candidate matching pair. The shortest path corresponding to the optimal disparity map of the scanline is the one that minimizes the matching costs over all the scanline. It is marked here with a red color.

4.4 Sparse Stereo Matching

One aim of this dissertation is to accelerate the overall speed of stereo based 3D scene reconstruction starting from the fact that an image can be faithfully represented by a subset of its pixels, the non-uniform samples, to interpret the scene it represents. Non-uniform sampling consists of sampling a captured

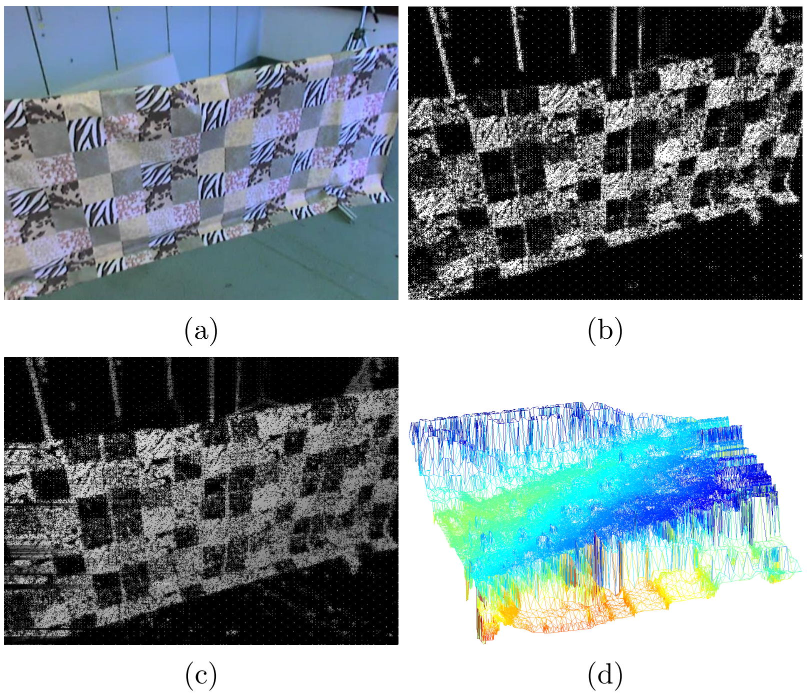


Figure 4.7: Sample result using the proposed stereo-based 3D reconstruction scheme. (a): The sample image. (b): The non-uniform samples of the sample image obtained via the BSP-Tritree algorithm of Chapter 3. (c): The sparse depth map obtained using the sparse DP that will be described in this section. (d): Top view of the 2.5D mesh obtained by integrating the output of BSP-Tritree with sparse DP.

digital image from the teleoperator cameras at an irregular grid. The image is sampled with a high rate where the intensity variation is high, otherwise it is low. The theory of non-uniform sampling of images and its relation to image content adaptive mesh was presented in Chapter 3. We also developed there the BSP-Tritree technique that can approximate an image with a mesh where its nodes are the sought non-uniform samples. Our task in this chapter is to benefit from the sparse structure of the samples and use this information when estimating the disparity map of a scene. A sparse stereo matching scheme will be developed in order to estimate the depth values only at the non-uniform samples of an image. The information obtained can be then integrated with the 2D mesh from BSP-Tritree to construct the 2.5D mesh approximation of the scene. An example is illustrated in Figure 4.7.

Sparsity of the data is a property that has been researched extensively in various fields including computer vision. By exploring the sparsity of the data, it is possible to design fast algorithms because only a small system of equations has to be solved. Since the obtained left image is sparse, it makes sense taking advantage of this fact. Sparse stereo is usually performed using a local approach. The main disadvantage of such methods, however, is the fact that the smoothness constraint stated in Definition 4.3 cannot be enforced within the computations. In addition, these methods have a lot of tendencies for errors at the object boundaries which was previously depicted in Figure 4.5. Another possibility is to apply semi-dense stereo matching techniques as in [152, 153] which determine the depth at some pixels of the image considered to be reliable. Applying the latter type of techniques do not meet the goal of this chapter since the found pixels do not have to coincide with the locations of the non-uniform samples. And for each non computed sample, the triangles of the mesh where it belongs will be missing from the reconstructed scene and thus deteriorating its quality. The idea is illustrated in Figure 4.8 by applying the stratified matching algorithm of [153] to the Tsukuba image. What is also shown is the content adaptive mesh computed with BSP-Tritree at 40 dB PSNR threshold along with the sparse disparity map by choosing the values of the semi-dense map at the nodes of the mesh. The resulting sparse depth map can be seen to contain numerous missing disparity values at the non-uniform samples. This will lead to a mesh with missing triangles that deteriorates the quality of the constructed 3D scene.

Apart from that, we choose DP to accomplish this task since it is one of the fastest optimization methods and leads to a good quality. DP was first introduced in stereo vision in the context of edge based methods [130, 131]. Then, the interest in it grew since it is one of the fastest optimization techniques and leads to good quality depth maps. DP is derived in this section to find the depth of the sparse non-uniform samples and, hence, the naming Sparse DP. The inputs are a sparse left image and a full right image, which are rectified such that the epipolar lines are aligned with the corresponding scanlines as was stated in Section 4.1. Let $\boldsymbol{p}_L(x, y)$ be a pixel in the left image lying on the y^{th} scanline and $\boldsymbol{p}_R(x', y')$ be the corresponding pixel in the right image. Since the images are rectified, $y = y'$ and the disparity function $d(x, y)$ at the pixel $\boldsymbol{p}_L$ is expressed as $d(x, y) = x - x'$.

4.4.1 Computation of the Data Costs

In Section 4.3, we discussed various pixel-based similarity measures that exist in the literature. The similarity measure C of the pixels that we will be using here is the absolute difference because it is simple to implement. The absolute difference is defined at a disparity value d of a pixel $\boldsymbol{p}_L$ of the left image to be

$$C(x, y, d(x, y)) = |\boldsymbol{p}_L(x, y) - \boldsymbol{p}_R(x - d(x, y), y)|. \tag{4.6}$$

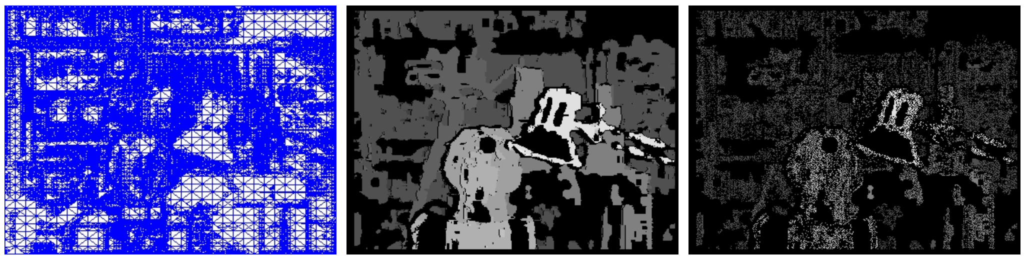


Figure 4.8: (a): The content adaptive mesh of the Tsukuba image shown in Figure 4.5a taken at 40 dB PSNR using BSP-Tritree. (b): The semi-dense disparity map resulting from the algorithm of [153]. (c): The sparse disparity map resulting by taking the disparity values of the previous map at the non-uniform samples.

In this case, the disparity value of $\boldsymbol{p}_L$ corresponds to the location where $C(x, y, d(x, y))$ reaches a minimum.

In general, any pixel-based cost function could be used instead of the absolute difference. The important issue in sparse DP is to evaluate only the costs over the support region of each non-uniform sample. The support region of a pixel $\boldsymbol{p}_L(x, y)$ is defined to be the region along which the cost is aggregated. Suppose that the support region SR of $\boldsymbol{p}_L$ is a box filter of size $w \times h$. If $\boldsymbol{p}_L(x, y)$ is a non-uniform sample, C is evaluated at all the pixels of S. This is justified since the aggregated cost of $\boldsymbol{p}_L$ must be computed over all of S as will be shown in Equation (4.7). Consequently, Equation (4.6) must be performed in sparse DP across all the non-uniform samples in the image along with the pixels belonging to their support region.

4.4.2 Aggregation of the Data Costs

The disparity surface of the image is smooth in general except at the edges of the objects. Taking the similarity measures of the non-uniform samples illustrated in Equation (4.6) alone as the cost volume, over which sparse DP optimization takes place, might not consider the interaction of the neighboring pixels properly. To resolve this issue, the sought disparity surface is assumed to be locally smooth since the neighboring pixels of $\boldsymbol{p}_L$ are likely to have the same disparity value. The initial costs are thus aggregated over a neighborhood region SR which is also called the support region of $\boldsymbol{p}_L$. The aggregated cost C_a of $\boldsymbol{p}_L$ defined over its support region SR is

$$C_a(x, y, d(x, y)) = \sum_{(x_i, y_i) \in SR} \vartheta_{SR}(x_i, y_i, d(x, y)) \cdot C(x_i, y_i, d(x, y)), \qquad (4.7)$$

where ϑ_{SR} is a weighting function defined over SR. The aggregated cost is the one that is associated to the data energy term defined in Equation (4.3) that has to be minimized. Therefore, this operation has to be repeated over all the non-uniform samples and the total costs form the cost volume of the sought sparse disparity map of the image. Aggregating the data costs is a very important step in a DP based stereo correspondence algorithm. It helps in obtaining a smooth and more consistent disparity map and in reducing the streaking effect of DP. The latter occurs due to the inconsistencies that result from minimizing the error function in Equation (4.5) separately on each scanline, see the disparity map in Figure 4.5d for example. The most common aggregation form is to apply a box filter which is nothing but an averaged summation over a fixed rectangular window around each pixel. Increasing the size of the window minimizes this effect, however, it has the disadvantage of over-smoothing the disparity map. Several windowing approaches were conducted in the literature with the goal to find the optimal region of support for each pixel that increases the reliability of the matching costs as in [154, 155, 156]. The drawback that incurs with the application of such approaches is the increasing computational time since they are complex to evaluate and there is a wide range of possibilities that has to be evaluated in order to compute the disparity map between two stereo images. This reason is what mainly lead some of the latest techniques in DP-based stereo matching to be implemented on GPUs [144, 145, 146, 147]. It is important to mention that there are other approaches that do not concentrate on the aggregation step but increase the quality of DP by optimizing on a tree [140, 141, 142], by enforcing inter-scanline consistency [131] or by performing consecutive horizontal and vertical sweeps across the image [125, 126].

One of the main advantages of proposed sparse DP scheme is that the aggregated costs have to be *only* evaluated at the non-uniform samples; hence, the process will be sped up. By looking at Figure 4.7c for example, the number of non-uniform samples needed to obtain the 3D mesh of the scene amounts to 42% of the total number of pixels. Hence, the effort of the evaluation of the aggregated costs reduces to 42% of the total effort as well.

4.4.3 Disparity Optimization Using Sparse DP

In DP, the stereo correspondence problem is formulated as finding the optimal path through the disparities that minimizes an energy function defined over a scanline. The energy function is usually minimized by conducting a search in the cost volume defined by C_a and it is written as

$$E(\Delta) = E_{data}(\Delta) + \lambda \cdot E_{smooth}(\Delta), \tag{4.8}$$

where $E_{data}(\Delta)$ is the energy term associated with the aggregated costs reflected by Equation (4.7), $E_{smooth}(\Delta)$ is the smoothness term and λ is a constant to weigh the relative importance of depth discontinuities with respect to the data costs. The smoothness term is written as a function of the difference between a pixel and its direct neighbors as was shown in Equation (4.4). In sparse DP, Equation (4.8) is minimized along a sparse scanline. The data term does not change but has to be evaluated on the non-uniform samples as was previously explained. The smoothness term, however, should be changed in order to account for the sparsity of the data. The neighbor of the pixel under consideration is in this case the closest non-uniform sample in the scanline. The smoothness term should be therefore written as

$$\widetilde{E}_{smooth}(\Delta) = \sum_{x} \iota\left(\left|d(x) - d(x_{nb})\right|\right), \tag{4.9}$$

where x_{nb} represents the neighboring non-uniform sample and the index y was dropped from the summation since the minimization is conducted independently along each scanline. In a standard DP algorithm, the function ι is usually chosen to be monotically increasing with the difference between the disparities of the neighbors. This is also valid in our case but the distance between the non-uniform samples should be *additionally* taken into account. This can be visualized because the closer the non-uniform samples are, the more probable that there is high intensity variation in the corresponding region of the image. While when the samples are distant from each other, the intensity variation is low. Consequently, the smoothness term should consider penalizing the closer samples less than the distant ones since the discontinuities are more probable to occur due to the high variation of the intensity. In other words, the function ι should be high if the neighboring samples are close and it should be monotonically decreasing as the distance between the samples increases. In this work, the function ι we chose is

$$\iota = \frac{1}{(x - x_{nb})^2}, \tag{4.10}$$

since it insures that Equation (4.9) is continuous and it allows for interpolation between the neighboring samples. The effect of this function can be best understood as to encourage the disparity discontinuities to coincide with the color or the intensity edges of the image. This can be seen from the fact that the distance between the samples is inversely proportional to the magnitude of the image gradients. Taking this effect into account in the smoothness term of the energy function has resulted in a good performance in global optimization approaches [136, 16].

Using the derived equations, the DP optimization is performed at each scanline y to extract the best disparity path. As in DP, the optimization process of sparse DP is composed of two steps. The first one is of a forward accumulation stage which builds up a cost volume of the scanline. A matrix $\widetilde{\mathbf{C}}$ is initiated to null. It is important to mention here that when filling this matrix, the function ι should reflect the distance between the samples only when penalizing the costs related with the neighboring sample, e.g. $\widetilde{C}\left(x_{nb}, d\left(x\right) - 1\right)$. To penalize the cost of a pixel going to a different disparity level, e.g. $\widetilde{C}\left(x, d\left(x\right) + 1\right)$, the distance between the samples should be ignored since this cost is not related to the previous sample. In such a case, the function ι must be set to unity. Thus, the matrix $\widetilde{\mathbf{C}}$ has to be filled from left to right to take the ordering constraint into account as

$$\begin{aligned}\widetilde{C}\left(x, d\left(x\right)\right) = C_a\left(x, d\left(x\right)\right) + \min\big(&C_a\left(x_{nb}, d\left(x\right) - 1\right) + \lambda \cdot \iota,\\ &C_a\left(x_{nb}, d\left(x\right)\right), C_a\left(x, d\left(x\right) + 1\right)\big).\end{aligned} \tag{4.11}$$

After that, a backward-tracing step follows to extract the optimal disparity path of the sparse scanline from $\widetilde{\mathbf{C}}$. This is repeated for all the scanlines and the obtained result is a disparity map as the one shown in Figure 4.7c. By combining the disparity map obtained from sparse DP and the mesh of BSP-Tritree, the 2.5D mesh of the scene can be constructed as shown in Figure 4.7d. The proposed algorithm is illustrated in Table 4.1.

4.5 Results and Discussion

The essence of our analysis is to assess the performance of the sparse stereo matching algorithm in terms of quality and speed. The quality will be measured using the Middlebury data sets of [16, 157] since they are provided with their ground truth depth maps. Two DP-based stereo algorithms will be tested in our results. The first one is the DP implementation presented in [16] which consists of simple box filtering technique as the aggregation cost. The second algorithm is the DP implementation of [147] where the aggregation is performed using the high quality adaptive weight approach described in [156]. The latter consists of aggregating the data cost function based on both color and geometric proximity of a pixel in the image. The advantage of such aggregation scheme is that it avoids the problems of image segmentation by using a continuous weighting function. We will be applying to each of these methods the proposed sparse DP approach. The non-uniform samples are computed in all the cases with the BSP-Tritree scheme derived in Chapter 3. We will also test in our experiments, the (local) 5×5 normalized correlation approach. The sparse depth maps resulting from all the

Table 4.1: The Sparse DP stereo matching algorithm.

```
Given a pair of stereo images and the non-uniform samples of the reference
image. Let $SP$ be the shortest path and $ID$ be a matrix of pointers.

For every scanline, do the following:

Cost Accumulation:
1- Initiate the aggregated costs $\mathbf{C}_a$ at the non-uniform samples
2- For each non-uniform sample $x = 1 : i$ and disparity level $d = 1 : n$
    2a- Construct the entries $\widetilde{C}(x, d)$ with Equation (4.11)
    2b- Store the index of the vertex corresponding to the "min"
        operation of Equation (4.11) in $ID(x, d)$

Backward-Tracing:
1- Find the vertex associated with the minimum cost $\widetilde{C}(x, d)$
2- Set the last point of the shortest path $SP(i)$ to the vertex
3- For the remaining non-uniform samples $x = i - 1 : 1$
    3a- Find the pointer $ID(x, d)$ associated with $SP(x + 1)$
    3b- Set $SP(x)$ to $ID(x, d)$
```

algorithm are shown in Figure 4.9. The non-uniform samples were computed at 40 dB PSNR threshold. The error rate of the depth maps shown in the figure are illustrated in Table 4.2. They were obtained by taking the ground truth depth maps found in [16, 157], extracting the disparity values of the non-uniform samples and then computing the percentage of the error between the resulting sparse disparity maps and the ground truth values. The results show that the error rate of the DP based algorithms has increased less than 4% on average with respect to the total number of pixels when compared to the original DP algorithms. Compared to the correlation based stereo, however, it is easy to notice that the quality of the DP-based sparse stereo methods are a lot better despite the resulting decrease in the quality. To get a feeling on what does this increase in the error rate means in the quality of the depth maps produced by sparse DP, we show in Figure 4.11 the depth maps produced by the original algorithm of [147] along with the reconstructed dense disparity maps of the two sparse DP versions. The dense disparity maps of the sparse DP techniques were obtained by interpolating the sparse disparity maps shown in the first and second rows of Figure 4.9 using Equation (3.11) with the weights taken as the barycentric coordinates of the pixel in the corresponding triangle of the mesh. As we can see, the

Table 4.2: Percentage error of different stereo matching algorithms. The non-uniform samples were computed at 40 dB PSNR.

Image	DP of [16]		DP of [147]		Normalized
	Sparse	Original	Sparse	Original	Correlation
Tsukuba	11.27	7.79	10.46	7.13	14.41
Venus	7.34	6.17	6.76	6.01	12.83
Teddy	21.25	17.33	21.47	15.46	33.17
Cones	20.63	15.96	20.2	14.67	30.38
Average	15.12	11.82	14.72	10.8	22.81

visual quality did not deteriorate significantly. This is also confirmed in Table 4.3 obtained from the Middlebury evaluation web-tool for 0.5 pixels error threshold. On the contrary, the table shows us that the sparse version of [147] is better than other existing dense DP based algorithms. Nevertheless, the performance of sparse DP deteriorates near depth discontinuities. This fact occurs since the interpolation of the sparse disparity map takes into account the intensity edges of the image and does not consider neither the occluded pixels nor the geometrical edges which leads to the higher values of *nonocc* and *disc* in Table 4.3 respectively. This is a side effect that we have to live with for the overall time of the matching process will significantly decrease as will be later seen in this section while obtaining disparity maps with higher quality when compared to the correlation approach that was applied in [2] to telepresence. The latter will naturally increase the confidence of 3D reconstruction in a TPTA scenario. What can also be noticed is that the streaking effect by using sparse DP is minimized. This can be visualized by looking at the interpolated disparity map of the Tsukuba image in the third column of Figure 4.11 and the corresponding one in Figure 4.5d. The reason for that is the application of the adaptive depth discontinuity function in Equation (4.9) which takes into consideration the distance between the sparse samples.

Before going to the speed analysis of our sparse stereo matching technique, there is still an issue that has to be clarified. What is the necessary PSNR threshold at which the content adaptive meshing technique proposed in Chapter 3 must be set so that the complete depth map of the scene can be recovered with an acceptable error as in Figure 4.11? Why did we choose 40 dB in the previous results as a threshold? The answer to this question is simple. We varied the PSNR threshold of BSP-Tritree between 30 and 65 dB. At each value, we computed the non-uniform samples of each Middlebury test image. The depth of those samples were then computed using the proposed sparse DP and then the complete depth maps were interpolated as was previously stated. Finally, the percentage error rate of each recon-

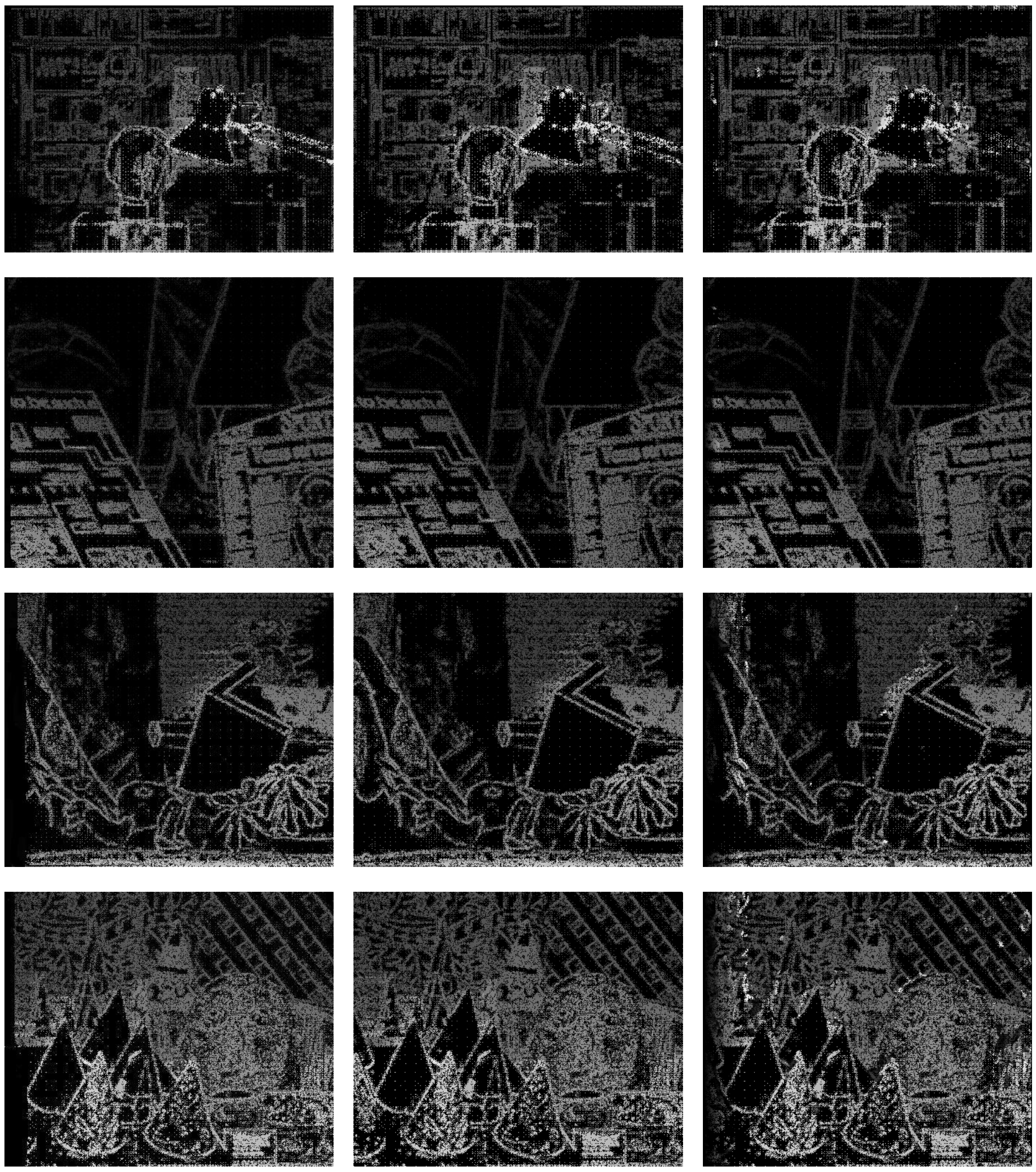
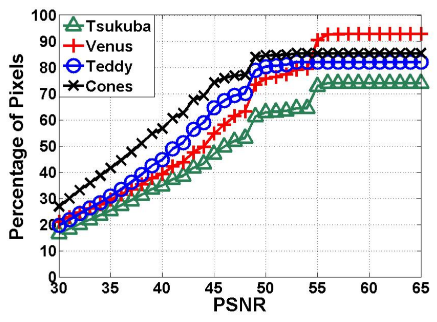
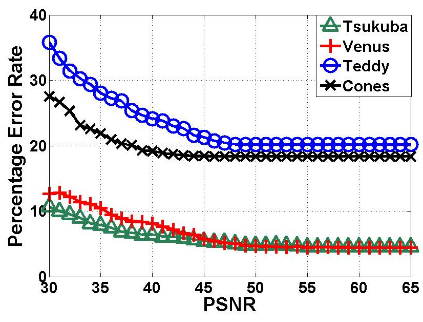


Figure 4.9: Visual outcome of the proposed sparse DP algorithm applied to several DP based stereo matching methods. The non-uniform samples were obtained at 40 dB PSNR threshold with BSP-Tritree. The first column is the sparse version of [16]. The second column shows the sparse version of [147]. The third column reflects the normalized correlation.

structed depth map was computed with respect to the ground truth depth map. The results of the test are depicted in Figure 4.10. We can also see in the figure, the percentage of the number of pixels needed at each PSNR level. By looking at the two plots, we can easily notice that the change in the error rate becomes minimal above a threshold of 40 dB. In addition, we



(a)



(b)

Figure 4.10: Dependency of the quality of the depth map on the number of pixels used. (a): Variation of the number of pixels versus the PSNR. (b): Variation of the percentage error rate versus the PSNR.

can also notice that the optimal threshold for each image lies a little bit above the PSNR value that corresponds to 50% of the total number of the pixels in the image. This means that in order to obtain a reconstruction with an acceptable error, it is only necessary to have around half of the pixels in each image. This constraint can also be given as a stopping criterion instead of PSNR for BSP-Tritree. However, the threshold was always set in our computations to 40 dB because it results in acceptable depth maps qualities. This can be better understood by depicting how does the variation in the non-uniform samples affects the visual quality of a reconstructed disparity map. Figure 4.12 shows an example of varying the PSNR threshold of the BSP-Tritree meshing scheme on the interpolated disparity map of the Tsukuba image. We use in this test the sparse DP version of [147]. What can be seen are the disparity maps that corresponds to setting the BSP-Tritree meshing scheme to 30, 35 and 45 dB respectively while the 40 dB threshold is illustrated in the second column of Figure 4.11. It is not difficult to realize here that above a 40 dB threshold, the visual quality of the depth map does not change in a noticeable manner.

The superiority of the depth map quality of DP is what makes them more attractive than correlation based techniques. However, they are usually avoided in telepresence scenarios as was stated in [2] since they tend to increase the overall latency in the system. This can also be noticed here in the comparison between the different algorithms in Table 4.4. The simulations were performed using an AMD Athlon XP 64 bit processor (2.2 Ghz, 2 GB RAM) with a Linux operating system and C++ programming language. Fortunately, the time needed by the sparse DP strategy is significantly less than that of DP for it only matches the non-uniform samples. As the size of the image increases, the improvement in time also increases since

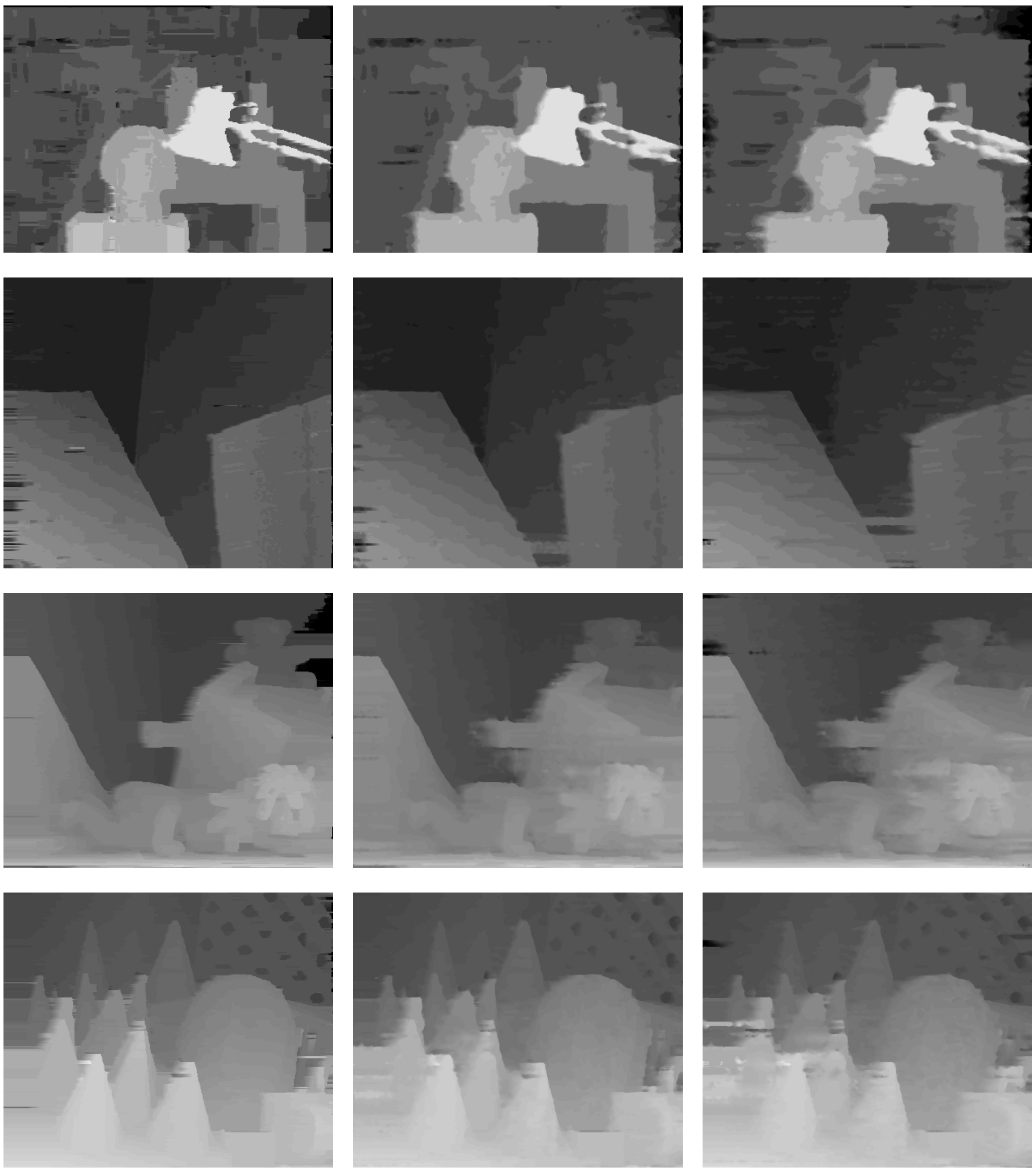


Figure 4.11: The resulting dense depth maps. First column: DP of [147]. Second column: The interpolated depth maps of its proposed sparse version. Third column: The interpolated depth maps of the proposed sparse version of [16].

the sampling eliminates the non-significant pixels in the image and hence the unnecessary disparity computations are avoided. At an image resolution of 640×480 pixels and at 150 disparity levels, the time necessary to compute the disparity map decreases on average to 50% of the original time without significantly deteriorating the quality of the depth maps, see Table 4.2.

To enhance the speed of sparse DP, we parallelized the computations by

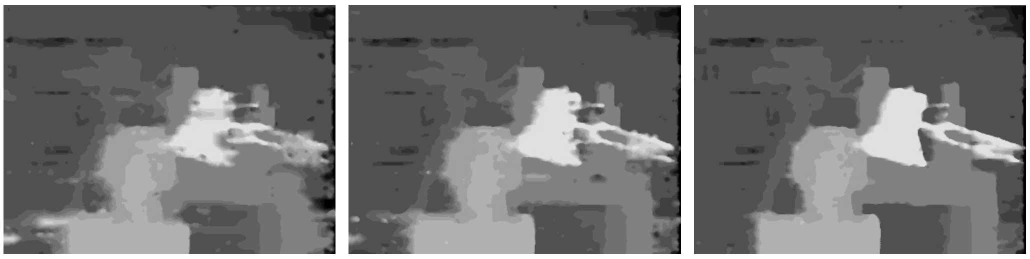
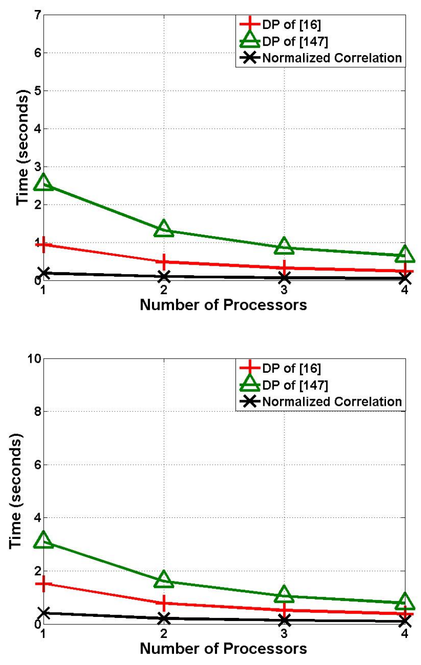


Figure 4.12: Effect of varying the PSNR threshold on the visual quality of the interpolated disparity map of the Tsukuba image of [16] with BSP-Tritree. The sparse DP version of [147] was used to compute the disparity maps. (a): PSNR = 30 dB. (b): PSNR = 35 dB. (c): PSNR = 45 dB.

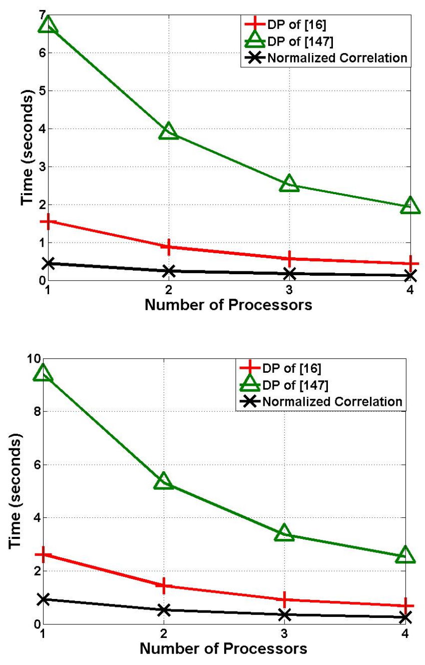
equally assigning to each CPU in a cluster of CPUs a set of scanlines in a similar manner that what was proposed in [2]. The measurements were performed using a cluster of four AMD processors where each is of a similar specification to the one used in the previous experiment. Figure 4.13 shows the average time of generating one disparity map while varying the number of CPUs used in parallel. For comparison, we tested the DP based techniques of [16,147] along with the normalized correlation approach. We applied each of these algorithms to several VGA video sequences with maximal disparity ranges of 100 and 150 respectively and the average time is plotted. It is clear that the correlation approach is the fastest and achieve the highest frame rate; however, the latency introduced by the sparse DP versions is minimized. The two DP approaches can achieve now an average speed corresponding to 1.5 to 3 frames per seconds (fps) depending on the disparity range. When compared to the original approaches, the computation of the dense depth maps would have required at least twice the number of CPUs to reach the frame rate obtained with sparse stereo. Consequently, it makes sense to use the combination of content adaptive image mesh approximation along with sparse DP in a TPTA scenario. The application of mesh approximation will generate the mesh needed to render the scenes on a GPU and reduce the number of points to be processed. The usage of sparse DP will increase the fidelity of the reconstructed scenes when compared to the correlation scheme while minimizing the latency.

4.6 Summary

A scheme was presented in this chapter to accelerate stereo-based scene acquisition. The reference image of a stereo pair is first meshed using the content adaptive meshing scheme of images described in Chapter 3. Then, the disparity values of the nodes of the mesh (non-uniform samples) are computed by developing a sparse stereo matching strategy based on dynamic programming. Sparse DP takes into account the sparsity of the image in its optimization. The matching costs are only evaluated at the non-uniform samples of the image. The smoothness cost in its turn is adapted to take the distance of the samples into account. This will make DP increases the smoothness penalty between the close non-uniform samples since they are located near the edges of the image and vice versa. The tests that were conducted show that the non-uniform samples are sufficient to recover the dense disparity map of the scene with an acceptable error. In addition, they show that using sparse DP, depth maps are now obtained up to 50% faster than with using the original DP algorithms which reduces the latencies induced and makes them applicable in TPTA scenarios.



(a) Sparse Algorithms



(b) Original Algorithms

Figure 4.13: Performance of different stereo matching algorithms depending on the number of processors used. In the test, VGA sequences were used. In the upper row, the maximum disparity level is 100 while it is 150 in the lower.

Table 4.3: Evaluation of the proposed sparse DP approach applied to the algorithms of [147, 16] with several other DP techniques using the Middlebury web-tool at 0.5 pixels threshold. The sparse algorithms are designated in the table by Sparse DP I and Sparse DP II respectively.

Algorithm	Avg.	Tsukuba			Venus			Teddy			Cones		
	Rank	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
DP of [147]	28.3	24.2 $_{34}$	26.0 $_{35}$	24.9 $_{34}$	10.9 $_{26}$	12.1 $_{26}$	27.6 $_{32}$	19.6 $_{23}$	27.0 $_{25}$	33.0 $_{21}$	16.5 $_{28}$	23.7 $_{27}$	29.5 $_{29}$
DP of [146]	28.4	19.0 $_{22}$	20.7 $_{27}$	17.5 $_{14}$	12.7 $_{29}$	14.0 $_{29}$	26.1 $_{31}$	26.3 $_{31}$	32.5 $_{31}$	36.8 $_{29}$	23.7 $_{34}$	29.9 $_{34}$	31.5 $_{30}$
Sparse DP I	31.8	13.8 $_{16}$	16.0 $_{17}$	28.9 $_{35}$	13.2 $_{31}$	14.5 $_{31}$	36.3 $_{37}$	30.0 $_{33}$	36.4 $_{34}$	45.5 $_{34}$	28.1 $_{38}$	34.3 $_{37}$	43.4 $_{38}$
DP of [140]	31.8	22.4 $_{32}$	23.1 $_{31}$	22.3 $_{28}$	12.1 $_{28}$	12.9 $_{28}$	21.7 $_{27}$	32.4 $_{36}$	38.9 $_{36}$	45.6 $_{35}$	23.7 $_{35}$	30.8 $_{35}$	31.7 $_{31}$
DP of [16]	32.0	19.6 $_{26}$	20.6 $_{26}$	22.8 $_{29}$	23.5 $_{39}$	24.3 $_{38}$	32.8 $_{33}$	30.0 $_{34}$	36.3 $_{33}$	36.1 $_{28}$	22.0 $_{33}$	29.6 $_{33}$	33.7 $_{32}$
Sparse DP II	34.7	17.8 $_{19}$	19.9 $_{27}$	32.5 $_{40}$	19.0 $_{42}$	20.0 $_{42}$	42.2 $_{43}$	32.5 $_{41}$	38.3 $_{39}$	46.5 $_{41}$	31.5 $_{43}$	37.1 $_{43}$	48.4 $_{44}$

Table 4.4: Time Evaluation in seconds of several stereo matching algorithms.

Image	Size	Disparity Range	DP of [16]		DP of [147]		Correlation	
			Sparse	Original	Sparse	Original	Sparse	Original
Tsukuba	384×288	15	0.06	0.09	0.22	0.47	0.019	0.07
Venus	434×383	20	0.13	0.18	0.38	0.93	0.03	0.11
Teddy	450×375	59	0.37	0.53	1.27	2.45	0.12	0.38
Cones	450×375	59	0.42	0.52	1.27	2.45	0.17	0.39
Figure 4.7a	640×480	100	0.94	1.52	2.24	6.6	0.19	0.42
Figure 3.2a	640×480	150	1.49	2.58	3.07	9.33	0.38	0.96

Chapter 5

Motion Estimation

In a general TPTA scenario, the teleoperator is subject to navigate in the remote scene in order to fulfill its tasks. Determining the pose of the teleoperator in the remote environment is yet another important problem in machine vision since these values are necessary for several tasks as tracking its position and registration of the 3D models. Determining an accurate estimate of the motion leads, however, to more computational requirements. The reason for that is the numerous terms that have to be accounted for in the mathematical formulation of the problem. The aim of this chapter is to suggest a new approach that enhances the motion estimation of a moving camera without significantly increasing the complexity of the underlying problem.

5.1 Overview of Motion Estimation

Extracting the motion parameters of a moving camera is a significant aim in machine vision. Based on some features in a sequence of images, it is possible to obtain the initial estimates of the 3D structure of a scene along with the corresponding camera positions. The pose of a camera is composed of a 3×3 rotation matrix $\mathbf{R}$ that describes the orientation of the camera in the real world and a 3×1 translation vector $\mathbf{t}$ which reflects the distance between the origin of the camera coordinate system and that of the world coordinate system. The pose with the intrinsic parameters form the camera matrix which is of size 3×4 and it is written in the from $\mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]$, see Equation (2.1). In this chapter, this notation will be slightly modified and the camera matrix at a time instinct i will be expressed as

$$\mathbf{K}_i\,[\mathbf{I}_3 \mid \mathbf{0}] \cdot \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}^T & 1 \end{bmatrix} = \mathbf{K}_i\,[\mathbf{I}_3 \mid \mathbf{0}] \cdot \mathbf{M}_i, \tag{5.1}$$

where $\mathbf{I}_3$ stands for the 3×3 identity matrix and $\mathbf{0}$ is the 3×1 zero vector. The 4×4 matrix $\mathbf{M}_i$ represents the position of the camera. It describes the rigid transformation between the coordinate system of the world and that of the camera. Determining the motion can be also stated as estimating the rigid transformation $\mathbf{M}_i$ that maps the coordinates of the camera to that of the world coordinate system. To simplify the notations, the first position of the camera will be set in this chapter as the origin of the world, i.e. $\mathbf{M}_1 = \mathbf{I}_4$. To determine the position at instinct i, it is necessary to first find the relative rigid transformation $\mathbf{M}_i^{i-1}$ that maps this position to the coordinate system of the previous time instinct $i - 1$. Then, the motion of the camera with respect to the origin of the world can be written as the product of $\mathbf{M}_i^{i-1}$ with all the relative homographies of the previous poses

$$\mathbf{M}_i = \prod_{j=i}^{2} \mathbf{M}_j^{j-1}. \tag{5.2}$$

In the case of a single moving camera with known intrinsic parameters, determining the relative pose boils down to estimating the essential matrix between the consecutive views. The essential matrix represents the algebraic formulation of the epipolar geometry constraint that was introduced in Chapter 4 and depicted in Figure 4.1. It incorporates all the information needed to extract the rigid motion parameters. Several techniques have been developed to compute this entity, starting from the famous eight-point algorithm that was formulated by Longuet-Higgins in [158] arriving to the seven-, six-, and five-point algorithm [20, 159, 160, 161] and finally to Gauss-Newton iterative techniques [162, 26]. Once the essential matrix is computed, the motion can be extracted by computing its singular value decomposition (SVD). The factorization of this entity leads, however, to four possible solutions of the motion. To resolve this ambiguity, the 3D coordinates of any pixel in the image are computed with these possibilities. The right solution of the motion is the one that leads to the 3D point lying in front of both cameras. This is usually referred to in the literature as the chieralities test [20, 160].

In a TPTA scenario, the teleoperator is equipped with a stereo camera. To benefit from that, one can utilize the additional information provided by the movement of the two cameras of the rig to remove the ambiguity in the solution. In this case, it is necessary to compute the tensors or the compatible essential matrices between the different views of the two cameras as was illustrated in [20, 27] for example. The motion estimate of the stereo camera is in this case unique but it is computed up to a scaling factor. Therefore, it is necessary to follow such methods with an approach to recover the scale. One possible way is to integrate some inertial sensors or a global positioning system (GPS) on the teleoperator to determine the transformation between the coordinate system of the cameras and that of the real world as was done in [163] for example.

Fortunately, the problem in hand have several other constraints that are beneficial for the computations of the motion. As the teleoperator moves, a partial 3D model is constructed in each frame using the sparse stereo reconstruction technique developed in the previous chapter. The intrinsic parameters of the camera can be safely assumed to be known by using the technique proposed in Chapter 2 for variable zoom lenses or using a technique like [74] for monofocal lenses. The extrinsic parameters of the stereo rig are also known since the orientation between the two cameras of the stereo rig is fixed. Therefore, the relative Euclidean position between the stereo camera (teleoperator) and each reconstructed partial 3D model is known. To compute the relative motion between the consecutive views of the teleoperator, it is sufficient to estimate the 3D to 3D rigid transformation between the points of the partial 3D models. This problem is referred to in photogrammetry as the *absolute orientation* and can be better explained by looking at Figure 5.1. Suppose that $\mathbf{Q} = \{\mathbf{P}_1, \cdots, \mathbf{P}_N\}$ is a set of N 3D points in the reference coordinate system. Let $\mathbf{Q}^i$ be the homogeneous coordinates of the points in $\mathbf{Q}$ with respect to the current position of the stereo camera and let $\mathbf{Q}^{i-1}$ be the ones with regard to the previous one. The relative motion between the views is computed in this case by solving the following least-squares problem

$$\mathbf{Q}^{i-1} = \mathbf{M}_i^{i-1} \cdot \mathbf{Q}^i, \tag{5.3}$$

which is nothing but the rigid transformation between the two partial 3D models reconstructed with respect to the different camera positions. The absolute orientation is the process of determining the rigid transformation between two sets of corresponding 3D points. This problem is usually solved using quaternions to represent the rotation matrix or by applying the SVD [164, 165,166,167]. To determine the corresponding points for the computations, it is necessary to apply image feature detection and tracking techniques. These are usually susceptible to errors which make the linear solution computed from an absolute orientation method erroneous and requires correction.

To refine the motion, the obtained values can be used as initial guesses in the iterative closest point (ICP) technique that tries to find iteratively the best alignment of the two three dimensional models [168, 169, 170, 171]. Another way is to apply the bundle adjustment (BA) formulation to minimize the distance between the reprojected pixels of the 3D points and the corresponding pixels determined by the feature detector. The second strategy will present the main focus of this chapter.

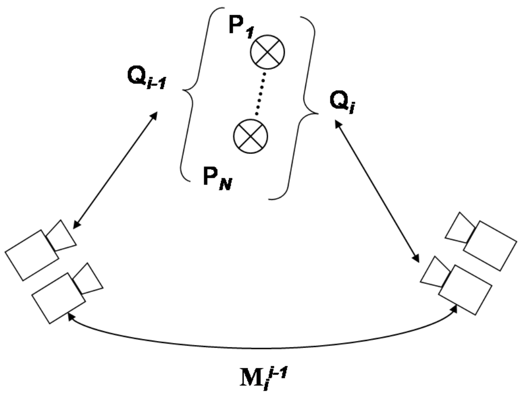


Figure 5.1: The 3D homogeneous coordinates of a static set of points $\mathbf{Q}$ are constructed with respect to each position of the stereo camera, i.e. $\mathbf{Q}_i$ and $\mathbf{Q}_{i-1}$. The transformation between the coordinates in the two positions is equivalent to the relative motion between the two stereo cameras, i.e. $\mathbf{M}_i^{i-1}$.

5.2 Problem Formulation

Bundle adjustment is to jointly adjust some noisy estimates of the 3D structure along with the motion of the cameras by correcting the bundle of rays simultaneously in all images. Given are some N point correspondences which are pre-computed using a feature detector and tracked over a sequence of images. In addition, the initial estimate of the 3D structure that the points form in space is provided along with the L initial poses of the cameras that represent where each of the images was captured. Using these initial estimates, a point from the 3D structure is projected on an image at a different position from the pre-computed feature point. An example of this setup is illustrated in Figure 5.2. The distance between the two points is called the reprojection error. The core of BA is to correct the motion estimates of the camera along with the 3D structure so that the reprojection error is minimized over all the points in all images. The cost function E to be optimized should reflect this setup. Using the perspective projection camera model described in Equation (2.1), it is written as

$$E = \sum_{i=1}^{L} \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j^i, \pi \mathbf{M}^i \mathbf{P}_j\right)^2, \tag{5.4}$$

where $\mathbf{p}_j^i$ are the homogeneous coordinates of the j^{th} 2D measured points in the i^{th} image (camera), $\mathbf{M}^i$ is a 4×4 transformation matrix representing the location of the i^{th} camera in the world coordinate system, $\mathbf{P}_j$ are the homogeneous coordinates of the j^{th} point in the 3D structure, π is a mapping from $\mathbb{R}^4$ to $\mathbb{R}^3$ to obtain the correct the dimensions and d is the geometrical distance measure between $\hat{\mathbf{p}}^i_j = \pi\mathbf{M}^i\mathbf{P}_j$ and the corresponding measured feature point $\mathbf{p}^i_j$. The measure d can be any scale invariant distance such as the Mahalanobis or the cosine distances. The cosine distance is used here to assess the error between reprojected image point $\hat{\mathbf{p}}^i_j$ using the estimates of the variables and the corresponding measured point $\mathbf{p}^i_j$. It describes the angle formed by $\hat{\mathbf{p}}^i_j$, the camera center $\boldsymbol{O}_i$ and $\mathbf{p}^i_j$, see Figure 5.2. It is expressed by

$$\mathrm{d}\left(\mathbf{p}^i_j, \hat{\mathbf{p}}^i_j\right) = 1 - \frac{\mathbf{p}^{i^T}_j \cdot \hat{\mathbf{p}}^i_j}{\|\mathbf{p}^i_j\| \cdot \|\hat{\mathbf{p}}^i_j\|}, \tag{5.5}$$

where $\|\cdot\|$ denotes the 2 norm. The cosine distance is invariant to scale and takes the correlation of the data into account. We chose to apply it instead of the Mahalanobis distance for two reasons. First, it is less complex than Mahalanobis in the sense that it makes the computations less demanding; i.e. it does not require the derivation of the covariance matrix, but just the norm of the vectors. Second, it has been shown in recent surveys that the cosine distance performs as well as the Mahalanobis if not better in detecting the similarities among the different features in the images, see [172,173,174]. It is worth to note that the performance of the Mahalanobis distance can be improved by choosing the optimal covariances for the problem in hand. Computing the suitable covariances, however, is a bottleneck by itself for it requires either some trial and error experiments or the application of machine learning algorithms. This fact additionally justifies the employment of the cosine distance of Equation (5.5).

BA is widely used in structure from motion problems to refine the estimates obtained from a single moving camera. Examples of algorithms employing this scheme are [20, 175, 176, 161, 177] while an excellent survey is presented in [178] discussing different methodologies to implement it. The goal of this chapter, however, is to extract the motion of the teleoperator to which a stereo rig is integrated. The structure can be assumed to be fixed since it can be computed using stereo reconstruction as was stated in the previous section. Doing so and keeping in mind that the motion variables of the different positions of the rig are independent from each other, renders the problem to a set of L *pose estimation* problems, one for each camera. This will allow us to rewrite the cost function of BA as

$$E_i = \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}^i_j, \pi\mathbf{M}_i\mathbf{P}_j\right)^2, \tag{5.6}$$

which has to be minimized separately for each camera. As a summary, estimating the pose of a moving stereo camera consists of three steps:

1. Calculate the relative motion of the teleoperator at instinct i with respect to that of the previous instinct $i-1$ by solving Equation (5.3).

2. Compute the relative position of the teleoperator to the world coordinate system using Equation (5.2).

3. Refine the pose estimate by minimizing non-linearly Equation (5.6).

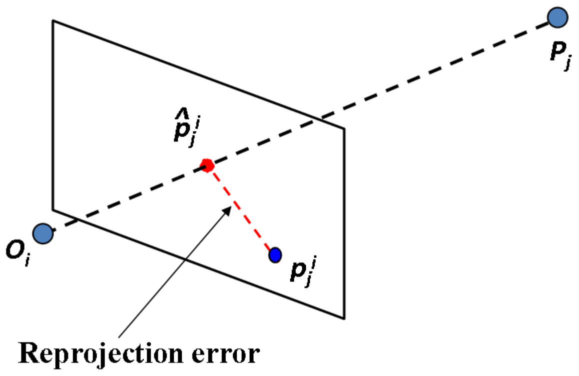


Figure 5.2: Structure of the problem.

5.3 Related Work

With the reformulation of the cost function of bundle adjustment in the form of Equation (5.6), the problem turns equivalent to estimating the relative pose between the 3D structure and the corresponding points on the image. In this direction, one can find several linear and non-linear algorithms that can accomplish this task as the techniques that were presented in [179, 180, 181, 182]. In our case, the linear initial estimate is computed via absolute orientation. The task is to refine this obtained value by employing some Newton based non-linear minimization schemes to overcome the errors introduced from the false matches or the inaccuracies incurred from the calibration errors. Before doing that, it is necessary to first describe how the motion variables are usually parameterized in this kind of problems.

5.3.1 Parametrization of the Variables

Assuming that the camera is calibrated and that the 3D structure is known, the cost function E_i has to be minimized for each new position of the camera, i.e. each $\mathbf{R}^i$ and $\mathbf{t}^i$. To refine the motion, its variables have to be parameterized in such a way that the minimization algorithm can avoid the local minima. The translation vector consists of 3 entries corresponding to its 3 degrees of freedom (DOF). To parameterize it, it is sufficient to take its entries as the variables. The rotation in the 3D space is described by a 3×3 matrix consisting in a total of 9 entries although it has only 3 DOF. Parameterizing the 3D rotation using the 3 Euler angles is not a good idea in practice since it results in numerical instabilities due to the singularities and the uneven regions that the angles cover. A rotation can be expressed using unit quaternions or using local perturbations of an existing rotation. However, each of these parameterizations has its own drawback. The unit quaternion has to be constrained so that the vector has a unit norm while the update step of the rotation using local perturbations does not result exactly in a rotation matrix [178]. In general, a rotational transformation in $\mathbb{R}^3$ is represented by the elements of the special orthogonal group SO_3. Its associated Lie algebra $\mathbf{so_3}$ is the set of 3×3 skew-symmetric matrices which are considered as the tangent space of SO_3 at the identity. The isomorphism $\mathbf{\Omega}$ that allows to identify $\mathbf{so}_3$ with $\mathbb{R}^3$ is defined as

$$\mathbf{\Omega}\left(\omega^i\right) : \mathbb{R}^3 \longrightarrow \mathbf{so}_3, \begin{bmatrix} \omega_1^i \\ \omega_2^i \\ \omega_3^i \end{bmatrix}_\times = \begin{bmatrix} 0 & -\omega_3^i & \omega_2^i \\ \omega_3^i & 0 & -\omega_1^i \\ -\omega_2^i & \omega_1^i & 0 \end{bmatrix}. \tag{5.7}$$

To connect the Lie algebra to the Lie group, the exponential map is usually used. The rotation matrix $\mathbf{R}^i$ can thus be expressed in the form $\mathbf{R}^i = \exp\left(\mathbf{\Omega}\left(\omega^i\right)\right)$. It can be written using the Taylor series expansion of the exponential function as

$$\mathbf{R}^i = \exp\left(\mathbf{\Omega}\left(\omega^i\right)\right) = \mathbf{I} + \mathbf{\Omega}\left(\omega^i\right) \cdot \frac{\sin\left(\|\omega^i\|\right)}{\|\omega^i\|} + \mathbf{\Omega}^2\left(\omega^i\right) \cdot \frac{1-\cos\left(\|\omega^i\|\right)}{\|\omega^i\|^2}. \tag{5.8}$$

The term to the right is known as the Rodrigues formula of the rotation matrix. If the magnitude of the vector ω^i is small, it can be approximated with $\mathbf{I} + \mathbf{\Omega}\left(\omega^i\right)$ which is nothing but the local perturbation around $\mathbf{R}^i$ that are usually used in BA [178]. Assuming that the magnitude of the vector is small is not always accurate especially if the initial estimate of the motion is random or if it contains high amount of noise. In our implementations of the Newton-based schemes in the vector space, we will be applying the whole term reflected in the above equation to have an accurate parametrization of the 3D rotation. Another reason is that the Manifold based scheme that will be introduced results in the accurate rotation matrix. Thus, taking the whole term makes the comparison among the techniques fair.

5.3.2 Levenberg-Marquardt in the Vector Space

The Levenberg-Marquardt (LM) algorithm is a Newton based optimization scheme that leads to the least-squares solution of the problem in a non-linear sense [183, 184]. In order to apply LM in the given problem, the cost function E_i needs to be formulated in the form $E_i = \mathbf{g}_i^T \mathbf{g}_i$ where $\mathbf{g}_i = [g_{1,i}, \ldots, g_{N,i}]^T \in \mathbb{R}^N$ and $g_{j,i} = \mathrm{d}\left(\mathbf{p}_j^i, \pi \mathbf{M}_i \mathbf{P}_j\right)$. We will be also noting by $\mathbf{a}_i = \left[\omega^{iT} \quad \mathbf{t}^{iT}\right]^T \in \mathbb{R}^6$ as the vector that represents the 6 parameters of the motion of a camera. LM tries to find the solution vector $\widetilde{\mathbf{a}}$ that minimizes Equation (5.6) iteratively. The update step at the k^{th} iteration can be expressed as

$$\mathbf{a}_i^{(k+1)} = \mathbf{a}_i^{(k)} - \left(\mathbf{J}_{\mathbf{g}_i}^T \cdot \mathbf{J}_{\mathbf{g}_i} + \mu \cdot \mathbf{I}_6\right)^{-1} \mathbf{J}_{\mathbf{g}_i}^T \cdot \mathbf{g}_i. \tag{5.9}$$

$\mathbf{I}_6$ is the 6×6 identity matrix, μ is the damping term that controls the step size of the iteration and $\mathbf{J}_{\mathbf{g}_i}$ is the Jacobian matrix which is computed by taking the first order derivatives of $\mathbf{g}_i$ with respect to $\mathbf{a}_i$

$$\mathbf{J}_{\mathbf{g}_i} = \begin{bmatrix} \frac{\partial g_{1,i}}{\partial \mathbf{a}_i^T} \\ \vdots \\ \frac{\partial g_{N,i}}{\partial \mathbf{a}_i^T} \end{bmatrix} \in \mathbb{R}^{N \times 6}. \tag{5.10}$$

The term $\mathbf{J}_{\mathbf{g}_i}^T \cdot \mathbf{J}_{\mathbf{g}_i}$ is the approximation of the Hessian matrix used in the Gauss-Newton algorithm. Therefore, $\mu \cdot \mathbf{I}_6$ can be also thought of as a regularization term for the Hessian in case if it is close to singularity. Due to this term, LM has the properties of both the steepest descent and the Gauss-Newton iterative techniques. Similarly to steepest descent, LM converges slowly to the solution if the initial estimate is far away from the real solution while it converges fast like Gauss-Newton methods when the initial estimate is close to the real solution. This is the main reason why LM is widely used as an optimization scheme in structure from motion problems. The reader interested in a thorough analysis of the method may refer to [183, 184, 178, 185, 20, 186].

5.3.3 Newton Algorithm in the Vector Space

The aim of a Newton-type algorithm is to find the minimum of a given cost function in an iterative scheme. In the case of the given problem, the update step used to minimize Equation (5.6) has the following form

$$\mathbf{a}_i^{(k+1)} = \mathbf{a}_i^{(k)} - \mathbf{H}_{E_i}^{-1} \cdot \nabla E_i, \tag{5.11}$$

where k is the iteration number as before, $\nabla E_i = 2\mathbf{J}^T_{\mathbf{g}_i} \cdot \mathbf{g}_i$ is the gradient of the cost function E_i and $\mathbf{H}_{E_i} \in \mathbb{R}^{6\times 6}$ is the corresponding Hessian. The matrix $\mathbf{H}_{E_i}$ is expressed by

$$\mathbf{H}_{E_i} = 2\mathbf{J}^T_{\mathbf{g}_i} \cdot \mathbf{J}_{\mathbf{g}_i} + \sum_{j=1}^{N} \left(g_{j,i} \cdot \mathbf{H}_{g_{j,i}} \right) . \tag{5.12}$$

The matrix $\mathbf{H}_{g_{j,i}}$ denotes the Hessian with respect to each term of $\mathbf{g}_i$, i.e. $\frac{\partial}{\partial \mathbf{a}_i^{\mathrm{T}}} \frac{\partial g_{j,i}}{\partial \mathbf{a}_i}$. By setting these terms to zero, the approximated Hessian obtained $\hat{\mathbf{H}}_{E_i} = 2\mathbf{J}^T_{\mathbf{g}_i} \cdot \mathbf{J}_{\mathbf{g}_i}$ is what is used in a Gauss-Newton iterative technique. By adding the damping term $\mu \cdot \mathbf{I}_6$ to $\hat{\mathbf{H}}_{E_i}$ leads us to the approximate Hessian used in LM, see Equation (5.9). The complexity induced by the computation of $\mathbf{H}_{g_{j,i}}$ is the main reason that makes LM more favorable in the optimization of BA. However, failing to take $\mathbf{H}_{g_{j,i}}$ into account will make the accuracy of the motion estimate diverges more from the global solution especially if the initial estimate of the motion is noisy or far from the global value. The main reason is that taking the full Hessian notation is more favorable to the large residual problems where the terms to be minimized, i.e. reprojection errors, are larger than the minimizer, i.e. the motion. A detailed study of this point is conducted in [187]. The LM scheme is, however, the state of the art technique in structure from motion problems since it is usually hoped that the initial estimates of the variables are accurate enough for the system to converge to the global solution. It is worth to note that it is also possible to add a damping factor $\mu \cdot \mathbf{I}$ to $\mathbf{H}_{E_i}$ like in Equation (5.9) to regularize the Hessian if it is close to singularity or to vary the speed of convergence depending on how far is the initial estimate from the global one. A good description on how to perform such adjustments is found in [185].

5.4 Newton Minimization on the Manifold

Computation of the second order terms in a Newton algorithm, i.e. right addend in Equation (5.12), is not favorable due to the complexity arising from the computation of the Hessian. To get a glimpse of how the computational complexity rises, the number of floating point operations (FLOPS) required per camera per iteration are illustrated in Table 5.1. As can be noticed, including these terms makes the complexity required for the motion refinement rises to 5 times more when compared to that needed in LM. The aim of this section is, therefore, to propose some optimization methods to enhance the accuracy of the motion estimate of a calibrated camera without a significant increase in the complexity. To do that, a Newton minimization scheme on the *manifold* will be derived.

A manifold is a topological space in which each point has a neighborhood that looks like the Euclidean space. Such an entity can be illustrated by a complex surface in space that is sometimes difficult to be expressed mathematically. Designing an optimization scheme on a manifold can be visualized as making the minimization iteratively walk on the surface of the curve described by the variables until the minimum is achieved. In order to perform such operations, it is necessary that the manifold possesses a special structure. It has to be differentiable and smooth so that notions like distances and angles, necessary to determine the direction of optimization in a Newton scheme, can be defined. Examples are the *Riemannian* manifolds where each has the nice property of being equipped with a tangent space that allows the transitions from a point to another one to be smooth. An instance of a Riemannian manifold is the special orthogonal group SO_3 that describes the $3\times$ 3 rotations matrices, see Equation (5.7). The tangent space to this group is its associated Lie algebra $\mathbf{so}_3$ which is related to the latter using the exponential map. To derive a Newton-type method, it is necessary to compute first and second order derivatives. An intrinsic Newton method, that is, a Newton method which is intrinsically defined on the Riemannian manifold without referring to any embedding vector space, requires the notion of a Riemannian gradient and a Riemannian Hessian. This idea often requires an overhead of differential geometric machinery. Here, a projected Newton-type method is formulated instead.

Definition 5.1 *A projected Newton-type algorithm exploits a local parameterization of the variables on the manifold. The direction and step size of the optimization are computed using the associated tangent space. The result is then projected back on the manifold to update the variables.*

In contrast to the intrinsic Newton methods, a projective Newton method is usually performed in two steps that are explained by the above definition. The mechanism of such schemes is illustrated in Figure 5.3. To apply this approach, however, a suitable local parameterization of the rigid motion of the camera must be first determined. For good references that provide detailed discussions about Lie theory and differential geometry, the interested reader may refer to [188, 189, 190, 191, 192].

The main motivation behind adopting such a strategy in this thesis is the fact that optimization on the manifold has proven recently to be an extreme success when applied to variant engineering problems. In the following, some of the latest developments in this domain will be stated. In [162, 26], the essential matrix of a single moving camera was determined by parameterizing the latter on the *essential* manifold. In [27], the notion of the essential matrix was extended to compute a group of compatible essential matrices by intersecting several essential manifolds. In [193], the special orthogonal group was used to impose the constraints of the rotation matrix in independent component analysis while it was applied in [194] to register the 3D objects as an alternative to the ICP algorithm. Finally, the *Grassmann* manifold was integrated with principal component analysis to track moving objects in [195]. As a consequence, it makes sense to benefit from this strategy and apply it to estimate the pose of a moving stereo camera.

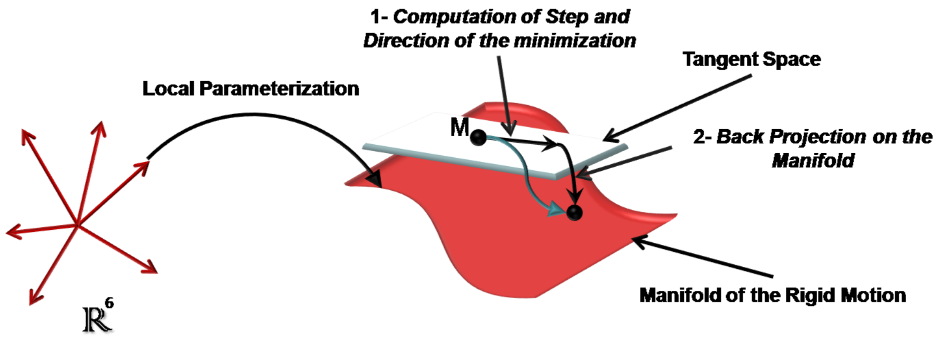

1- Computation of Step and Direction of the minimization
Tangent Space
Local Parameterization
M
2- Back Projection on the Manifold
Manifold of the Rigid Motion
$\mathbb{R}^6$

Figure 5.3: Projective Newton-type algorithm on the manifold. A local parameterization maps the variables from $\mathbb{R}^6$ to the manifold of the rigid motion. The minimization is then performed in two steps: The optimization direction and step size are computed on the tangent space to the manifold. The calculated values are then projected back on the manifold to update the motion.

5.4.1 Parametrization of the Motion on the Manifold

The rigid motion in $\mathbb{R}^6$ is represented by the elements of the special Euclidean group SE_3 which consists of all the 4×4 matrices $\mathbf{M} \in SE_3$ of the form

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{5.13}$$

where $\mathbf{R} \in SO_3$ and $\mathbf{t} \in \mathbb{R}^3$ are the rotation and the translation of the camera, respectively[1]. The Lie algebra se_3 associated to SE_3 is the set of 4×4 matrices $\mathbf{m} \in se_3$

$$\mathbf{m} = \begin{bmatrix} \mathbf{\Omega}(\omega) & \zeta \\ \mathbf{0} & 0 \end{bmatrix}, \tag{5.14}$$

[1]Note that the index i designating the camera number in Equation (5.6) will be dropped for simplicity since the derivation is the same for any camera under consideration.

where $\mathbf{\Omega}(\omega) \in so_3$ is a skew symmetric-matrix as shown in Equation (5.7) and ζ is a vector in $\mathbb{R}^3$. The matrix $\mathbf{m}$ is related to $\mathbf{M}$ via the exponential map, i.e. $\mathbf{M} = \exp(\mathbf{m})$. Using the Taylor series expansion of the exponential, it can be easily shown that the translation can be written as

$$\mathbf{t} = \left(\mathbf{I} + \frac{1-\cos(\alpha)}{\alpha^2} \cdot \mathbf{\Omega}(\omega) + \frac{\alpha - \sin(\alpha)}{\alpha^3} \cdot \mathbf{\Omega}(\omega)^2\right) \cdot \zeta, \tag{5.15}$$

where $\alpha = \sqrt{\frac{1}{2}\text{trace}\left(\mathbf{\Omega}^T(\omega) \cdot \mathbf{\Omega}(\omega)\right)}$. The rotation $\mathbf{R}$ remains the same as was defined in Equation (5.8), i.e. $\mathbf{R} = \exp(\mathbf{\Omega}(\omega))$. To design a Newton-type optimization scheme on a manifold, it is necessary to employ ω and ζ to parameterize the rotation and the translation, respectively.

Proposition 5.1 *A smooth and local parametrization ϕ of SE_3 for the camera rigid motion $\mathbf{M}$ is defined as*

$$\phi_{(\mathbf{M})} : \mathbb{R}^6 \to SE_3, \phi_{(\mathbf{M})}(\mathbf{a}) = \exp\left(\begin{bmatrix} \mathbf{\Omega}(\mathbf{D}_1\mathbf{a}) & \mathbf{D}_2\mathbf{a} \\ \mathbf{0} & 0 \end{bmatrix}\right) \cdot \mathbf{M}, \tag{5.16}$$

where the matrices $\mathbf{D}_1 = [\mathbf{I}\ \mathbf{0}] \in \mathbb{R}^{3\times6}$ and $\mathbf{D}_2 = [\mathbf{0}\ \mathbf{I}] \in \mathbb{R}^{3\times6}$ select the appropriate variables. These matrices were introduced to keep a similar notation for the motion parameters, i.e. $\mathbf{a}$, *as was done previously in the vector space.*

Proof The proof of this proposition is based on the fact that se_3 is the tangent space to SE_3. By the definition of a Riemannian manifold, this space is smooth in the neighborhood of a given point on the manifold. ■

The special Euclidean group SE_3 is homeomorphic as a topological space to $\mathbb{R}^3 \times SO_3$, i.e. $SE_3 \cong \mathbb{R}^3 \times SO_3$ [192]. This can be simply visualized from the fact that the rigid motion can be thought of as two consecutive motions, i.e. a pure translation $\mathbf{t}$ followed by a pure rotation $\mathbf{R}$, where each can be chosen independently from the other. The vector containing the camera parameters can be written in this case as $\mathbf{a} = \left[\omega^T \quad \mathbf{t}^T\right]^T \in \mathbb{R}^6$ since Equation (5.15) reduces to $\mathbf{t} \approx \zeta$. The homeomorphism leads us to another parametrization of the rigid motion.

Proposition 5.2 *An equivalent smooth and local parametrization ϕ of $\mathbb{R}^3 \times SO_3$ to Proposition 5.1 for the camera rigid motion $\mathbf{M}$ is*

$$\phi_{(\mathbf{M})} : \mathbb{R}^6 \to \mathbb{R}^3 \times SO_3, \phi_{(\mathbf{M})}(\mathbf{a}) = \begin{bmatrix} \exp(\mathbf{\Omega}(\mathbf{D}_1\mathbf{a})) & \mathbf{D}_2\mathbf{a} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \mathbf{M}. \tag{5.17}$$

Proof The proof of this proposition is due to the homeomorphism between the two topological spaces. It will be conducted by contradiction. By the definition of homeomorphism, there exist a one to one bijective function between the two spaces SE_3 and $\mathbb{R}^3 \times SO_3$. Assuming that the mapping presented by this proposition is not smooth and local then, the one given in Proposition 5.1 is the same due to the bijective function. The latter is not possible due to the properties of the tangent space to a Riemannian manifold.

$\blacksquare$

5.4.2 Minimization of the Cost Function

A projected Newton-type method computes a Newton step for the composition $E \circ \phi$ of the cost function E with the local parametrization ϕ and maps (projects) the update step back to the manifold using the parametrization again, see Figure 5.3. We slightly abuse the notation to write $E \circ \phi_{(\mathbf{M})}$ to denote the mapping

$$E \circ \phi_{(\mathbf{M})} : \mathbb{R}^6 \to \mathbb{R}, \tag{5.18}$$

which corresponds here to the cost function illustrated in Equation (5.6) to be minimized on the manifold. It is given by

$$\begin{aligned} E \circ \phi_{(\mathbf{M})}(\mathbf{a}) &:= \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j, \hat{\mathbf{p}}_j\right)^2 \qquad (5.19) \\ &= \sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j, \pi \mathbf{\Lambda}_\phi \mathbf{M} \mathbf{P}_j\right)^2, \end{aligned}$$

where $\mathbf{\Lambda}_\phi$ is any of the multiplicands of $\mathbf{M}$ from Propositions 5.1 or 5.2, and $\mathrm{d}\left(\mathbf{p}_j, \hat{\mathbf{p}}_j\right)$ is the distance function depicted in Equation (5.5). As in any Newton-based minimization scheme, the formulations of the gradient and the Hessian need to be derived. The gradient $\nabla(E \circ \phi_{(\mathbf{M})})(0) \in \mathbb{R}^6$ of the cost function of the projective Newton-type scheme on the manifold can be calculated from

$$\left.\frac{\mathrm{d}}{\mathrm{d}\varepsilon}(E \circ \phi_{(\mathbf{M})})(\varepsilon \mathbf{a})\right|_{\varepsilon=0} = \nabla(E \circ \phi_{(\mathbf{M})})(0)^T \cdot \mathbf{a}, \tag{5.20}$$

while the Hessian matrix $\mathbf{H}_{E \circ \phi_{(\mathbf{M})}}(0) \in \mathbb{R}^{6 \times 6}$ of the local cost function evaluated at zero can be calculated via polarization using the quadratic form

$$\left.\frac{\mathrm{d}^2}{\mathrm{d}\varepsilon^2}(E \circ \phi_{(\mathbf{M})})(\varepsilon \mathbf{a})\right|_{\varepsilon=0} = \mathbf{a}^T \cdot \mathbf{H}_{E \circ \phi_{(\mathbf{M})}}(0) \cdot \mathbf{a}. \tag{5.21}$$

The details of this concept can be found for the interested reader in [189].

Proposition 5.3 *By applying the parameterization of SE_3 defined in Proposition 5.1, the explicit formula for the gradient of the local cost function at zero is given by*

$$\begin{aligned}\nabla(E\circ\phi_{(\mathbf{M})})(0) &= 2\sum_{j=1}^{N}\left(\underbrace{[-\mathbf{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]^T\frac{\partial \mathrm{d}\left(\mathbf{p}_j,\hat{\mathbf{p}}_j\right)}{\partial\hat{\mathbf{p}}_j}}_{\to\mathbf{J_g}}\underbrace{\mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)}_{\to\mathbf{g}}\right) \\ &= 2\mathbf{J}_{\mathbf{g}}^T\cdot\mathbf{g},\end{aligned} \tag{5.22}$$

where $\frac{\partial \mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)}{\partial\hat{\mathbf{p}}_j}\in\mathbb{R}^3$ *is the standard Euclidean derivative of the distance function shown in Equation (5.5) with respect to* $\hat{\mathbf{p}}_j$. *In a similar manner to the gradient, the explicit formula of the Hessian is written as*

$$\begin{aligned}\mathbf{H}_{E\circ\phi_{(\mathbf{M})}}(0) &= 2\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}} \\ &+ 2\sum_{j=1}^{N}\left(\mathbf{D}_1^T\mathbf{\Omega}\left(\frac{\partial \mathrm{d}\left(\mathbf{p}_j,\hat{\mathbf{p}}_j\right)}{\partial\hat{\mathbf{p}}_j}\right)[\mathbf{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]\mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)\right) \\ &+ 2\sum_{j=1}^{N}\left([-\mathbf{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]^T\frac{\partial^2 \mathrm{d}\left(\mathbf{p}_j,\hat{\mathbf{p}}_j\right)}{\partial\hat{\mathbf{p}}_\mathbf{j}^2}[-\mathbf{\Omega}(\pi\mathbf{M}\mathbf{P}_j)\ \ \mathbf{I}_3]\mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)\right),\end{aligned} \tag{5.23}$$

where $\frac{\partial \mathrm{d}^2(\mathbf{p}_j,\hat{\mathbf{p}}_j)}{\partial\hat{\mathbf{p}}_j^2}\in\mathbb{R}^{3\times3}$ *is the second order Euclidean derivative of the distance function with respect to* $\hat{\mathbf{p}}_j$.

Proof The above proposition can be proven by combining the definitions of the gradients and Hessians illustrated in Equations (5.20) and (5.21) with the parameterization suggested in Proposition 5.1. ■

Proposition 5.4 *Using the parameterization of* $\mathbb{R}^3\times SO_3$ *defined in Proposition 5.2, the explicit formula for the gradient of the local cost function at zero is*

$$\begin{aligned}\nabla(E\circ\phi_{(\mathbf{M})})(0) &= 2\sum_{j=1}^{N}\left(\underbrace{\left(\mathbf{D}_2^T+\mathbf{D}_1^T\mathbf{\Omega}^T(\mathbf{R}\mathbf{P}_j)\right)\frac{\partial \mathrm{d}\left(\mathbf{p}_j,\hat{\mathbf{p}}_j\right)}{\partial\hat{\mathbf{p}}_j}}_{\to\mathbf{J_g}}\underbrace{\mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)}_{\to\mathbf{g}}\right) \\ &= 2\mathbf{J}_{\mathbf{g}}^T\cdot\mathbf{g},\end{aligned} \tag{5.24}$$

while that of the Hessian is in the form

$$\begin{aligned}\mathbf{H}_{E\circ\phi_{(\mathbf{M})}}(0) &= 2\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}} \\ &+ 2\sum_{j=1}^{N}\left(\mathbf{D}_1^T\mathbf{\Omega}\left(\frac{\partial \mathrm{d}\left(\mathbf{p}_j,\hat{\mathbf{p}}_j\right)}{\partial\hat{\mathbf{p}}_j}\right)\mathbf{\Omega}(\mathbf{R}\mathbf{P}_j)\mathbf{D}_1\mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)\right) \\ &+ 2\sum_{j=1}^{N}\left(\left(\mathbf{D}_2^T-\mathbf{D}_1^T\mathbf{\Omega}^T(\mathbf{R}\mathbf{P}_j)\right)\frac{\partial^2 \mathrm{d}\left(\mathbf{p}_j,\hat{\mathbf{p}}_j\right)}{\partial\hat{\mathbf{p}}_\mathbf{j}^2}(\mathbf{D}_2-\mathbf{\Omega}(\mathbf{R}\mathbf{P}_j)\mathbf{D}_1)\mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)\right).\end{aligned} \tag{5.25}$$

Proof The proof is similar to that of Proposition 5.4. In this case, the motion matrix $\mathbf{M}$ is split between the rotation matrix and translation parts when calculating the derivatives. This was done to simplify the notation since in contrast to the parameterization of SE_3, the one defined in Proposition 5.2 allows that. ■

By considering the computed gradient and Hessian using any of the proposed parameterizations, one can observe that their form is compact. There is no lengthy computations in the evaluation of the first and second order derivatives of the distance function as in Equations (5.10) and (5.12) in the vector space. The only derivatives that need to be evaluated are the first and second order Euclidean derivatives of the cost function with respect to $\hat{\mathbf{p}}_j$, i.e. $\frac{\partial \mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)}{\partial \hat{\mathbf{p}}_j}$ and $\frac{\partial \mathrm{d}^2(\mathbf{p}_j,\hat{\mathbf{p}}_j)}{\partial \hat{\mathbf{p}}_j^2}$, which are simple to obtain. The reason for this is that the computation of the derivatives with respect to the motion parameters reduces to matrix multiplications which is a property of the projective Newton type optimization algorithms on Riemannian manifolds [189]. Looking at Equations (5.20) and (5.21), one can see that the derivation in this case is made with respect to the parameter ε at zero which designates the direction in the tangent plane to the manifold and not the parameter vector $\mathbf{a}$ itself. This can be illustrated in our derivations by the terms $[-\mathbf{\Omega}\left(\pi \mathbf{M}\mathbf{P}_j\right) \ \ \mathbf{I}_3]^T$ and $(\mathbf{D}_2^T + \mathbf{D}_1^T \mathbf{\Omega}^T\left(\mathbf{R}\mathbf{P}_j\right))$ where each represents the derivative with respect to the camera parameters using each of the two parameterizations. In addition, by comparing Equation (5.22) to (5.23) and Equation (5.24) to (5.25), we can see that a lot of terms needed for the computations of the Hessian have to be also computed for the gradient, e.g. $\mathbf{\Omega}\left(\pi \mathbf{M}\mathbf{P}_j\right)$, $\mathbf{\Omega}\left(\mathbf{R}\mathbf{P}_j\right)\mathbf{D}_1$ and $\frac{\partial \mathrm{d}(\mathbf{p}_j,\hat{\mathbf{p}}_j)}{\partial \hat{\mathbf{p}}_j}$.

5.4.3 The Algorithm

Given an initial estimate of the motion $\mathbf{M}$ of a camera with respect to the center of the world coordinate system, the proposed Newton algorithm that refines the motion will iterate the following update scheme:

Step 1: (Newton step) Compute the vector $\mathbf{a} \in \mathbb{R}^6$ by solving

$$-\left(\mu \cdot \mathbf{I} + \mathbf{H}_{E\circ\phi_{(\mathbf{M})}}(0)\right) \cdot \mathbf{a} = \nabla\left(E \circ \phi_{(\mathbf{M})}\right)(0) \tag{5.26}$$

Step 2: Update the motion parameters of each camera:

$$\mathbf{M} \leftarrow \mathbf{\Lambda}_\phi \cdot \mathbf{M} \tag{5.27}$$

Step 3: Update the reprojections $\hat{\mathbf{p}}_j$ using the new value of the motion and compute the mean reprojection error.

Step 4: Repeat until convergence.

5.4.4 Analysis of the Derived Scheme

The projective Newton-based minimization algorithm is initialized by first solving the absolute orientation problem shown in Equation (5.3) to estimate the relative pose of the camera with respect to the previous one. The pose of the camera with respect to the world coordinate system is determined after that by multiplying the latter with all the relative previous poses as illustrated in Equation (5.2). Then, the method proceeds with the iterative scheme illustrated in the previous section. Each iteration, the algorithm performs an optimization step in Equation (5.26) on the tangent space of the manifold of the rigid motion followed by a projection step which is an analytic geodesic search described in Equation (5.27), see Figure 5.3. The proposed technique implements the Riemannian Newton algorithm on a small neighborhood of the set of local minima of the cost function E. Outside of this neighborhood where the Hessian is close to singularity or indefinite, the algorithm might fail. For this reason, Equation (5.26) is modified using the damping factor μ as in case of the LM technique to regularize the Hessian. This modification is necessary here to enlarge the attraction domain of the local minima and to control the speed of convergence of the algorithm. The direction of the step is determined via the Newton direction and it is zero only when the gradient is zero. Consequently, the algorithm converges to a critical point of E when the gradient is zero.

What still remains to be shown is that the proposed algorithm will converge to a minimum for a set of observed 3D points and an initial estimate of the motion M. The proof that will be used is the *global convergence theorem* derived in Chapter 6 of [196].

Theorem 5.5 *Let* $\mathbf{PN}$ *denotes a mapping from* $SE_3 \rightarrow SE_3$ *that calculates from the motion estimate* $\mathbf{M}^k$ *at the* k^{th} *iteration, the new estimate* $\mathbf{M}^{k+1}$*. The projective Newton algorithm is said to be convergent according to the global convergence theorem if it satisfies the following conditions:*

1. *The mapping* $\mathbf{PN}$ *is closed.*
2. *All the computed* $\mathbf{M}^k$ *by the mapping are contained in a compact set.*
3. *The mapping* $\mathbf{PN}$ *is a strictly decreasing function except at the solution.*

Proof

1. To prove the first condition of the theorem, we need to recall that each iteration of the proposed scheme is composed as said before of two mappings. The first one is from $SE_3 \rightarrow se_3$ and which is given

by the solution of the Newton step in Equation (5.26). The second is the mapping that projects back on the manifold, i.e. $se_3 \rightarrow SE_3$, and which is represented with Equation (5.27). These two mappings are continuous and smooth according to the properties of a Riemannian manifold; therefore, both mappings are closed. Since **PN** is composed of two closed mappings then the mapping defined by **PN** is also closed.

2. The mapping specified by **PN** generates some matrices of the rigid motion from the set SE_3 which is a compact set by the properties of a Riemannian manifold.

3. The regularization term represented by the damping factor μ in Equation (5.26) insures that the Hessian remains positive semi-definite. Hence, the direction of the minimization in this equation is always negative or decreasing. The step is zero only when the gradient is zero, the point at which the algorithm meets a fixed point (converges). Hence, the third condition of the theorem is also satisfied.

Since all the conditions are satisfied, the proposed scheme is globally convergent. The proof was conducted for the first parameterization stated in Proposition 5.1. A similar proof can be carried out for the second one. ■

5.4.5 Comparison of the Computational Complexity

Now that the algorithm was proven to be globally convergent, the next step is to evaluate its complexity and compare it to the state of the art methods depicted in Section 5.3. The main difference in the proposed scheme is that the optimization, i.e. evaluation of gradient and Hessian, is conducted on a Riemannian manifold. In order to emphasize on the computational complexity, we present in Table 5.1 a comparison of the costs required for the calculation of the gradient and the Hessian for each of the minimization algorithms in a single iteration. The factors multiplied by N are the operations that have to be evaluated for each point in each view while the others are only required once per view, e.g. Rodrigues formula in Equation (5.8) to compute the rotations. Each of the shown numbers in the table designates the equivalent amount of flops needed while the operations that require more than one flop, e.g. cosine and sine, are mentioned by there names since they depend on the machine used. LM requires the least computational effort to compute these terms while the Newton algorithm in the vector space requires the most, i.e. 5 times the computational resources. This shows why the evaluation of the full Hessian shown in Equation (5.12) is usually avoided in such problems. However, the amount of computations required by the two versions of the projective Newton algorithm on the manifold is 60% less on average than the latter. Compared to LM, the complexity induced by the proposed method is only twice higher. However, its application will lead to an improvement in the accuracy of the motion estimates and will make the result more robust against the noise. This can be seen from the fact that taking the full Hessian notation, i.e. the second order derivatives in Equation (5.12), makes the algorithm more robust against being stuck in local minima. This point will be visualized in the results.

Table 5.1: The total number of flops needed for the computation of the gradient and the Hessian per iteration. N is the number of points, sqrt is the scalar square root, cos is the scalar cosine and sin is the scalar sine. The terms exp and $\exp_4$ are the 3×3 and 4×4 matrix exponential respectively.

Algorithm	Number of flops
LM	$N(114 + 2 \cdot \text{sqrt}) \;+\; (227 + \exp + \text{sqrt} + \cos + \sin)$
Newton Vectorspace	$N(564 + 2 \cdot \text{sqrt}) \;+\; (1024 + \exp + \text{sqrt} + \cos + \sin)$
Newton SE_3	$N(241 + 2 \cdot \text{sqrt}) \;+\; (15 + \exp_4)$
Newton $\mathbb{R}^3 \times SO_3$	$N(235 + 2 \cdot \text{sqrt}) \;+\; (6 + \exp)$

5.4.6 Application to Structure and Motion

Although it is out of scope of this thesis, but the proposed scheme can be extended for simultaneous structure and motion optimization in BA. We will briefly sketch here how this issue can be fulfilled. Let us recall first that each point $\mathbf{P}_j$ of the 3D structure can be parameterized using its coordinates in 3D space, i.e. (X_j, Y_j, Z_j), see [20]. The cost function shown in Equation (5.19) should be reformulated to designate the mapping

$$\begin{aligned} E \circ \phi_{(\mathbf{M_1},\cdots,\mathbf{M_L},\mathbf{P}_1,\cdots,\mathbf{P}_N)}(\mathbf{z}) \quad &: \quad \mathbb{R}^{6L+3N} \longrightarrow \mathbb{R} \\ &:= \sum_{i=1}^{L}\sum_{j=1}^{N} \mathrm{d}\left(\mathbf{p}_j^i, \pi\mathbf{\Lambda}_\phi^i \mathbf{M}_i \mathbf{P}_j\right)^2, \end{aligned} \tag{5.28}$$

where the vector $\mathbf{z} \in \mathbb{R}^{6M+3N}$ holds the parameters of all the camera matrices and the 3D points, i.e. $\mathbf{z} = (\mathbf{a_1^T}, \cdots, \mathbf{a_L^T}, X_1, Y_1, Z_1, \cdots, X_N, Y_N, Z_N)$, while $\mathbf{\Lambda}_\phi^i$ is the corresponding multiplicand of the motion that depends on the parameterization used, see Propositions 5.1 and 5.2. The derivations of the gradient $\nabla\left(E \circ \phi_{(\mathbf{M_1},\cdots,\mathbf{M_L},\mathbf{P}_1,\cdots,\mathbf{P}_N)}\right)(0) \in \mathbb{R}^{\mathrm{6M+3N}}$ and the Hessian $\mathbf{H}_{E\circ\phi_{(\mathbf{M_1},\cdots,\mathbf{M_L}\mathbf{P}_1,\cdots,\mathbf{P}_N)}}(0) \in \mathbb{R}^{\mathrm{6M+3N\times6M+3N}}$ of the cost function at zero are then computed following a similar concept to Equations (5.20) and (5.21). These are used after that to calculate the solution vector $\mathbf{z}$ using the following update scheme

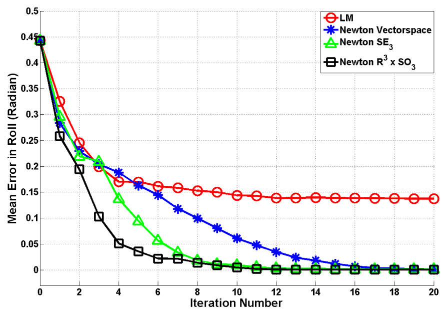
$$-\left(\mu \cdot \mathbf{I} + \mathbf{H}_{E\circ\phi_{(\mathbf{M_1},\cdots,\mathbf{M_L},\mathbf{P}_1,\cdots,\mathbf{P}_N)}}(0)\right) \cdot \mathbf{z} = \nabla\left(E \circ \phi_{(\mathbf{M_1},\cdots,\mathbf{M_L},\mathbf{P}_1,\cdots,\mathbf{P}_N)}\right)(0). \tag{5.29}$$

Of course, the Hessian matrix of the system is sparse in nature due to the formulation of BA. Therefore, the zero patterns of this entity can be exploited in the inversion operation to obtain the non-linear least-squares solution of $\mathbf{z}$. For more information about this topic, the interested reader may refer to [20, 197] to see how this issue is usually performed.

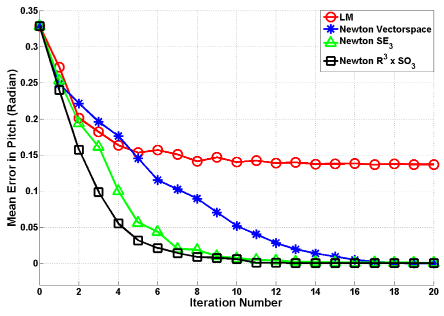
5.5 Results and Discussion

A comparison will be described in this section regarding the application of LM, the Newton algorithm in the vector space and the proposed Newton algorithm in its both versions to refine the motion estimate of a moving camera in the BA framework. To differentiate between the different Newton schemes, the Newton vectorspace denotes the algorithm described in Section 5.3.3, the Newton SE_3 designates the projective Newton algorithm with the parameterization of SE_3, the Newton $\mathbb{R}^3 \times SO_3$ denominates the other projective Newton scheme with the $\mathbb{R}^3 \times SO_3$ parameterization and LM is the Levenberg-Marquardt algorithm. All of the techniques will be tested using one simulated data set and three real image sequences. The goal of the tests is to demonstrate the performance of the algorithms in terms of the convergence and the accuracy of the estimated motion. For every algorithm, it is assumed that the cameras are intrinsically calibrated and that the 3D structure is already computed along with the corresponding image projections. It is important to point out that the four methods are implemented in Matlab and they possess the same structure. The only dissimilarity among them resides in the way the gradients and the Hessians are evaluated which was thoroughly discussed in the previous sections and illustrated in Table 5.1. Hence, the divergence in the execution is only dependent on this difference.

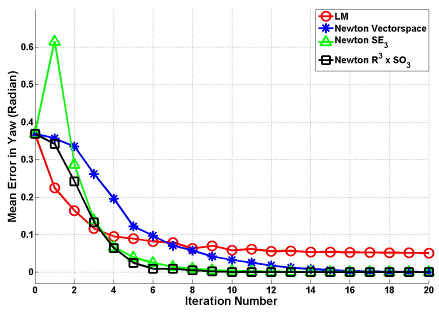
Figure 5.4 shows the output of the algorithms when applied to a simulated data set. The set was generated by creating 100 random 3D points and projecting them on four random 256×256 images. The feature points and the ground truth 3D structure present the inputs to the Newton algorithms while the initial estimate for the motion was randomly chosen. The set is tested in the noise free case. The error in the rotation is measured by transforming the rotation matrix to the Euler angles, i.e. roll, pitch and yaw, and then calculating the absolute difference between the estimated and the ground truth values. For the translation, the error is computed by evaluating the distance between the estimated vectors and the corresponding true values. The results illustrated in the plots show the average over 1000 trials. They demonstrate in both cases that taking the total Hessian and not just the approximation leads to more accuracy in the estimated motion.



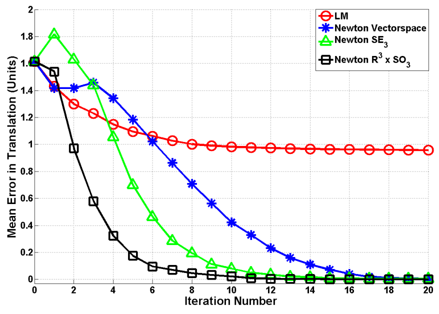
(a)



(b)



(c)



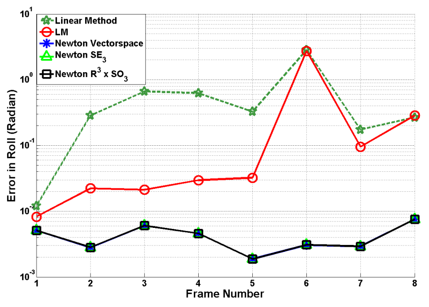
(d)

Figure 5.4: Result of the algorithms on the simulated data sets. From (a) to (d) in the respective order: Error in the roll, pitch, yaw and translation estimates versus the number of iterations to converge in the noise free set.

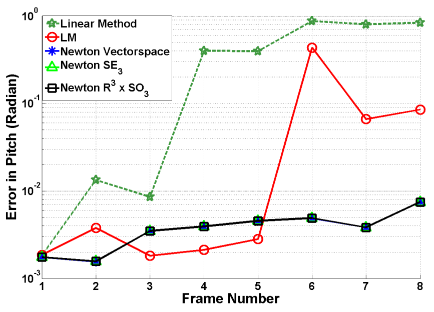
The second test will be conducted using a subset of the hall stereo sequence, that will be applied in Chapter 6, that consists of nine stereo images of size 640×480. The intrinsic and extrinsic parameters of the stereo rig were computed using the technique of [74]. The motion of the rig consists of both rotational and translational movements which are known. Using some features tracked over all the nine stereo images, we computed nine partial 3D structures using each stereo pair by calculating the 3D points with the camera matrices of the stereo rig. In order to improve the accuracy of the estimate of the scene, we determined the epipolar geometry between each pair using the robust version of the five-point algorithm of [162] that was proposed in [26]. The epipolar constraint was then applied to eliminate the outliers and to increase the number of feature points by guiding the correspondences. The absolute orientation technique of [164] was then used to compute the linear relative pose estimates of the motion by calculating a rigid transformation between the sequential partial structures and, hence, resulting in eight initial estimates for the motion. We used these estimates to obtain an initial alignment of the nine partial structures into a single one using Equation (5.2). The structure was then inputted along with the initial estimates of the motion to refine its alignment with the minimization schemes. The result of this experiment is depicted in Figure 5.5. We show here the average error in the estimated motion for each frame as was done in the simulated case but using the ground truth values of the motion and the average mean reprojection error versus the number of iterations. As a reference, we also provided the output of the linear algorithm of [164]. The first thing one can notice is that the all the schemes are able to reduce the mean reprojection error to the order of 10^{-6} to the exception of LM. One can also realize that the average error in the motion variables is almost constant among all the frames of the sequence and is much lower than the output of LM. This result motivates for the employment of the manifold based Newton algorithm in TPTA scenarios since the mean reprojection error does not propagate as with LM and in contrast to the Newton scheme in the vector space it has a rather similar complexity.

Since the problem in hand is the camera pose estimation, it is also possible to test the algorithms on video sequences originating from a single moving camera. Therefore, the last two data sets we will use are the corridor and the dinosaur sequences of the visual geometry group (VGG) where each one consists of 11 and 36 uncalibrated frames, respectively[2]. The initial estimates of the motion in these sets cannot be obtained as was suggested in this chapter. To obtain these entities, we matched and tracked some feature points over each sequence. We estimated after that the intrinsic parameters of the images by applying the camera self-calibration technique of [42]. Using the algorithm of [26], we computed the robust estimates of the essential matrices between the sequential images, rejected the feature matches that do not satisfy the epipolar constraint and then computed more correspondences by guiding the matches. The initial estimates of the motion are computed by factorizing the essential matrices, triangulating some of the match points and checking which of the possible camera matrices satisfy the chieralities [20]. The 3D structure was then obtained by computing the 3D points with the initial estimates of the camera matrices. Using the different minimization schemes, we

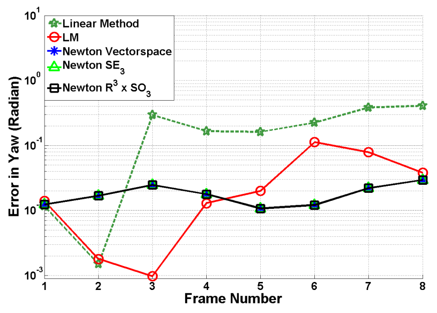
[2]The sequences can be downloaded from the website of the VGG group at the University of Oxford: www.robots.ox.ac.uk/~vgg/data/data-mview.html.



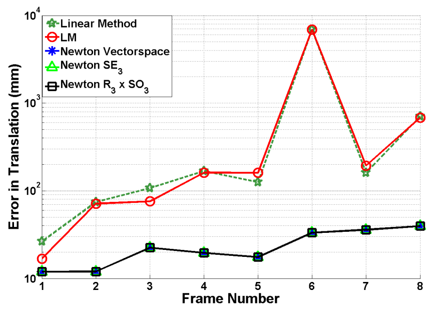
(a)



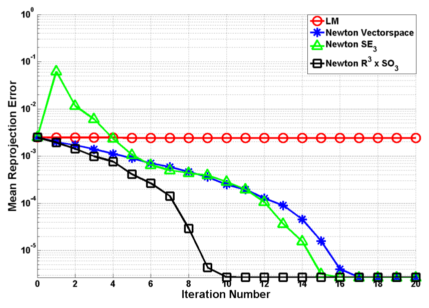
(b)



(c)



(d)



(e)

Figure 5.5: Result of the algorithms on the Hall stereo sequence using close initial estimates of the motion. a to d: Error in the estimates of the roll, pitch, yaw and translation respectively. e: Mean reprojection error versus the number of iterations averaged over all the sequence.

refined the motion estimates and the results are shown in Figures 5.6 and 5.7 respectively. In both cases, the LM algorithm has failed in recovering most of the poses of the camera while the others have lead to a good output. The correctness of the motion estimates of the corridor sequence cannot be numerically calculated because this set does not possess a ground truth value of the motion. To the exception of LM, however, the obtained shapes of the motion look pretty the same as the one obtained in [198]. With the Dinosaur

Table 5.2: The results of the minimization algorithms on the dinosaur sequence along with the ground truth values of the motion. The mean and standard deviation of the overall camera poses are in degrees. The outliers are the camera positions that are not lying on the circle.

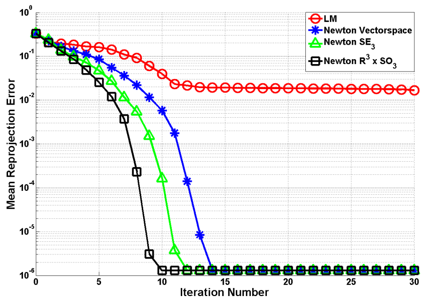
Algorithm	**Mean**	**Std**	**Number of Outliers**
Ground Truth	10	0.05	0
LM	1.11	58.3	20
Newton Vectorspace	10	0.360	0
Newton SE_3	10	0.389	0
Newton $\mathbb{R}^3 \times SO_3$	10.32	16.71	2

sequence, the Newton method with the SE_3 parameterization was able to capture the whole circular motion of the camera. This is also the case with the Newton algorithm in the vector space. The Newton technique with the $\mathbb{R}^3 \times SO_3$ parameterization was also able to capture most of the camera poses to the exception of two. The ground truth set of this camera was taken in steps of 10° rotation with a standard deviation of 0.05 [199]. The values obtained with each of the algorithms are depicted in Table 5.2 along with the number of outliers which reflect the positions that are not lying on the circle. This table also confirms that the estimated motion is accurate keeping in mind that the structure estimate is noisy. The latter was not refined to emphasize on the robustness of the algorithms in estimating the motion.

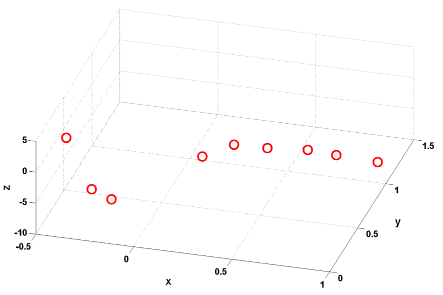
By comparing both parameterization schemes of the projective Newton-type algorithm, we can notice that their performance was almost the same in all simulations except with the dinosaur sequence. There, the technique with the parameterization on $\mathbb{R}^3 \times SO_3$ diverged twice. The reason for that might be incurred from the fact that this parameterization treats each entity, i.e. rotation matrix and translation vector, independently which was also seen in the form of the obtained gradient and Hessian in Equations (5.24) and (5.25). In contrast, the parameterization on SE_3 treats all the variables jointly which might has lead to this slight improvement in the output.

5.6 Summary

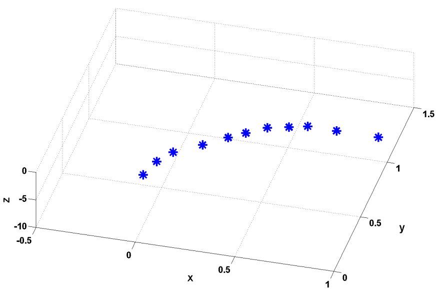
This chapter derived a projective Newton-type minimization scheme that refines the motion of a moving camera by optimizing on a Riemannian manifold. Two parameterizations were proposed that benefits from the underlying Lie group structure of the rigid motion. The cost function of BA was modified to accommodate for the parameterizations and then the full numerical formulas of the gradients and the Hessians to be used in the process were elicited. The theoretical proof of the convergence of the algorithm was inferred using the global convergence theorem. When compared to a similar Newton-based technique in the vector space, the proposed algorithm has a simpler evaluation of the gradient and the Hessian since the derivations on a Riemannian manifold are more compact. Compared to LM, the method has a comparable complexity although the complete form of the Hessian is used and not just its approximation as done in the previous scheme. Tested on simulated and real image sequences, the proposed approach leads to more accurate results and it is more robust against the noise. The derived method seems to be promising for the camera pose estimation problem arising generally in computer vision applications and specifically in TPTA scenarios.



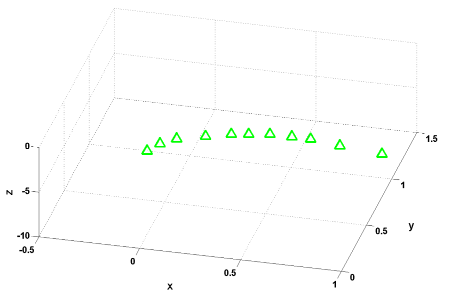
(a)



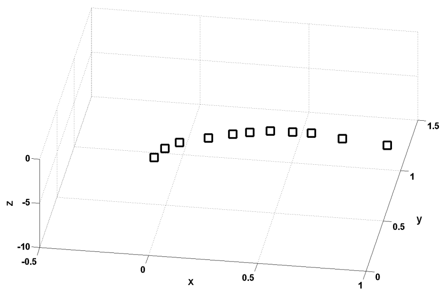
(b)



(c)



(d)



(e)

Figure 5.6: Application of the algorithms to the Corridor sequence. (a): Mean reprojection error versus the number of iterations averaged over all the sequence. (b): Output of LM. (c): Output of Newton vectorspace. (d): Output of Newton SE_3. (e): Output of Newton $\mathbb{R}^3 \times SO_3$.

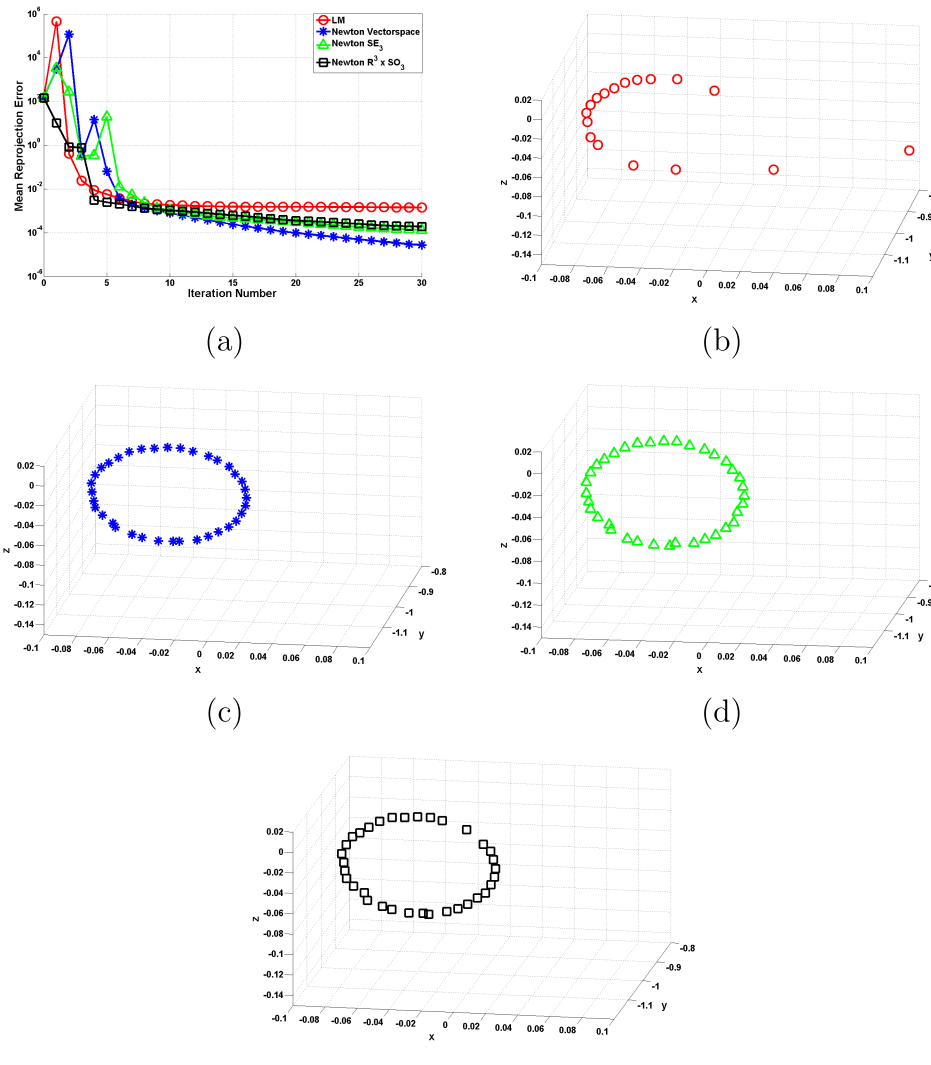


Figure 5.7: Application of the algorithms to the Dinosaur sequence. (a): Mean reprojection error versus the number of iterations averaged over all the sequence. (b): Output of LM. (c): Output of Newton vectorspace. (d): Output of Newton SE_3. (e): Output of Newton $\mathbb{R}^3 \times SO_3$.

Chapter 6

Scene Reconstruction in Telepresence

After deriving the key components of this thesis, we would like to emphasize on the performance of the schemes in a TPTA scenario. The work of this thesis is related to the subproject M3 which is a part of the SFB 453 telepresence and teleaction project. The subproject, along with several other ones, had to be integrated in a demonstrator for the project appraisal. The one that we were involved in was the "Multi-modal Multi-User Telepresence and Teleaction system". Its objective can be summarized as two human operators have to explore and navigate in a remote environment in order to locate and repair a broken pipe. To what concerns this thesis, the task was to construct the 3D model of a room so that the teleoperators are able to navigate through the door to another room where the broken pipe is located. Each teleoperator was equipped with a stereo head, see Figure 6.1. On the top of the head of one of the teleoperators, a time of flight camera (PMD) was also integrated[1]. The latter was registered with the stereo cameras so that it is possible to compare the output of the stereo-based system to that of the PMD.

The VGA stereo images of the teleoperator are captured, compressed using the MPEG4 scheme and then sent on the network. At the operator site, the images are decoded and then rectified to align the epipolar lines of both stereo images with the scanlines using [119]. We used the BSP-Tritree algorithm of Chapter 3 to construct the reduced mesh of the image and then computed the depth of the mesh nodes using the sparse stereo matching strategy that was described in Chapter 4. The maximum disparity range was set to 150 pixels. The depth information were then employed to construct

[1]www.pmdtec.com

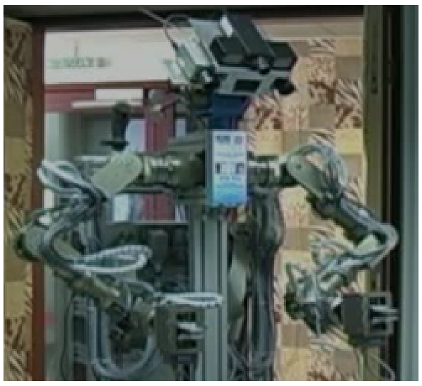


Figure 6.1: The remote teleoperator where our stereo camera was integrated. On the top of the stereo head, we placed a PMD 3D camera (in black) which we used to judge the quality of the acquired scenes with our stereo system.

the 2.5D mesh of the frame. The motion trajectory of the teleoperator's stereo head was known and it was used to register the sequential meshes. The generated mesh was encoded and broadcasted over the network using the MPEG4 BIFS standard [21]. At the operator site, the received mesh was rendered using the image sphere techniques of [200]. The throughput (fps) of the 3D reconstruction scheme is illustrated in Figure 6.2. The rate was measured from the instant a stereo image is received till the mesh update with MPEG4 BIFS is generated. The average frame rate we achieved varies between 1 fps with the sparse DP version of [147] and 1.5 fps with the sparse scheme of [16] by employing 4 CPUs. When compared to the normalized correlation method, the latency that one can notice is around 1.5 fps at the cost of reconstructing more confident virtual scene for the teleoperator. The visual results of the scan illustrated in Figure 6.3 show that the quality of the reconstructed scenes using the sparse DP are preserved. The constructed 3D scene of the scan is pretty overlapping with that of the PMD camera. This result also justifies the application of sparse stereo in such scenarios. The latter will not deteriorate the visual quality of the reconstructed scenes and will reduce the latency of the DP-based stereo reconstruction at the teleoperator side. Furthermore, it is possible now to attain a real-time rate by employing a computer cluster with a fewer number of processing units and, thus, reducing the total cost.

The motion of the stereo head in the above demonstrator was preset and, therefore, it was not necessary to employ the motion estimation scheme that was developed in Chapter 5. Going back to the result's section of that chapter, however, we showed in Figure 5.5 the error in the motion estimate using a subset of the stereo sequence that was captured from the demonstrator when the proposed algorithm is applied. The error in the proposed New-

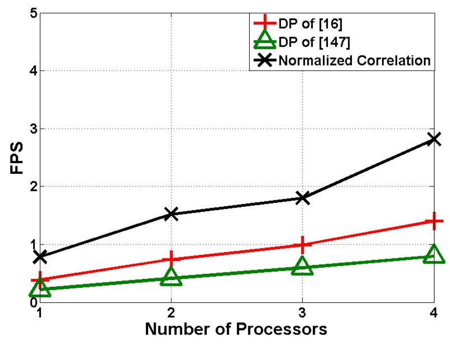


Figure 6.2: Evaluation of the overall throughput in fps of the DP based sparse scene acquisition system. For reference, we also tested the normalized correlation approach. The maximum disparity range was set to 150 pixels.

ton scheme was better and almost constant when compared to Levenberg-Marquardt which is usually used in such problems. In order to compute the motion in a TPTA scenario or in any computer vision problem, feature points need to be detected and tracked over the sequence. A fast tracker that can accomplish this job is the KLT tracker what was developed by Tomasi and Kanade in [201] and then enhanced by Bouguet in [202]. Although it is fast, KLT fails to track the points when the movement between the consecutive frames is large. This explains why it was not possible to apply the motion estimation scheme to all of this stereo sequence. What is needed to overcome this problem is to employ a wide baseline image matching technique. How to compute the image matches for wide baseline cameras is a wide research field by itself. Although there have been a lot of work is this area as the methods proposed in [203, 204, 205, 206], further research is necessary to develop suitable algorithms for TPTA scenarios.

In the presented demonstrator, there was no plan to use stereo cameras with automatic zoom lenses at that time. Integrating the MLS algorithm to compute the camera parameters, however, is not difficult to realize. Looking at Figure 1.2, the operator has the ability to control the automatic zoom using the control lines. The zoom control is also required for the MLS algorithm in order to know at which values the camera parameters need to be computed. Once the parameters are interpolated, they can be fed for the stereo rectification and the 3D reconstruction algorithms. It is worth to note that while varying the zoom, the zooming factor has to be kept the same in both cameras of the teleoperator. This is necessary to ensure that the stereo matching strategy remains valid. Otherwise, it will be necessary to follow other techniques that are capable of matching images with different resolutions. Examples of such methods are proposed in [207, 208]. The rectification theme actually opens the door for a future research topic. It is important to see how the errors in the interpolated intrinsic parameters affect the stereo rectification process and how can MLS be tuned to minimize the effect.

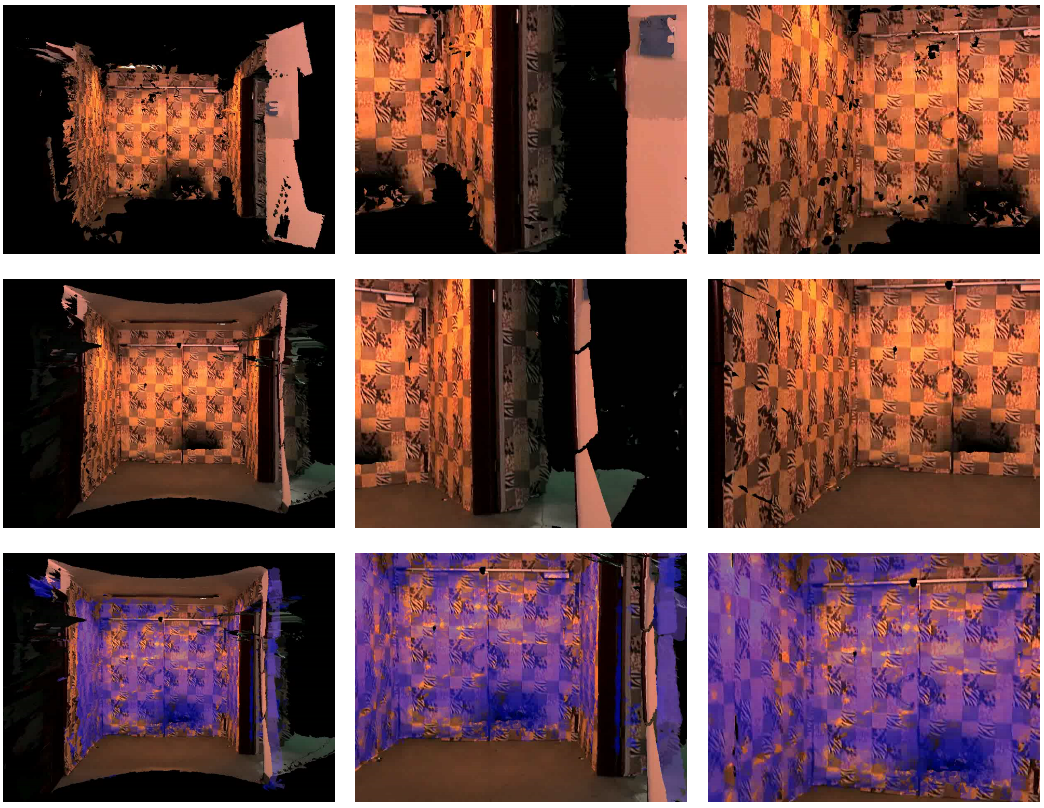


Figure 6.3: Scan of a room-environment. The first row shows three views of the output of the proposed system using for stereo matching the sparse DP version of [16]. The second row shows the same three views using the PMD 3D camera. The third row shows the output of the proposed system (shaded) overlapped with that of the PMD camera.

Chapter 7

Conclusions and Outlook

We presented in this thesis several techniques to enhance the stereo-based 3D scene reconstruction. Our motivation is to make this process more suitable for a TPTA scenario where several issues have to be considered. The reconstruction process should not impose any restrictions on the TPTA application. It should also result in an accurate perception of the scene while minimizing the delays incurred from the computations. In the following, we will recall the major contributions that were performed:

1. **Calibration of automatic zoom cameras with MLS:** A technique called MLS was derived in Chapter 2 based on the data fitting theory to calibrate a zoom camera. The method requires the system to be calibrated at a number of focus and zoom settings to generate some measurement points of the intrinsic parameters. Then, MLS proceeds by approximating each intrinsic parameter with local polynomial functions at the unmeasured points. Compared to previous fitting methods, MLS was able to increase the accuracy of the variables. In addition, it was less sensitive to the number of measured focus and zoom settings. The reason for these improvements is the gained adaptivity in fitting the scattered data since the final solution is the concatenation of several local solutions. To reduce its computational complexity, the CMLS technique was developed. The latter clusters and approximates the MLS curves with several polynomial functions. Compared to MLS, CMLS requires very simple computation in a TPTA scenario; however, the latter results in more accurate values of the intrinsic parameters. In contrast to standard camera self-calibration schemes, the MLS approach can be applied in TPTA without imposing any constraints on the teleoperator. It does not require waiting for several images to calibrate the camera and does not impose any critical motions.

2. **Mesh representation of images with BSP:** In Chapter 3, a new technique based on binary space partitions was presented to approximate an image with a content adaptive mesh in which the nodes are its non-uniform samples. The goal was to represent the intensity variation of the pixels located inside a triangle of the mesh using the equation of the plane defined by its three vertices. A cost function using PSNR was then defined and maximized to locate the triangles that best describe the points lying within. BSP starts by dividing an image into two triangles along one of the diagonals. In order to subdivide each triangle recursively, three partitioning schemes were applied. A simple one that divides each triangle into two equal ones if the PSNR threshold is not satisfied. The other two make use of SVD and Kmeans to estimate the direction of the cut. The results showed that the size of the meshes constructed with BSP is smaller than the ones obtained by the state of the art methods. Furthermore, the quality of the reconstructed images was preserved. In terms of speed, the BSP method using the simple clustering scheme, i.e. BSP-Tritree, was found suitable for TPTA scenarios since it can be easily expanded to real-time due to the parallelism incurred from the application of BSP.

3. **Stereo matching with sparse DP:** One of the new ideas explored in thesis and Chapter 4 was to benefit from the sparse structure of the images obtained from an image content adaptive meshing scheme to accelerate stereo matching. For this reason, a sparse stereo matching algorithm based on dynamic programming was developed that computes the disparity values only at the nodes of the mesh (non-uniform samples) of the reference image. The method starts by computing the matching costs just at the samples. A smoothness cost was then proposed to take the distance between the neighboring samples into account. This was required to let DP increase the smoothness penalty between the close samples since they are located near the edges of the image and vice versa. The conducted tests showed that the non-uniform samples are sufficient to recover the dense disparity map of the scene with an acceptable error. In addition, they demonstrated that depth maps are now obtained up to 50% faster than when using dense DP stereo algorithms. This makes the approach applicable in TPTA scenarios because it requires less computational resources and, hence, less latencies. In addition, the application of DP increases the fidelity of the computed depth maps. This can be visualized since it introduces lower error rate than the local approaches that have been used in recent stereo-based telepresence systems.

4. **Camera motion estimation via Newton-type minimization on the manifold:** To estimate the global motion of the camera, i.e. teleoperator, a projective Newton-type scheme was derived in Chapter 5. The algorithm refines the motion in the bundle adjustment (BA) framework (reprojection errors) via optimization on the Riemannian manifold of the rigid motion. Two parameterizations were proposed for the motion variables. The first one uses the underlying Lie group structure of the special Euclidean group SE_3 while the second one benefits from the homeomorphism between the latter and $\mathbb{R}^3 \times SO_3$. The cost function of BA was modified accordingly to accommodate each of the parameterization schemes. Then, the complete formulas of the gradients and the Hessians to be used in the optimization were calculated. Compared to a similar Newton-based technique in the vector space, the proposed methods is simpler since the evaluation of the gradient and the Hessian are much cheaper on the manifold. Compared to Levenberg-Marquardt, the algorithm has a comparable complexity in terms of the evaluation of the gradient and Hessian. In addition, its application leads to more accurate results and it is more robust against the noise. The obtained results suggest the usage of such methods in TPTA since they do not significantly increase the complexity and they lead to more accurate estimates of the motion.

In the course of the thesis, we also found several issues that can extend this work and present interesting directions for future research:

- The MLS technique proposed in Chapter 2 approximates independently each intrinsic parameter. This motivates the exploration of techniques that can jointly interpolate the five intrinsic parameters by considering them along with the focus and zoom inputs as a seven dimensional space. An interesting idea is to investigate a suitable parametric representation of the manifold structure of this space. The representation must allow us to have a two way mapping between the parameters and the manifold structure and not just a one way mapping as in manifold learning scheme, see [56, 57, 58, 59, 60, 61]. Such a feature will provide the possibility to embed the seven dimensional space into a lower dimensional one where the data can be approximated in an easier manner. By back projecting the computed model into the higher space, it will be possible to recover the real values of the intrinsic parameters required for camera calibration. A similar idea was performed in [62] were the authors used some deformation models to accommodate for such mapping in image interpolation.

- Sparse DP was derived without taking into consideration if a non-uniform sample is occluded or not. It would be a more accurate procedure to find a suitable occlusion model in sparse DP to make it detect the occluded non-uniform samples in the other image. This knowledge should be later incorporated to modify the position of the sample in the mesh to the location of the nearest non-occluded pixel in the image that retains the quality of its mesh approximation.

- The sparse stereo matching strategy was applied in this thesis to the dynamic programming which is a scanline-based optimization scheme. The global approaches, however, usually result in disparity maps with better quality but they are slow to be applied in real-time applications as in TPTA. It would be interesting to apply this strategy to the global methods. It is necessary to analyze the effect of the sparsity of the image on the quality of their result and their speed to assess if it is feasible to apply them in TPTA. Some preliminary results on this topic have been recently obtained in [31].

- The implemented stereo-based 3D reconstruction scheme considers only static scenes. If moving objects are to occur in a TPTA scenario, it is more convenient to update only their location and leave the static ones intact. One methodology would be by first processing the stereo images to detect the moving objects. For this task, several algorithms should be tested and compared to measure their efficiency in TPTA as the ones proposed in [209,210,211]. Once detected, the moving objects are subtracted from the scene and their depth should be updated by tracking their movement.

- It would be interesting to extend the developed camera motion estimation technique on the manifold to adjust structure and motion simultaneously as was sketched in Section 5.4.6. It is important to see how a projective Newton-type minimization scheme on the manifold would affect the overall optimization in bundle adjustment. It is significant to visualize whether or not the relative performance of the algorithm remains the same in such cases.

- As noticed in the previous chapter, it is necessary to investigate a wide baseline matching strategy suitable for TPTA scenarios in order to track some points and use them to estimate the motion of the teleoperator. In addition, it is important to study the effect of the proposed calibration scheme on the stereo rectification process and see how it can be tuned to minimize the induced errors.

Appendix A

Abbreviations, Notation and Symbols

A.1 Abbreviations

2D	two dimensional or dimensions
3D	three dimensional or dimensions
BA	bundle adjustment
BIFS	binary format for scene
BSP	binary space partitions
CMLS	clustered moving least-squares
DOF	degrees of freedom
DP	dynamic programming
FLOPS	floating point operations
FPS	frames per second
CPU	central processing unit
GPU	graphical processing unit
ICA	independent component analysis
ICP	iterative closest point
LLE	local linear embedding
LM	Levenberg-Marquardt
LS	least-squares
MLS	moving least-squares
MPEG	moving picture expert group
MSE	mean squared error
PCA	principal component analysis
PSNR	peak signal to noise ratio
RMSE	root mean squared error
SSR	sum of the squared residuals

SST	total sum of squares
SVD	singular value decomposition
TPTA	telepresence and teleaction
UIPE	undistorted image plane error
VGA	video graphics array, denotes an image of 640×480 pixels

A.2 Mathematical Notation

$a, \mathbf{a}, \mathbf{A}$	scalar, vector, matrix
$(\cdot)^T$	transpose operation of the argument
$\lfloor\cdot\rfloor$	floor operation of the argument
$\lVert\cdot\rVert$	the 2 norm of the argument
$\lvert\cdot\rvert$	absolute value of the argument
$\leftarrow$	right hand side into left hand side
$\rightarrow$	left hand side into right hand side
$:$	left hand side is defined as the right hand side
$:=$	left hand side is defined to be equal to the right hand side
$\triangle$	area of the triangle
∞	infinity symbol
∇	gradient operator
$\in$	left hand side belongs to right hand side
$\circ$	composition of a function and a parameterization
$\mathrm{d}/\mathrm{d}a$	first order derivative with respect to a
$\mathrm{d}^2/\mathrm{d}a^2$	second order derivative with respect to a
$\partial/\partial a$	first order partial derivative with respect to a
$\partial^2/\partial a^2$	second order partial derivative with respect to a
lim	the limit operator
max	the maximum operator
min	the minimum operator
cos	cosine of the argument
exp	matrix exponential of the argument
sin	sine of the argument
sqrt	square root of the argument
$\mathbf{se}_3$	Lie algebra of the special Euclidean group
$\mathbf{so}_3$	Lie algebra of the special orthogonal group
$\mathbb{R}$	set of real numbers
$\mathbb{R}^i$	vector of real numbers with a length of i
$\mathbb{R}^{i\times j}$	matrix of real numbers with a dimension of $i \times j$
SE_3	special Euclidean group of 3D rigid transformations
SO_3	special orthogonal group of 3D rotation matrices

A.3 Mathematical Symbols

i, j, k, l	scalars or indices
β_o	coefficient of the line opposite to $\boldsymbol{v}_o$
β_p	coefficient of the partition line in a triangle subdivision
χ	measurements vector of focus and zoom settings
χ_i	also χ_j, single measurement of a focus and zoom setting
δ_i	where $i = \{1, 2, 3\}$, are the barycentric coordinates of $\boldsymbol{p}_I$ in T
$\vec{\ell}$	direction of the partitioning line in a triangle
ϵ	PSNR threshold for the mesh approximation
$\vec{\eta}$	normal vector to the plane defined by the triangle T
γ	vector coefficients of ϖ for any intrinsic parameter
γ_{ij}	single coefficient of ϖ for any intrinsic parameter
γ_S	vector coefficients of a cluster S
ι	monotonic decreasing function for E_{smooth}
κ	radial distortion coefficient
λ	the weighting for E_{smooth}
μ	damping term for Newton algorithms
ω_i	where $i = \{1, 2, 3\}$, are the DOFs of a 3D rotation matrix
ϕ	parameterization of the motion on the manifold
π	mapping from $\mathbb{R}^4$ to $\mathbb{R}^3$
υ	mean value of the intensities of the pixels in a mask
ϖ	bivariate polynomial function
ϑ	weighting function
ϑ_{SR}	weighting function defined over SR, needed for C_a
ψ	basis vector of the polynomial function ϖ
ρ	function that results in the value of intrinsic parameter at each measured focus and zoom setting s_i and o_i
σ	skew parameter describing the angle of pixel's axis
τ	tangential distortion coefficient
θ	vector of intrinsic parameters of measured focus and zoom settings
θ_i	value of an intrinsic parameter at each measured focus and zoom setting
$\hat{\theta}_j$	estimated value of an intrinsic parameter at a focus and zoom setting
$\hat{\theta}$	vector of estimated values of an intrinsic parameter at several focus and zoom settings
$\bar{\theta}_j$	mean of the intrinsic parameters in a cluster
ξ	number of the coefficients in ϖ

Δ	a configuration of all disparity values in an image
$\mathbf{\Gamma}$	matrix of the coefficients for all intrinsic parameters
$\mathbf{\Lambda}_\phi$	multiplicand of the motion in ϕ, i.e. $\phi_{(\mathbf{M})}(\mathbf{a}) = \mathbf{\Lambda}_\phi \mathbf{M}$
$\mathbf{\Omega}$	isomorphism from $\mathbb{R}^3$ to $\mathbf{so}_3$, it denotes a skew-symmetric matrix of the argument
$\mathbf{\Pi}^2_m$	space of polynomials of degree m
$\mathbf{\Psi}$	matrix of basis vectors of the polynomial function ϖ
$\mathbf{\Theta}$	matrix of the values of all intrinsic parameters of at each measured focus and zoom setting s_i and o_i
$\mathbf{\Upsilon}$	a measurement matrix containing the points $\boldsymbol{p}_I$ of a triangle
Ξ	magnitude of the second derivative of a pixel
$\mathbf{a}$	also $\mathbf{a}_i$, 6×1 vector of the motion parameters for the i^{th} camera
c	skew parameter describing the angle of a pixel's axis
d	disparity function of a pixel
d	distance function between the arguments
$\boldsymbol{e}_L$	the epipole in the left image of a stereo camera
$\boldsymbol{e}_R$	the epipole in the right image of a stereo camera
f	focal length
f_x	x component of the focal length
f_y	y component of the focal length
$\mathbf{g}$	also $\mathbf{g}_i$, reformulation of E for the i^{th} camera, i.e. $E_i = \mathbf{g}_i^T \mathbf{g}_i$, in BA
h	height of a window or mask in the image
m	order or degree of the polynomial ϖ
$\mathbf{m}$	4×4 matrix representing the Lie algebra $\mathbf{se}_3$
n	size of a subset of data points
o	also o_i and o_j, single zoom setting of the automatic zoom
$\boldsymbol{p}$	a point in an image
$\boldsymbol{p}_L$	also $\boldsymbol{p}_L(x, y)$, point in the left image of a stereo camera
$\boldsymbol{p}_R$	also $\boldsymbol{p}_R(x', y')$, point in the right image of a stereo camera
$\boldsymbol{p}_I$	also $\boldsymbol{p}_{Ij}$, a point in an image with the intensity as the third coordinate
$\boldsymbol{p}_m$	mid-point between the centers of the two clusters in a triangle
$\mathbf{p}$	a point in an image with homogeneous coordinates
$\mathbf{p}^i_j$	also $\mathbf{p}_j$, homogeneous coordinates of the j^{th} measured 2D point in the i^{th} camera
$\hat{\mathbf{p}}^i_j$	also $\hat{\mathbf{p}}_j$, homogeneous coordinates of the j^{th} reprojected 2D point in the i^{th} camera
r	$= \sqrt{{x_n}^2 + {y_n}^2}$. Needed for computation of x_d and y_d
s	also s_i and s_j, single focus setting of the automatic zoom

sk	skewness of the data (statistical term)
$\mathbf{t}$	3×1 vector denoting the 3D translation
$\mathbf{t}_i$	3D translation vector of the camera at instinct i
$\boldsymbol{v}_i$	vertex of a triangle in the mesh approximation of an image
$\boldsymbol{v}_n$	new vertex in the mesh approximation of an image
$\boldsymbol{v}_o$	opposite vertex to $\boldsymbol{v}_n$
w	width of a window or mask in the image
x	also x_i and x_j, column index in the image
$\hat{x}$	also $\hat{x}_i$ and $\hat{x}_j$, reprojected column index in the image
x_0	x coordinate of the principal point
x_d	distorted x coordinate of the pixel
x_n	normalized x coordinate of the pixel
x_{nb}	x coordinate of the neighboring sample in sparse DP
y	also y_i and y_j, row index in the image
$\hat{y}$	also $\hat{y}_i$ and $\hat{y}_j$, reprojected row index in the image
y_0	y coordinate of the principal point
y_d	distorted y coordinate of the pixel
y_n	normalized y coordinate of the pixel
B	baseline of a stereo camera
C	data costs for stereo matching
C_a	data costs C aggregated over the support region SR
$\widetilde{\mathbf{C}}$	a matrix for all the costs, i.e. cost volume, in DP
$\mathbf{D}_1$	selection matrix, equals to $[\mathbf{I}\ \mathbf{0}]$
$\mathbf{D}_2$	selection matrix, equals to $[\mathbf{0}\ \mathbf{I}]$
E	also E_i, error or cost function to be optimized
E_{data}	data error or cost function associated with C
E_{smooth}	smoothness error term or cost function for stereo matching
$\widetilde{E}_{smooth}$	smoothness error term for sparse DP
G	number of calibration points multiplied by that of the calibration grids
$\mathbf{H}_{\mathbf{E}_i}$	Hessian matrix
$\hat{\mathbf{H}}_{\mathbf{E}_i}$	approximated Hessian matrix, $2\mathbf{J}_{\mathbf{g}_i}^T \cdot \mathbf{J}_{\mathbf{g}_i}$
$\mathbf{H}_{g_{j,i}}$	Hessian with respect to each term of $\mathbf{g}_i$
$\mathbf{H}_{E\circ\phi_{(\mathbf{M})}}(0)$	Hessian of the composition $E \circ \phi_{(\mathbf{M})}$ on the manifold
$\nabla(E\circ\phi_{(\mathbf{M})})(0)$	gradient of the composition $E \circ \phi_{(\mathbf{M})}$ on the manifold
I	intensity value of the pixel
$\hat{I}$	constructed intensity value of the pixel
$\mathbf{J}_\mathbf{g}$	also $\mathbf{J}_{\mathbf{g}_i}$, the Jacobian matrix

$\mathbf{K}$	3×3 matrix of the intrinsic parameters
$\mathbf{K}_i$	matrix of the intrinsic parameters at instinct i
L	number of poses of a camera
$\mathbf{M}_i$	4×4 matrix representing the position of the camera at instinct i
$\mathbf{M}_i^{i-1}$	relative position of the camera between instincts i and $i-1$
N	size of the data points
$O(\cdot)$	order of complexity
$\boldsymbol{O}$	camera center
$\boldsymbol{O}_L$	camera center of the left image of a stereo camera
$\boldsymbol{O}_R$	camera center of the right image of a stereo camera
$\boldsymbol{P}$	a point in space
$\mathbf{P}$	a point in space with homogeneous coordinates
$\mathbf{P}_j$	j^{th} point in space with homogeneous coordinates
$\mathbf{Q}$	a set of homogeneous coordinates of 3D points in space
$\mathbf{Q}^i$	homogeneous coordinates of the 3D points with respect to the camera position at instinct i
$\mathbf{R}$	3×3 matrix denoting the 3D rotation
$\mathbf{R}_i$	3D rotation matrix of the camera at instinct i
R^2	coefficient of determination
S	region to be clustered
S_i	also S_j, a cluster of the region S
SR	support region of a pixel over which C_a is calculated
T	a triangle in the mesh approximation of an image
$\mathbf{W}$	diagonal matrix of weights
X	horizontal coordinate of a point in space
Y	vertical coordinate of a point in space
Z	depth of a point in space

Bibliography

[1] M. Ferre, M. Buss, R. Aracil, C. Melchiorri, and C. Balaguer, Eds., *Advances in telerobotics*, Springer, 1st edition, 2007.

[2] J. Mulligan, X . Zabulis, N. Kelshikar, and K. Daniilidis, "Stereo-based environment scanning for immersive telepresence," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 304–320, Mar. 2004.

[3] I. Feldmann, S. Askar, N. Bradenburg, P. Kauff, and O. Schreer, "Real-time segmentation for advanced disparity estimation in immersive videoconference applications," in *Int. Conf. Computer Graphics, Visualization and Computer Vision*, Feb. 2002, pp. 119–126.

[4] T. Burkert, J. Leupold, and G. Passig, "A photorealistic predictive display," *Presence: Teleoperators and Virtual Environments*, vol. 13, no. 1, pp. 22–43, Feb. 2004.

[5] B. J. Lei, C. Chang, and E. A. Hendriks, "An efficient image-based telepresence system for videoconferencing," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 335–347, Mar. 2004.

[6] E. M. Izquerdo, "Stereo matching for enhanced telepresence in three-dimensional videocommunications," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 629–643, Aug. 1997.

[7] S.-M. Rhee, R. Ziegler, J. Park, M. Naef, M. Gross, and M.-H. Kim, "Low-cost telepresence for collaborative virtual environments," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 1, pp. 156–166, Jan. 2007.

[8] S.-Y. Lee, S. C. Ahn, H.-G. Kim, and M. Lim, "Real-time 3D video avatar in mixed reality: an implementation for immersive telecommunication," *Simulation & Gaming*, vol. 37, no. 4, pp. 491–506, Dec. 2006.

[9] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. V. Moere, and O. Staadt, "blue-c: a spatially immersive display and 3D video portal for telepresence," in *ACM SIGGRAPH*, Jul. 2003, pp. 819–827.

[10] S. Lang, S. Würmlin, J. Borchers, L. Hovestadt, and M. Gross, "blue-c: Using 3D video for immersive telepresence applications," in *Conf. Human Factors in Computing Systems*, Apr. 2004.

[11] S.-Y. Lee, I.-J. Kim, S. C. Ahn, H. Ko, M. Lim, and H.-G. Kim, "Real time 3D avatar for interactive mixed reality," in *ACM SIGGRAPH*, Aug. 2004, pp. 75–80.

[12] R. Yang, C. S. Kurashima, H. Towles, A. Nashel, and M. K. Zuffo, "Immersive video teleconferencing with user steerable views," *Presence: Teleoperators and Virtual Environments*, vol. 16, no. 2, pp. 188–205, Apr. 2007.

[13] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 765–776, 2005.

[14] C. Zhang and T. Chen, "A self-reconfigurable camera array," in *Eurographics Symp. Rendering*, Jun. 2004.

[15] H. Schirmacher, L. Ming, and H.-P. Seidel, "On-the-fly processing of generalized lumigraphs," in *Proc. Eurographics*, Sep. 2001, pp. 165–174.

[16] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Computer Vision*, vol. 47, no. 1, pp. 7–42, Apr. 2002.

[17] J. Mulligan, V. Isler, and K. Daniilidis, "Performance evaluation of stereo for tele-presence," in *Int. Conf. Computer Vision*, Jul. 2001, pp. 558–565.

[18] F. Kahl, B. Triggs, and K. Åström, "Critical motions for auto-calibration when some intrinsic parameters can vary," *J. Mathematical Imaging and Vision*, vol. 13, no. 2, pp. 131–146, Oct. 2000.

[19] P. Sturm, "Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length," *Image and Vision Computing*, vol. 20, no. 5–6, pp. 415–426, Mar. 2002.

[20] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2nd edition, 2004.

[21] "Coding of audio-visual objects, part 11: scene description and application engine (BIFS, XMT, MPEG-J)," ISO/IEC 14496-11.

[22] M. Sarkis, C. T. Senft, and K. Diepold, "Modeling the variation of the intrinsic parameters of an automatic zoom camera system using moving least-squares," in *IEEE Conf. Automation Science and Engineering*, Sep. 2007.

[23] M. Sarkis, C. T. Senft, and K. Diepold, “A novel technique to model the variation of the intrinsic parameters of an automatic zoom camera using adaptive delaunay meshes over moving least-squares surfaces,” in *IEEE Int. Conf. Image Processing*, Sep. 2007.

[24] M. Sarkis, C. T. Senft, and K. Diepold, “Calibrating an automatic zoom camera with moving least squares,” *IEEE Trans. Automation Science and Engineering*, (Accepted for Publication).

[25] M. Sarkis and K. Diepold, “A fast solution to the approximation of 3D scattered point data from stereo images using triangular meshes,” in *IEEE RAS Int. Conf. Humanoid Robots*, Nov. 2007.

[26] M. Sarkis, K. Diepold, and K. Hüper, “A fast and robust solution to the five-point relative pose problem using gauss-newton optimization on a manifold,” in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Apr. 2007, pp. I–681–I–684.

[27] M. Sarkis, K. Diepold, and K. Hüper, “Pose estimation of a moving humanoid using gauss-newton optimization on a manifold,” in *IEEE RAS Int. Conf. Humanoid Robots*, Nov. 2007.

[28] M. Sarkis, O. Lorscheider, and K. Diepold, “Efficient content adaptive mesh representation of an image using binary space partitions and singular value decomposition,” in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Mar. 2008, pp. 1109–1112.

[29] M. Sarkis and K. Diepold, “Content adaptive mesh representation of images using binary space partitions,” *IEEE Trans. Image Processing*, (Accepted for Publication).

[30] M. Sarkis and K. Diepold, “Towards real-time stereo using non-uniform image sampling and sparse dynamic programming,” in *Int. Symp. 3D Data Processing, Visualization and Transmission*, Jun. 2008.

[31] M. Sarkis and K. Diepold, “Sparse stereo matching using belief propagation,” in *Int. Conf. Image Processing*, Oct. 2008.

[32] M. Sarkis, K. Diepold, and A. Schwing, “Enhancing the motion estimate in bundle adjustment using projective newton-type optimization on the manifold,” in *IS&T/SPIE Symp. Electronic Imaging - Image Processing: Machine Vision Applications II*, Jan. 2009.

[33] E. Hayman, *The use of zoom within active vision*, Ph.D. thesis, Robotics Research Group, Department of Engineering Science, University of Oxford, Oxford, UK, Sep. 2000.

[34] R. L. Thompson, P. R. McAree, R. W. Daniel, and D. W. Murray, "Operator matching in visual teleoperation," *Robotic and Autonomous Systems*, vol. 50, pp. 69–80, 2005.

[35] B. J. Tordoff and D. W. Murray, "A method of reactive zoom control from uncertainty in tracking," *Computer Vision and Image Understanding*, vol. 105, no. 2, pp. 131–144, Feb. 2007.

[36] R. Kingslake, *Lens design fundamentals*, Academic Press, Jun. 1978.

[37] R. Kingslake, *Optical system design*, Academic Press, Oct. 1983.

[38] C. C. Slama, *Manual of photogrammetry*, American Society of Photogrammetry and Remote Sensing, 4th edition, Jun. 1980.

[39] D. C. Brown, "Close-range camera calibration," *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, Aug. 1971.

[40] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, Oct. 1992.

[41] M. Li and J.-M. Lavest, "Some aspects of zoom lens camera calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 11, pp. 1105–1110, Nov. 1996.

[42] M. Pollefeys, R. Koch, and L. Van Gool, "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters," in *Int. Conf. Computer Vision*, Jan. 1998, pp. 90–95.

[43] N. Canterakis, "Analytic reduction of the kruppa equations," in *Proc. 24th DAGM Symp.*, Sep. 2002, pp. 591–599.

[44] P. Sturm, "Self-calibration of a moving zoom-lens camera by pre-calibration," *Image and Vision Computing*, vol. 15, no. 8, pp. 583–589, Aug. 1997.

[45] X. Cao, J. Xiao, H. Foroosh, and M. Shah, "Self-calibration from turn-table sequences in presence of zoom and focus," *Computer Vision and Image Understanding*, vol. 102, no. 3, pp. 227–237, Jun. 2006.

[46] J. M. Lavest, G. Rives, and M. Dhome, "Modeling an object of revolution by zooming," *IEEE Trans. Robotics and Automation*, vol. 11, no. 2, pp. 267–271, Apr. 1995.

[47] J. Davis and X. Chen, "Calibrating pan-tilt cameras in wide-area surveillance networks," in *IEEE Int. Conf. Computer Vision*, Oct. 2003, pp. 144–149.

[48] S. N. Sinha and M. Pollefeys, "Pan-tilt-zoom camera calibration and high-resolution mosaic generation," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 170–183, Sep. 2006.

[49] P. Sturm, "A case against kruppa's equations for camera self-calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1199–1204, Oct. 2000.

[50] R. Koch, "3D-scene modeling from image sequences," in *ISPRS Workshop on Photogrammetric Image Analysis*, Sep. 2003.

[51] T. A. Clarke and J. G. Fryer, "The development of camera calibration methods and models," *The Photogrammetric Record*, vol. 16, no. 91, pp. 51–66, Apr. 1998.

[52] F. Remondino and C. Fraser, "Digital camera calibration methods: considerations and comparisons," in *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Sep. 2006, pp. 266–272.

[53] R. G. Willson, *Modeling and calibration of automated zoom lenses*, Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Jan. 1994.

[54] Y. S. Chen, S. W. Shih, Y. P. Hung, and C. S. Fuh, "Camera calibration with a motorized zoom lens," in *Int. Conf. Pattern Recognition*, Sep. 2000, pp. 495–498.

[55] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent component analysis*, Wiley, 1st edition, 2001.

[56] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *J. Machine Learning*, vol. 4, pp. 119–155, Dec. 2003.

[57] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *PNAS*, vol. 100, pp. 5591–5596, May. 2003.

[58] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computing*, vol. 15, pp. 1373–1396, Jun. 2003.

[59] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 113–127, Jun. 2006.

[60] S. Lafon, Y. Keller, and R. R. Coifman, "Data fusion and multicue data matching by diffusion maps," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1794–1797, Nov. 2006.

[61] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 550, pp. 2319–2323, Dec. 2000.

[62] R. Souvenir, Q. Zhang, and R. Pless, "Image manifold interpolation using free-form deformations," in *IEEE Int. Conf. Image Processing*, 2006, pp. 1437–1440.

[63] K. Tarabanis, R. Y. Tsai, and D. S. Goodman, "Modeling of a computer-controlled zoom lens," in *IEEE Int. Conf. Robotics and Automation*, May 1992, pp. 1545–1551.

[64] C. Daniel and F. S. Wood, *Fitting equations to data*, John Wiley & Sons, 1980.

[65] S. Arlinghaus, *Practical handbook of curve fitting*, CRC press, 1994.

[66] N. R. Draper and H. Smith, *Applied regression analysis*, John Wiley & Sons, 1998.

[67] D. H. Mclain, "Two dimensional interpolation from random data," *The Computer J.*, vol. 19, no. 2, pp. 178–181, May 1976.

[68] D. Levin, "The approximation power of moving least-squares," *Mathematics of Computation*, vol. 67, no. 224, pp. 1517–1531, Oct. 1998.

[69] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[70] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Conf. Visualization*, Nov. 2002, pp. 163–170.

[71] D. Levin, "Mesh-independent surface interpolation," in *Geometric Modeling for Scientific Visualization*, Jan. 2004, pp. 37–49.

[72] B. Mederos, L. Velho, and L. H. de Figueiredo, "Moving least squares multiresolution surface approximation," in *Brazilian Symp. Computer Graphics and Image Processing*, Oct. 2003, pp. 19–26.

[73] C. Shen, J. F. O'Brien, and J. R. Shewchuk, "Interpolating and approximating implicit surfaces from polygon soup," in *ACM SIGGRAPH*, Aug. 2004, pp. 896–904.

[74] J. Y. Bouguet, *Camera calibration toolbox for MATLAB*, Computational Vision Group, California Institute of Technology, Pasadena, CA, USA, 2001, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[75] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[76] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," in *ACM Int. Conf. Computer Graphics and Interactive Techniques*, Jul. 1980, pp. 124–133.

[77] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, John Wiley & Sons, Inc., 3rd edition, 2003.

[78] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[79] B. A. Devijver and J. Kittler, *Pattern recognition: A statistical approach*, Prentice Hall, 1st edition, 1982.

[80] Y. Ohtake, A. Belyaev, and H. P. Seidel, "An integrating approach to meshing scattered point data," in *ACM Symp. Solid and Physical Modeling*, Jun. 2005, pp. 61–69.

[81] Stanford Computer Graphics Laboratory, "The stanford bunny," http://graphics.stanford.edu/data/3Dscanrep/.

[82] M. H. Gross, R. Gatti, and O. G. Staadt, "Fast multiresolution surface meshing," in *IEEE Conf. Visualization*, Oct. 1995, pp. 135–142.

[83] A. D. Sappa and M. .A. Garcia, "Coarse-to-fine approximation of range images with bounded error adaptive triangular meshes," *SPIE J. Electronic Imaging*, vol. 16, no. 2, Apr. 2007.

[84] R. Pajarola, "Overview of quadtree-based terrain triangulation and visualization," Tech. Rep. UCI-ICS-02-01, Information and Computer Science, University of California Irvine, 2002.

[85] W.-C. Chen, H. Towles, L. Nyland, G. Welch, and H. Fuchs, "Toward a compelling sensation of telepresence: Demonstrating a portal to a distant (static) office," in *IEEE Visualization Conf.*, Oct. 2000, pp. 327–333.

[86] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *ACM SIGGRAPH*, Aug. 1997, pp. 209–216.

[87] L. Demaret, N. Dyn, and A. Iske, “Image compression by linear splines over adaptive triangulations,” *Signal Processing*, vol. 86, no. 7, pp. 1604–1616, Jul. 2006.

[88] Y. Wang, O. Lee, and A. Vitro, “Use of two-dimensional deformable mesh structures for video coding, part II–the analysis problem and a region-based coder employing an active mesh representation,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 6, pp. 647–659, 1996.

[89] M. A. García, B. X. Vintimilla, and A. D. Sappa, “Efficient approximation of gray-scale images through bounded error triangular meshes,” in *IEEE Int. Conf. Image Processing*, Oct. 1999, pp. 168–172.

[90] Y. Yang, M. N. Wernick, and J. G. Brankov, “A fast approach for accurate content-adaptive mesh generation,” *IEEE Trans. Image Processing*, vol. 12, no. 8, pp. 866–880, Aug. 2003.

[91] F. Davoine, M. Antonini, J. Chassery, and M. Barlaud, “Fractal image compression based on delaunay triangulation over vector quantization,” *IEEE Trans. Image Processing*, vol. 5, no. 2, pp. 338–346, 1996.

[92] V. C. Da Silva and J. M. De Carvalho, “Image compression via tritree decomposition,” in *Brazilian Symp. Computer Graphics and Image Processing*, Oct. 2000, p. 339.

[93] L. Demaret, N. Dyn, M. S. Floater, and A. Iske, “Adaptive thinning for terrain modelling and image compression,” in *Advances in Multiresolution for Geometric Modelling*, 2004, pp. 321–340.

[94] D. Hale, “Atomic images - a method for meshing digital images,” in *The 10th Int. Meshing Roundtable*, Oct. 2001, pp. 185–196.

[95] Y. Yang, J. G. Brankov, and M. N. Wernick, “Content-adaptive mesh modeling for fully-3D tomographic image reconstruction,” in *IEEE Int. Conf. Image Processing*, Sep. 2002, pp. 621–624.

[96] P. Vandewalle, L. Sbaiz, J. Vandewalle, and M. Vetterli, “How to take advantage of aliasing in bandlimited signals,” in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2004, pp. 948–951.

[97] X. Zhu, A. T. S. Ho, and P. Marziliano, “Image authentication and restoration using irregular sampling for traffic enforcement applications,” in *Int. Conf. Innovative Computing, Information and Control*, Aug. 2006, pp. 62–65.

[98] X. Zhu, A. T. S. Ho, and P. Marziliano, “A new semi-fragile image watermarking with robust tampering restoration using irregular sampling,” *Elsevier Science*, vol. 2, no. 5, pp. 62–65, June 2007.

[99] M. Kardouchi, J. Konradz, and C. V. Azquezy, "Estimation of large-amplitude motion and disparity fields: Application to intermediate view reconstruction," in *IS&T/SPIE Symp. Electronic Imaging*, Jan. 2001, pp. 340–351.

[100] M. Petrou, R. Piroddi, and A. Talebpour, "Texture recognition from sparsely and irregularly sampled data," *Computer Vision and Image Understanding*, vol. 102, no. 1, pp. 95–104, 2006.

[101] G. Ramponi and S. Carrato, "An adaptive sampling algorithm and its application to image coding," *Image and Vision Computing*, vol. 19, no. 7, pp. 451–460, May 2001.

[102] F. Marvasti, Ed., *Nonuniform Sampling*, chapter 6, Springer, 1st edition, 2001.

[103] Y. Itoh, "An edge-oriented progressive image coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 135–142, Apr. 1996.

[104] M. Domanski, M. Bartkowiak, and M. Szkudlarski, "Nonuniform sampling of chrominance and its application to intra-frame coding," in *Int. Workshop Systems, Signals and Image Processing*, Nov. 2002.

[105] K.-L Hung and C.-C. Chang, "New irregular sampling coding method for transmitting images progressively," in *IEE Proc. Vision, Image and Signal Processing*, Feb. 2003, pp. 44–50.

[106] L.-C. Kuo and S.-J. Wang, "A feature-based scalable codec for image compression," in *Int. Conf. Advanced Information Networking and Applications*, Mar. 2005, pp. 87–90.

[107] S.-J. Wang, L.-C. Kuo, H.-H. Jong, and Z.-H. Wu, "Representing images using points on image surfaces," *IEEE Trans. Image Processing*, vol. 14, no. 8, pp. 1043–1056, Aug. 2005.

[108] Y. Wang and O. Lee, "Active mesh–a feature seeking and tracking image sequence representation scheme," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 610–624, 1994.

[109] M. Rabbani and P. W. Jones, *Digital image compression techniques*, SPIE Press, 1st edition, 1991.

[110] R. Goldman, *Pyramid Algorithms: A dynamic programming approach to curves and surfaces for geometric modeling*, Morgan Kaufmann, 1st edition, 2002.

[111] H. Radha, M. Vetterli, and R. Leonardi, "Image compression using binary space partitioning trees," *IEEE Trans. Image Processing*, vol. 5, no. 12, pp. 1610–1624, 1996.

[112] S. Ø. Wille, "A structured tri-tree search method for generation of optimal unstructured finite element grids in two and three dimenstions," *Int. J. Numerical Methods in Fluids*, vol. 14, no. 7, pp. 861–881, 1992.

[113] M. F. Cohen and J. R. Wallace, *Radiosity and realistic image synthesis*, Morgan Kaufmann, 1st edition, 1993.

[114] G. H. Golub and C. F. Van Loan, *Matrix computations*, John Hopkins University Press, 3rd edition, 1996.

[115] D. Scharstein and R. Szeliski, "Middlebury stereo vision research page," http://vision.middlebury.edu/stereo/data/.

[116] S. M. Savaresi and D. L. Boley, "On the performance of bisecting k-means and pddp," in *SIAM Int. Conf. Data Mining*, Apr. 2001, pp. 1–14.

[117] C. Ding and X. He, "K-means clustering via principal component analysis," in *Int. Conf. Machine Learning*, Jul. 2004, pp. 225–232.

[118] M. Pollefeys, R. Koch, and L. Van Gool, "A simple and efficient rectification method for general motion," in *Int. Conf. Computer Vision*, Sep. 1999, pp. 496–501.

[119] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, Jul. 2000.

[120] D. Oram, "Rectification for any epipolar geometry," in *British Machine Vision Conf.*, Sep. 2001, pp. 653–662.

[121] I. E. G. Richardson, *H.264 and MPEG-4 video Ccmpression: video coding for next generation multimedia*, John Wiley & Sons, Inc., 1st edition, 2003.

[122] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy, "Real time correlation based stereo: algorithm implementations and applications," Tech. Rep. RR-2013, INRIA, 1993.

[123] A. Darabiha, J. Rose, and W. J. MacLean, "Video-rate stereo depth measurement on programmable hardware," in *Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2003, pp. 203–210.

[124] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *Int. J. Computer Vision*, vol. 47, no. 1, pp. 229–246, Apr. 2002.

[125] C. Sun, "Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques," *Int. J. Computer Vision*, vol. 47, no. 1, pp. 99–117, May 2002.

[126] C. Leung, B. Appleton, and C. Sun, "Fast stereo matching by iterated dynamic programming and quadtree subregioning," in *Britich Machine Vision Conf.*, Sep. 2004, pp. 97–106.

[127] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby, "PMF: A stereo correspondence algorithm using a disparity gradient limit," *Perception*, vol. 14, no. 4, pp. 449–470, Mar. 1985.

[128] W. Guo-Qing, W. Brauer, and G. Hirzinger, "Intensity- and gradient-based stereo matching using hierarchical gaussian basis functions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1143–1160, Nov. 1998.

[129] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *Int. J. Computer Vision*, vol. 35, no. 3, pp. 269–293, Dec. 1999.

[130] H. H. Baker and T. O. Binford, "Depth from edge and intensity based stereo," in *Int. Joint Conf. Artificial Intelligence*, Aug. 1981, pp. 631–636.

[131] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 7, no. 2, pp. 139–154, 1985.

[132] D. Marr and T. Poggio, "A cooperative computation of stereo-disparity," *Science*, vol. 194, no. 4262, pp. 283–287, Oct. 1976.

[133] J. E. W. Mayhew and J. P. Frisby, "The computation of binocular edges," *Perception*, vol. 9, no. 1, pp. 69–86, Jan. 1980.

[134] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Computer Vision*, vol. 70, no. 1, pp. 41–54, Oct. 2006.

[135] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, no. 6053, pp. 314–319, Oct. 1985.

[136] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

[137] C. L. Zitnick and T. Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 675–684, Jul. 2000.

[138] J. Mairal, R. Keriven, and A. Chariot, "Fast and efficient dense variational stereo on GPU," in *Int. Symp. 3D Data Processing, Visualization and Transmission*, Jun. 2006, pp. 97–104.

[139] J. Woetzel and R. Koch, “Real-time multi-stereo depth estimation on GPU with approximative discontinuity handling,” in *European Conf. Visual Media Production*, Mar. 2004, pp. 245–254.

[140] O. Veksler, “Stereo correspondence by dynamic programming on a tree,” in *IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2005, pp. 384–395.

[141] Y. Deng and X. Lin, “A fast line segment based dense stereo algorithm using tree dynamic programming,” in *European Conf. Computer Vision*, May 2006, pp. 201–212.

[142] C. Lei, J. Selzer, and Y.-H. Yang, “Region-tree based stereo using dynamic programming optimization,” in *IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2006, pp. 2378–2385.

[143] A. Criminisi, J. Shotton, A. Blake, C. Rother, and P. H. S. Torr, “Efficient dense stereo with occlusion by four-state dynamic programming,” *Int. J. Computer Vision*, vol. 71, no. 1, pp. 89–110, Jan. 2007.

[144] R. Yang, M. Pollefeys, and S. Li, “Improved real-time stereo on commodity graphics hardware,” in *IEEE Conf. Computer Vision and Pattern Recognition Workshops*, Jun. 2004.

[145] M. Gong and R. Yang, “Image-gradient-guided real-time stereo on graphics hardware,” in *Int. Conf. 3-D Digital Imaging and Modeling*, Jun. 2005, pp. 548–555.

[146] M. Gong and Y.-H. Yang, “Near real-time reliable stereo matching using programmable graphics hardware,” in *IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2005, pp. 924–931.

[147] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nistér, “High quality real-time stereo using adaptive cost aggregation and dynamic programming,” in *Int. Symp. 3D Data Processing, Visualization and Transmission*, Jun. 2006, pp. 798–805.

[148] S. Geman and D. Geman, “Stochastic relaxation, gibbs distributions, and the bayesian restoration of images,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, Nov. 1984.

[149] H. Hirschmüller, “Stereo processing by semi-global matching and mutual information,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[150] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér, “Real-time global stereo matching using hierarchical belief propagation,” in *British Machine Vision Conf.*, Sep. 2006, pp. 989–998.

[151] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Trans. Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[152] R. Sara, “Finding the largest unambiguous component of stereo matching,” in *European Conf. Computer Vision*, May 2002, pp. 900–914.

[153] J. Kostkova and R. Sara, “Stratified dense matching for stereopsis in complex scenes,” in *British Machine Vision Conf.*, Sep. 2003, pp. 339–348.

[154] Y. Xu, D. Wang, T. Feng, and H.-Y. Shum, “Stereo computation using radial adaptive windows,” in *Int. Conf. Pattern Recognition*, Aug. 2002, pp. 595–598.

[155] O. Veksler, “Fast variable window for stereo correspondence using integral images,” in *Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2003, pp. 556–561.

[156] K.-J. Yoon and I.-S.Kweon, “Adaptive support-weight approach for correspondence search,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650–656, Apr. 2006.

[157] D. Scharstein and R. Szeliski, “High-accuracy stereo depth maps using structured light,” in *IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2003, pp. 195–202.

[158] H. Longuet-Higgins, “A computer algorithm for reconstructing a scene from projections,” *Nature*, vol. 293, pp. 133–135, Sep. 1981.

[159] O. Pizarro, R. Eustice, and H. Singh, “Relative pose estimation for instrumented, calibrated platforms,” in *Digital Image Computing, Technologies and Applications Conf.*, 2003.

[160] D. Nistér, “An efficient solution to the five-point relative pose problem,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, Jun. 2004.

[161] H. Stewénius, C. Engels, and D. Nistér, “Recent developments on direct relative orientation,” *ISPRS J. Photogrammetry and Remote Sensing*, vol. 60, no. 4, Jun. 2006.

[162] U. Helmke, K. Hüper, P.Y. Lee, and J.B. Moore, “Essential matrix estimation using gauss-newton iterations on a manifold,” *Int. J. Computer Vision*, vol. 74, no. 2, pp. 117–136, Aug. 2007.

[163] W. Zhao, D. Nistér, and S. Hsu, “Alignment of continuous video onto 3D point clouds,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1305–1318, 2005.

[164] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Optical Society of America A*, vol. 4, no. 4, pp. 629–642, Apr. 1987.

[165] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least squares fitting of a two 3D point sets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 698–700, Apr. 1987.

[166] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Optical Society of America A*, vol. 5, no. 4, pp. 1127–1135, Apr. 1988.

[167] E. Bandari, N. Goldstein, I. Nesnas, and M. Bajracharya, "Efficient calculation of absolute orientation with outlier rejection," in *BMVA Symp. Spatiotemporal Image Processing*, Mar. 2004.

[168] P. Besl and N. Mckay, "A method of registration of 3-D shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[169] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Int. Conf. 3D Digital Imaging and Modeling*, May 2001, pp. 145–152.

[170] N. Gelfand, L. Ikemoto, S. Rusinkiewics, and M. Levoy, "Geometrically stable sampling for the ICP algorithm," in *Int. Conf. 3-D Digital Imaging and Modeling*, Oct. 2003, pp. 260–267.

[171] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust euclidean alignment of 3D point sets: The trimmed iterative closest point algorithm," *Image and Vision Computing*, vol. 23, no. 3, pp. 299–309, Mar. 2005.

[172] J. R. Smith, *Integrated Spatial and Image Feature System: Retrieval analysis and compression*, Ph.D. thesis, Columbia University, New York, NY, USA, 1997.

[173] D. Zhang and G. Lu, "Evaluation of similarity measure for image retrieval," in *Int. Conf. Neural Networks & Signal Processing*, Dec. 2003, pp. 928–931.

[174] Y. Xue, C. S. Tong, and W. Zhang, "Survey of distance measures for NMF-based face recognition," in *Computational Intelligence and Security*, Nov. 2006, pp. 1039–0149.

[175] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *Int. J. Computer Vision*, vol. 59, no. 3, pp. 207–232, Sep. 2004.

[176] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 516–441, Dec. 2005.

[177] K. Ni, D. Steedly, and F. Dellaert, "Out-of-core bundle adjustment for large-scale 3D reconstruction," in *Int. Conf. Computer Vision*, Oct. 2007, pp. 1–8.

[178] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proc. Int. Workshop on Vision Algorithms*, Sep. 2000, pp. 298–375.

[179] R. Kumar and A. R. Hanson, "Robust methods for estimating pose and a sensitivity analysis," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 60, no. 3, pp. 313–342, Nov. 1994.

[180] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–780, 1999.

[181] C.-P. Lu, "Fast and globally convergent pose estimation from video images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 610–622, 2000.

[182] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 1–22, 2003.

[183] K. Levenberg, "A method for the solution of certain non-linear problems in least-squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, Jul. 1944.

[184] D. Marquardt, "An algorithm for the least-squares estimation for non-linear parameters," *SIAM J. Applied Mathematics*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[185] J. Nocedal and S. Wright, *Numerical optimization*, Springer, 2000.

[186] M. Lourakis and A. Argyros, "Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustement?," in *Int. Conf. Computer Vision*, Oct. 2005, pp. 1526–1531.

[187] J. E. Dennis, D. M. Gay, and R. E. Walsh, "An adaptive nonlinear least-squares algorithm," *ACM Trans. Mathematical Software*, vol. 7, no. 3, Sep. 1981.

[188] C. J. Taylor and D. J. Kriegman, "Minimization on the lie group SO(3) and related manifolds," Tech. Rep. 9405, Yale University, Apr. 1994.

[189] C. Udriste, *Convex functions and optimization methods on riemannian manifolds*, Kluwer Academic Publishers, 1994.

[190] A. Baker, *Matrix groups: An introduction to lie group theory*, Springer, 2003.

[191] Y. Ma, S. Soatto, J. Kosecka, and S. Shankar Sastry, *An invitation to 3-D vision*, Springer, 2005.

[192] S. M. LaValle, *Planning algorithms*, Cambridge University Press, 2006.

[193] M. D. Plumbley, "Lie group methods for optimization with orthogonality constraints," in *Int. Conf. Independent Component Analysis and Blind Signal Separation*, Oct. 2007, pp. 1245–1252.

[194] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian, "Global registration of multiple 3D point sets via optimization-on-a-manifold," in *Eurographics Symp. Geometry processing*, Jul. 2005, pp. 187–196.

[195] T. Wang, A. G. Backhouse, and I. Y. H. Gu, "Online subspace learning on grassmann manifold for moving object tracking in video," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Mar. 2008, pp. 969–972.

[196] D. G. Luenberger, *Linear and nonlinear programming*, Springer, 2nd edition, Sep. 2003.

[197] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004.

[198] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D model construction for turn-table sequences," in *Proc. SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, Jun. 1998, pp. 154–170.

[199] W. Niem, "Robust and fast modelling of 3D natural objects from multiple views," in *Proc. SPIE Image and Video Processing II*, Feb. 1994, pp. 338–397.

[200] S. Behrendt, "Rendering dynamic real–world scenes using image spheres," in *Int. Symp. Visual Computing*, Nov. 2006, pp. 467–479.

[201] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.

[202] J. Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm," 2000.

[203] D. Tell and S. Carlsson, "Wide baseline point matching using affine invariants computed from intensity profiles," in *European Conf. Computer Vision*, Jun. 2000, pp. 814–828.

[204] B. Georgescu and P. Meer, “Point matching under large image deformations and illumination changes,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 674–688, 2003.

[205] T. Goedeme, T. Tuytelaars, and L. Van Gool, “Fast wide baseline matching for visual navigation,” in *Int. Conf. Computer Vision and Pattern Recognition*, Jun. 2004, pp. 24–29.

[206] Y. Kanazawa and K. Uemura, “Wide baseline matching using triplet vector descriptor,” in *British Machine Vision Conf.*, Sep. 2006, pp. 267–276.

[207] Y. Dufournaud, C. Schmid, and R. Horaud, “Matching images with different resolutions,” in *IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2000, pp. 612–618.

[208] Y. Dufournaud, C. Schmid, and R. Horaud, “Image matching with scale adjustment,” *Computer Vision and Image Understanding*, vol. 93, no. 2, Feb. 2004.

[209] A. R.J. François, “Real-time multi-resolution blob tracking,” Tech. Rep. IRIS-04-422, Institute for Robotics and Intelligent Systems, University of Southern California, 2004.

[210] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi, “Optical flow-based real-time object tracking using non-prior training active feature model,” *Real-Time Imaging, Special Issue on Video Object Processing*, vol. 1, no. 3, pp. 204–218, 2005.

[211] T. Moritani, S. Hiura, and K. Sato, “Real-time object tracking without feature extraction,” in *Int. Conf. Pattern Recognition*, Aug. 2006, pp. 747–750.