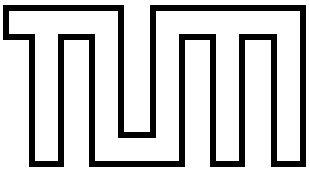


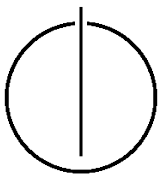
FAKULTÄT FÜR INFORMATIK

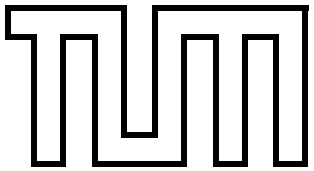
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

# Point and Line Feature based Bundle Adjustment for a Real-Time Tracking System

Georg Barbieri





FAKULTÄT FÜR INFORMATIK

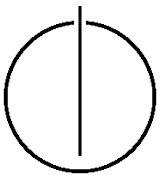
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

# Point and Line Feature based Bundle Adjustment for a Real-Time Tracking System

# Bündelausgleichsrechnung unter Einbezug von Punkt- und Linienmerkmalen für ein Echtzeittrackingsystem


| | |
|---|---|
| Author: | Georg Barbieri |
| Supervisor: | Prof. Gudrun Klinker, Ph.D. |
| Advisor: | Dipl.-Inf. Christian Waechter |
| Date: | February 15, 2012 |

I assure the single handed composition of this master's thesis only supported by declared resources.

München, den 15. Februar 2012 Georg Barbieri

# Abstract

Bundle Adjustment is a widely used discipline in computer vision. Reconstructions of a scene, obtained from images, can be improved with this technique minimising the reprojection error in the images. This thesis provides an approach of performing a bundle adjustment procedure on a scene consisting of points, lines and camera poses.
A method is presented to reconstruct points, lines and camera poses from an image sequence, obtained from a calibrated camera. These reconstructions are then used as initialisation for the bundle adjustment procedure. In this minimisation algorithm the points, lines and camera poses are refined simultaneously, minimising the the reprojection error of points and lines in the images. Appropriate parametrisations for points, lines and camera poses are given such that a bundle adjustment problem can be formulated to optimise these entities.
The reconstruction of points, lines and cameras, that is needed as initialisation for the bundle adjustment procedure, is obtained from three images each time computing the trifocal tensor from the image correspondences. From this tensor an euclidean reconstruction of the camera poses is computed. Then points and lines are reconstructed by triangulation. The single reconstructions are concatenated to a large scene.
The Algorithm is tested on synthetic data as well as on real images. The experimental results given for varying image errors on points and lines as well as for varying configurations of the scene. Even in the case of noisy point and line data the algorithm performs robustly. The presented concept might be used to augment a tracking system that considers points as image features only.

# Contents

# Outline of the Thesis

**Chapter 1:** In the first chapter the introduction to the thesis is provided. First the problem is defined. Then the related work to the topic is presented. In the subsequent section the theoretical background on the the topics that are treated in this thesis are introduced. Furthermore some theoretical and mathematical background is provided.

**Chapter 2:** In the second chapter the reconstruction process from point and line features across three views is describe in detail. The methods used in the thesis are explained and then summarised in terms of an algorithms. In the end of the chapter the single steps are outlined in a summary.

**Chapter 3:** The third chapter treats the topic of Bundle Adjustment in the presence of point and line features. First an appropriate parametrisation of points, lines and cameras is provided, then the bundle adjustment procedure is explained in detail. In the end of the chapter the procedure is summarised and outlined as algorithm.

**Chapter 4:** Chapter four provides an evaluation and the results of the reconstruction and refinement process. First the evaluation on synthetic data is shown in various tests, then the algorithm is applied and evaluated on real image data.

**Chapter 5:** In the last chapter the developed method is summarised and possibilities for further developments and improvements are discussed.

# 1 Introduction

## 1.1 Motivation

Computer vision is a discipline that deals with extracting information from images. One of the main task is to extract information about objects in the three-dimensional world from images such as their position, shape, colour or kind. Furthermore knowledge about the camera system that is recording the images of the scene can be provided. Thus, the internal camera parameters like the focal length as well the external camera parameters can be obtained.
One field of research in computer vision where image features are used to estimate the relations of camera views and scene is tracking. Such tracking algorithms estimate the camera poses and the scene structure from images. Since image data are noisy the reconstructions are obtained with a certain error. This error in the reconstruction can be reduced with bundle adjustment procedures, that optimises the reconstructions.
Most of the feature tracking algorithms are based on point features. Significant points in the images are extracted, correspondences across multiple images are found and the spatial relations of the cameras and the scene points are reconstructed from the image correspondences.
In human made environments however straight edges are present everywhere. The projection of a three-dimensional straight edge into the image plane of a perspective camera causes in a straight line. Lines have some interesting properties and behave different to points in several ways. Imagine a scene in a human made environment. Lets take the example in figure 4.10 on page 68. Seven images of a scene are illustrated. The scene consists of many edges and corners, that can be extracted in the images. The camera that recorded the scene moved smoothly around the scene and provides images from different perspectives. It is obvious to see that many significant points are not visible across the complete sequence or may be occluded in some perspectives by other objects. These points can not be used as correspondence across the complete scene. The edges however are more robust against occlusion. Even so an edge may be occluded partially in some of the images, the partially visible line in the image provides the same informations as the complete line does, if lines are considered as infinite lines without endpoints. Thus, lines are less affected by occlusion than points. Another advantage of lines towards points is the fact that lines can be detected more accurate in an image than points. Since one line can provide the same informations as many points lying on a line, the amount of data can be reduced, substituting the points by a line fitted trough them. Doing this mismatches of points can be avoided considering one single line instead of a bundle of points lying arbitrarily on this line.
Despite these advantages care is needed when dealing with lines. If an image of a 3D point is provided, it is clear that the three-dimensional point must lie on the ray going trough the camera centre and the image line. Thus, the unknown informations can be reduced to a one-dimensional space. If however an image of a line is provided, the three-dimensional

line lies on a plane that is formed by the camera centrer and the image line. Thus, the search space for the three-dimensional line has two dimensions. This shows that image points provide more information for the reconstruction than image lines. These coherences are illustrated in figure 1.1.

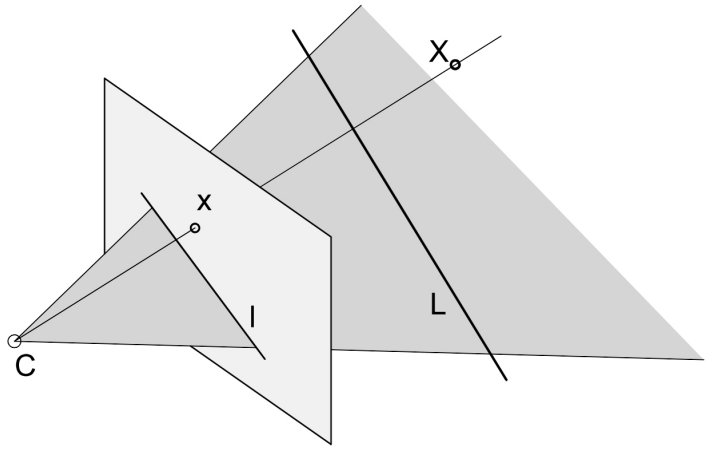



Figure 1.1: ***Projection of points and lines:*** *$\boldsymbol{C}$ denotes the camera centre, $\boldsymbol{x}$ and $\boldsymbol{l}$ respectively denotes the point and line in the image. $\boldsymbol{X}$ and $\boldsymbol{L}$ stands for the corresponding point and line in 3D. The point $\boldsymbol{X}$ lies on the ray going trough the camera centre $\boldsymbol{C}$ and the image point $\boldsymbol{x}$. The line $\boldsymbol{L}$ lies on the plane formed by the camera centre $\boldsymbol{C}$ and the image line $\boldsymbol{l}$.*

Points in the euclidean space are represented simply by its euclidean coordinates. Lines however can not be represented in such a geometrically meaningful way. Thus, caution is needed when representing lines. The representation of points lines and cameras will be an important issue in this thesis.
Klein and Murray presented a camera pose estimation algorithm in 2007 [32]. Based on point feature correspondences across images they estimate the poses of the camera-views and the point-positions in 3D from images obtained from an image stream in real time. In a second process, a map is build consisting of interesting points and the cameras. This map, that is enlarged permanently, will continuously be refined by a global minimisation process, called bundle adjustment. The application performs very well and robust.
Based on this approach the idea arose to augment such a system considering not only points, but also line features in the images. Thus, the reconstruction of the scene is composed of points, lines and cameras. One of the challenging problems is the bundle adjustment process. This minimisation procedure must be formulated such that the estimation of the scene is improved simultaneously for points, lines and cameras based on a reprojection error for points and lines in the images. The objective is to adapt the points, lines and cameras in 3D such that the reprojection error in the images is minimised.
The minimisation process must be initialised with estimations of the three-dimensional points, lines and cameras. This initialisation does not need to be optimal, but the better it is, the better the bundle adjustment algorithm performs. Thus, algorithms are required that reconstruct points, lines and cameras from image correspondences. Since the tracking process must be real-time capable, the reconstruction algorithms should perform fast but as accurate as well. The task of the minimisation algorithm however is not to perform in

real time, but to provide accurate results.
In chapter 2 and chapter 3 algorithms for the desired Tasks are presented. In chapter 4 the algorithms are evaluated.

## 1.2 Related Work

This section provides an overview on the research that is done on topics related to the reconstruction of points and lines from images as well as on bundle adjustment procedures to improve the reconstructions. The reconstruction of scenes from images has been an active field of research for many years in the disciplines of photogrammetry and computer vision. The chapter can also be seen as a historical overview on computer vision. Starting with the first trials of scene reconstructions, the state of the art techniques are outlined, then tracking algorithms that requires the knowledge of reconstruction procedures are presented. Finally bundle adjustment techniques are introduced. These techniques, that has their origin in the field of photogrammetry and are used to refine the reconstructions, are pointed out in the context of computer vision.

### 1.2.1 Scene Reconstruction

The objective of computer vision is to extract the three dimensional structure of the world from images. In 1965 Roberts presented a method to obtain three-dimensional description of the world from the edge informations in an image. To do this he made assumptions on the scene that where used as previous knowledge for the reconstruction [48]. In 1976 Rosenfeld et al. investigated on the understanding and interpretation of lines in an image in the context of a three-dimensional scene [49]. Kanade in the year 1980 examined the problem of obtaining three-dimensional informations from images based on origami models. The paper deals with the question: “how do we understand the possible three-dimensional configurations from a collection of lines ”. Based on a 3D model a 3D meaning is assigned to the lines in the image [31]. Feature based stereo correspondence algorithms where developed for instance by Marr and Poggio in 1976 [38], Mayhew and Frisby in 1981 [41] and Lucas and Kanade in 1981, that developed a fast technique for image registration [37]. Marr in 1982 explored the human vision from a philosophical and computational point of view [39]. In the 1980s a lot of work was put in the improving of image analysis. Thus, image pyramids where introduced, for instance by Burt and Adelson in 1983 [10]. Canny in 1986 described a new approach for the computation of edge points in the image [11]. At this time a wide variety of different algorithms for image understanding and 3D reconstruction was available. Blake and Zisserman in 1987 for instance introduced a new concept for fitting piecewise continuous functions to visual data [8]. Poggio et al. in 1985 described regularisation methods for ill posed problems [46]. Faugeras and Hébert 1987 investigated on the representation of information in 3D and described algorithms to construct these representations [19]. In the 1990s a lot of attention was given to the concept of projective reconstruction. In this process no knowledge about the intrinsic camera parameters is required (for the camera parameters refer to chapter 1.3.2). Faugeras in 1992 examined the problem of determining the kind of 3D reconstruction that can be received from a stereo rig for which no three-dimensional reference data for the calibration is available. The only available infor-

mations are pixel correspondences between the two images [20]. Hartley in 1992 described how internal camera parameters can be determined from a set of three-dimensional points [27]. In 1994 Hartley investigates on projective reconstructions of geometric configurations from two of more perspective views. He used the fundamental matrix that describes the epipolar correspondences between two views. He describes a method how the fundamental matrix can be determined from 7 point correspondences across two images or from 6 point correspondences across three images [22].
The concept of the fundamental matrix was introduced in 1981 from Longuet and Higgins. They described an algorithm for reconstructing the three-dimensional structure of a scene from a stereo pair with unknown spatial relations. If eight point correspondences across the images are provided, the relative orientation of the two projections can be obtained by solving a system of eight linear equations. The spatial coordinates of all visible points can then be obtained [35]. In 1998 Zhang considered the essential matrix for calibrated images pairs as well as the fundamental matrix for uncalibrated image pairs. He gave methods to compute either the fundamental and the essential matrix [65]. These methods are still used in modern approaches.
In 1992 Tomasi Kanade proposed a factorisation method to obtain the shape of the scene as well as the camera pose [60]. Triggs in 1996 described an algorithm to recover the structure of the scene and the cameras from multiple perspective cameras by factorisation methods [62]. This mehtod can be seen as generalisation of the algorithm of Tomasi Kanade. Nagel Enkelmann in 1986 used the optical flow techniques for the mapping from one image to the next one in an image stream [43]. In 1992 Bergen et al. examined the use of four different flow techniques for motion estimation [5]. The techniques are affine flow, planar surface flow, rigid body motion, and general optical flow. The technique of optical flow for motion estimation was improved in several ways, for instance by Black in 1996 allowing multiple flows in one image caused by transparency, depth discontinuities, independently moving objects, reflections, shadows and others [7]. Bruhn et al. in 2005 combined local optical flow methods with global optical flow methods [9].
Also the stereo-optical methods where content of research. Okutomi and Kanade 1993 presented a stereo matching method that uses multiple stereo pairs with various baselines to obtain precise distance estimates [44]. Seitz et al. presented a quantitative comparison of several multi view stereo reconstruction algorithms [51].
To describe the spatial relation of an image triplet the trifocal tensor was introduced. The discovery of the tensor may be attributed to Spetsakis, Aloimonos and to Weng. Spetsakis and Aloimonos in 1991 presented a multi frame motion estimation technique based on the principle of the trifocal tensor. Weng presented in 1988 a robust technique for motion and structure estimation from image sequences. He used the principle of the trifocal tensor for scene reconstruction from line correspondences [64]. Shashua in 1994 showed the existence of a trilinear relationship between three perspective images [52]. In 1995 he presented new results on the trilinera relationsip [53]. Hartley in 1995 [23] and 1997 [24] showed that the relations of Shashua for points and for lines can be expressed in a common way, the trifocal tensor. In further research other properties of the trifocal tensor where proposed. Faugeras and Mourrain in 1995 explored the geometric and algebraic relations between correspondences of points and lines in an arbitrary number of images. They provided a new method for deriving the trilinear relations [18].
For reconstructing lines different approaches where proposed. Hartley and Sturm in 1997

proposed an optimal triangulation method for points from two images [28]. Schmid and Zisserman in 1997 presented a method for matching individual line segments between images [50]. Chiba and Kanade in 1998 proposed an automatic line tracking method for line segments over an image sequence using the gray level information of the images and the geometric attributes of the line segments [13]. In 1999 Baillard et al. presented an algorithm that automatically reconstructs buildings from aerial images based on line segments [3]. Hartley and Zisserman in 2003 suggested various methods for point and line reconstructions based on triangulation [25]. Matinec and Pajdla in 2003 propose a factorisation method for line reconstruction from many perspective images. [40]. Bartoli and Sturm in 2005 presented a structure from motion algorithm based on line features [4].

### 1.2.2 Tracking and Mapping Techniques

The methods of computer vision where also used for feature tracking applications. Smith Cheeseman in 1987 presented a method for estimating the relationship and the expected error between coordinate frames. This estimation method can be used to estimate the likelihood whether a particular reference object in its field of view lies in front of a camera attached to a robot. This work may be seen as the invention of the EKF-SLAM (extended Kalman filter - simultaneous localization and mapping) methods [56]. Another important work on the topic of simultaneous localisation and mapping was published in 1991 by Leonard and Durrant-Whyte. They formulated the problem of localisation and mapping of a robot in an environment as “chicken egg problem ”and discussed the problem of precise motion of a robot in an environment on the basis of an accurate self made map, sensing the environment [34]. In 2003 Montemerlo et al. presented an improved algorithm of the simultaneous localisation and mapping problem, called FastSLAM. The improvements affect the efficiency of the algorithm as well a the accuracy of the map [42]. Davison et al. 2007 presented a method called monoSLAM. This real time algorithm can recover the 3D trajectory of a monocular camera that is moved through a previously unknown scene. The algorithm is proposed a the first pure-vision SLAM algorithm for structure from motion problems that achieves real-time behaviour and drift free performance.[15]. Eade and Drummond in 2006 presented a real-time monocular SLAM system for a single camera system, applying the FastSLAM-type particle filter to single-camera SLAM [17]. Klein and Murray in 2007 presented a method for the estimation of camera poses in an unknown scene. The system called PTAM is designed specially for hand-held cameras in small workspaces. The novelty compared to existing SLAM methods was the splitting of the tracking and mapping procedure, that where executed in two separate tasks. While one task deals with the robust tracking of the camera and performs in real time, the other task generates a map of 3D point features. On this mapping task a non real time capable bundle adjustment procedure is executed that continuously refines the map [32]. In 2008 this PTAM system was extended by Castle et al. to allow one or more cameras to work in several maps, separately or simultaneously [12].

### 1.2.3 Optimisation Techniques

Bundle adjustment originally comes from the field of photogrammety. Many textbooks about this topic are available. Atkinson [1] gives an introduction in the field of non-linear

optimisation problems for photogrammety. Björck in 1996 provided a textbook that treats several methods of least squares problems such as methods for sparse least squares problems, iterative methods, modified least squares, weighted problems, and constrained and regularized problems applicable for a great number of scientific areas [6].
In the context of computer vision Triggs et al. in 2000 gave a survey of the theory and methods of bundle adjustment. It provides a survey of the theory and methods of bundle adjustments for computer vision. The paper is dedicated to people that already have some knowledge and experience with bundle adjustment methods. Topics such as the choice of an appropriate cost function and robustness, numerical methods such as sparse Newton methods and recursive methods are treated [63]. Hartley and Zisserman introduced in their textbook several non-linear optimisation algorithms for computer vision applications [25]. Full global optimization techniques that later where realised to be the same as the bundle adjustment technique where proposed. For instance the method of Taylor et al. in 1991 that addressed a special case of the structure from motion problem for static scenes [59] or the method of Szeliski and Kang in 1993. They presented a shape and motion estimation algorithm based on non-linear least squares [58]. Also Azarbayejani et al. in 1995 presented a formulation for recursive recovery of motion, structure and the focal length from feature correspondences from an image sequence. The used techniques later where assigned to the field of bundle adjustment [2]. Sibley in 2008 presented an approach for deriving a relative objective function for bundle adjustment for a simultaneous localisation an mapping method [54].

## 1.3 Multiple View Geometry

In this section the principals of multiple view geometry that are used for this thesis are introduced. In [21] Forsyth and Ponce give detailed explanations on the various principles of computer vision. Hartley and Zisserman in [25] provide the geometrical and algebraical concepts of multiple view geometry and computer vision, completed with algorithms and helpful suggestions. From there most concepts of computer vision used here where borrowed.

### 1.3.1 Projective Geometry

The projective geometry describes perspective transformations in the two dimensional and three dimensional space as well as the mapping from the three dimensional space to the two dimensional space. In contrast to the euclidean geometry, that describes the metric properties of figures, the projective geometry allows projective transformations. For instance, two parallel lines that lie in the same plane do intersect in the projective space in the point at infinity. This point that belongs to the set of ideal points plays an important role in the projective geometry. Throughout this thesis the projective space is denoted with $\mathbb{P}$ whereas the euclidean space is denoted with $\mathbb{R}$. Points in the projective space are represented as homogeneous coordinates. In the projective 3-space $\mathbb{P}^3$ a point $\mathbf{X}$ is represented as homogeneous 4 vector, adding a $1$ as last coordinates to the euclidean coordinates according to $\mathbf{X} = (X, Y, Z, 1)^\top$. To a homogeneous point a projective transformation can be applied. Let $\mathbf{H}$ be a $4 \times 4$ matrix and representing the projective transformation. The matrix $\mathbf{H}$ is a homo-

geneous matrix with 16 elements and 15 degrees of freedom plus an overall scaling of the matrix. This matrix can represent several forms of linear transformations. Zisserman and Hartley in [25] in chapter 2 - chapter 4 give a detailed overview of the projective geometry in two and three dimensions.

### 1.3.2 Camera Model

A camera projects a scene from the 3D space into a 2D image plane. Points and lines can be mapped using a $3 \times 4$ camera matrix $\mathbf{P}$. This camera matrix encapsulates the pose of the camera in the world, called *extrinsic camera parameters* as well as the internal calibration parameters of the camera, called *intrinsic camera parameters*. Thus, $\mathbf{P}$ has $11$ degreed of freedom, $6$ for the camera pose and $5$ for the internal calibration. A camera matrix $\mathbf{P}$ is composed of

$$\mathbf{P} = \mathbf{KR}\left[\mathbf{I} \mid -\tilde{\mathbf{C}}\right]. \tag{1.1}$$

The matrix $\mathbf{K}$ contains the internal parameters of the camera, the matrix $\mathbf{R}$ and the vector $\tilde{\mathbf{C}}$ describe the orientation and the position of a camera in a world frame. Hartley and Zisserman in [25] describe several camera models. Here a general projective camera model is used.
The $3 \times 4$ camera matrix $\mathbf{P}$ can be rewritten as $\mathbf{P} = [\mathbf{M} \mid \mathbf{p}_4]$. It is recommended from [25] chapter 6.2 to normalise the camera matrix so that $\|\mathbf{m}^3\| = 1$ where $\mathbf{m}^3$ is the $3^{rd}$ row of the matrix $\mathbf{M}$. Furthermore if $det(\mathbf{M}) < 0$, the camera matrix $\mathbf{P}$ must be multiplied by $-1$. This is important to define a right handed camera frame.

#### Extrinsic Camera Parameter

The frame attached to a camera is related to a world frame via a translation and a rotation. In the three-dimensional space there are $6$ degrees of freedom that describe the pose of the camera frame relative to the world frame, $3$ for the translation and $3$ for the rotation. These parameters are called the extrinsic camera parameter.
Suppose that $\mathbf{C} = (X_C, Y_C, Z_C, 1)^\top$ is a homogeneous vector representing the centre of the camera frame. It specifies the translation of the camera frame relative to the world frame.
The matrix $\mathbf{R}$ is a $3 \times 3$ rotation matrix that represents the rotation of the camera frame relative to the world frame.
A rotation matrix $\mathbf{R}$ of the form

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1.2}$$

can be composed of a sequence of three rotations around the three axis $x$, $y$, $z$. Thus, $\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi)$. The angle $\psi$ denotes the rotation around the $x$-axis, $\theta$ denotes the rotation around the $y$-axis and $\phi$ denotes the rotation around the $z$-axis. The order of rotation influences the final result. Here the rotations are defined in the order of first rotating around the x-axis, then around the y-axis and finally around the z-axis. The rotation matrices $\mathbf{R}_x$,

$\mathbf{R}_y$, $\mathbf{R}_z$ are defined as

$$\mathbf{R}_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{bmatrix} \tag{1.3}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{1.4}$$

$$\mathbf{R}_z(\phi) = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.5}$$

from the equations 1.3 to 1.5 the rotation matrix $\mathbf{R}$ can be obtained according to

$$\begin{aligned} \mathbf{R} &= \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi) \\ &= \begin{bmatrix} cos(\phi)cos(\theta) & cos(\phi)sin(\psi)sin(\theta) - cos(\psi)sin(\phi) & sin(\phi)sin(\psi) + cos(\phi)cos(\psi)sin(\theta) \\ cos(\theta)sin(\phi) & cos(\phi)cos(\psi) + sin(\phi)sin(\psi)sin(\theta) & cos(\psi)sin(\phi)sin(\theta) - cos(\phi)sin(\psi) \\ -sin(\theta) & cos(\theta)sin(\psi) & cos(\psi)cos(\theta) \end{bmatrix}. \end{aligned} \tag{1.6}$$

**Intrinsic Camera Parameter**

The intrinsic parameters of a camera may be described as the focal length of the lens, the principal point, the pixel size and a skew factor in the case of non square pixel. The matrix $\mathbf{K}$, also called *calibration matrix*, encapsulates those parameters according to

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.7}$$

The parameters $\alpha_x$ and $\alpha_y$ represent the focal length with respect to the pixel dimensions in $x$ and $y$ direction and are determined according to $\alpha_x = fm_x$ and $\alpha_y = fm_y$ where $f$ is the focal length of the lens and $m_x = 1/(pixelsize_x)$ and $m_y = 1/(pixelsize_y)$ are the numbers of pixel per unit distance in image coordinates in $x$ and $y$ direction. The parameter $s$ is the skew factor of the pixels. In most cases it is equal to zero. The parameters $x_0$ and $y_0$ represent the principal point. It describes the centre of projections in the image frame.

### 1.3.3 Camera Decomposition

A $3 \times 4$ camera matrix $\mathbf{P}$ can be decomposed into $\mathbf{P} = \mathbf{KR}\left[\mathbf{I} \mid -\tilde{\mathbf{C}}\right]$. The intrinsic parameter matrix $\mathbf{K}$ is a $3 \times 3$ upper triangular matrix, The $3 \times 3$ rotation matrix $\mathbf{R}$ is an orthogonal matrix and $\tilde{\mathbf{C}}$ is the inhomogeneous 3-vector of the camera centrer in the world frame. In [25], chapter 6.2.4, Hartley and Zisserman present methods to find the camera centre, the orientation of the camera and the intrinsic camera matrix.

The camera centre $\mathbf{C} = (X_C, Y_C, Z_C, T)^\top$ may be obtained according to

$$\begin{aligned} X_C &= \ \ det([\mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4]) \\ Y_C &= -det([\mathbf{p}_1,\ \mathbf{p}_3,\ \mathbf{p}_4]) \\ Z_C &= \ \ det([\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_4]) \\ T &= -det([\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3]) \end{aligned} \tag{1.8}$$

The vector $\mathbf{p}_i$ denotes the $i^{th}$ column of the matrix $\mathbf{P}$. The inhomogeneous 3-vector of the camera centre is therefore $\tilde{\mathbf{C}} = (X_C/T, Y_C/T, Z_C/T)^\top$.
The camera matrix can be written as $\mathbf{P} = \left[\mathbf{KR} \mid -\mathbf{KR}\tilde{\mathbf{C}}\right] = \left[\mathbf{M} \mid -\mathbf{M}\tilde{\mathbf{C}}\right]$. From the $3 \times 3$ matrix $\mathbf{M}$ the matrices $\mathbf{K}$ and $\mathbf{R}$ may be found using the RQ-decomposition. This decomposition decomposes the matrix $\mathbf{M}$ into a product of an upper triangular and an orthogonal matrix according to $\mathbf{M} = \mathbf{KR}$. The RQ-decomposition is described in [25], appendix A4.4.1, page 579.

### 1.3.4 Camera Calibration

For the development of the system a customary Webcam, the Logitech QuickCam Pro 4000 was used. As objective a fish eye lens was mounted. Experiments has shown that with a wide angle lens better results can be achieved than with a normal angle lens. Images where recorded with a resolution of 640x480 pixels. The camera was calibrated according to the technique of Zhang, proposed in [66]. As calibration tool the Ubitrack system, introduced in [30] by Huber et al. and in [47] by Pustka et al. was used. With the calibration a distortion of the camera images was achieved such that straight lines in the world are visualised as streight lines in the image. Furthermore a precise calibration matrix $\mathbf{K}$ of the camera was obtained. This intrinsic parameter matrix is of the form

$$\mathbf{K} = \begin{bmatrix} 373,17518 & 0 & 320,86825 \\ 0 & 389,21025 & 241,71514 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.9}$$

The camera has a sensor size of $1/4''$. This results in a pixel size: $5.6\mu m \times 5.6\mu m$.

When working with synthetic data an approximation of the calibration matrix was used. This calibrations matrix is of the form

$$\mathbf{K} = \begin{bmatrix} 380 & 0 & 320 \\ 0 & 380 & 240 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.10}$$

### 1.3.5 Two View Geometry

Suppose, two images of the same scene are given, recorded from different positions of view. These views ma be acquired from a moving camera in different time steps. The relations between these two cameras are described by the epipolar geometry, that depends only on the internal parameters of the cameras and its relative pose. The fundamental matrix $\mathbf{F}$ represents the algebraic relations of the epipolar geometry. Hartley and Zisserman cover

the epipolar geometry in [25], chapter 9. In chapter 10 the perspective reconstruction of a camera pair from the fundamental matrix is described. Chapter 11 describes methods for computing the fundamental matrix from image correspondences. Chapter 12 describes triangulation methods to recover the scene structure.
The fundamental matrix **F** is a $3 \times 3$ matrix. It can be obtained from point correspondences across two images. From the fundamental matrix a perspective reconstruction of the camera pair can be obtained in the general case. If any knowledge about the internal camera parameter or the scene is provided, an euclidean reconstruction up to a scale factor is possible. Knowing the cameras, the scene points can be recovered from the image correspondences.

### 1.3.6 Three View Geometry

What the fundamental matrix **F** is for two views, the trifocal tensor $\mathcal{T}$ is for the three view geometry. Given three views of a scene with image correspondences, a tensor can be computed that encapsulates the relative geometric relations between the three cameras. Hartley and Zisserman introduce the trifocal tensor in [25], chapter 15. In chapter 16 they describe the computation of the trifocal tensor from point and line correspondences.
The $3 \times 3 \times 3$ tensor can be determined from point and line correspondences across three cameras. It can be used to determine the three camera matrices, again up to a perspective transformation. With further knowledge about the scene or the intrinsic camera parameters an euclidean reconstruction can be obtained. One important difference between the trifocal tensor and the fundamental matrix is, that informations from line correspondences can directly be included in the computation of the trifocal tensor. The fundamental matrix however can deal only with point correspondences. In the present problem this fact was the decisive reason to use the trifocal tensor for the initial reconstruction. In Chapter 2 the reconstruction process from an image using the trifocal tensor is explained in detail.

### 1.3.7 Points and Lines 2D

Hartley and Zisserman in [25], chapter 2.2 explain the relations of points in the projective plane. A point **x** in the euclidean 2 space $\mathbb{P}^2$ is simply represented by its euclidean coordinates $\mathbf{x} = (x, y)^\top$. If the point is represented in the projective 2 space $\mathbb{R}^2$, a 1 is added to the point vector according to $\mathbf{x} = (x, y, 1)^\top$. Thus, the point is represented as homogeneous vector.
A line **l** in the plane is described by the equation $ax + by + c = 0$. Thus, the line can be expressed by the homogeneous vector $\mathbf{l} = (a, b, c)^\top$. A homogeneous point **x** lies on the line **l** if and only if the equation $\mathbf{x}^\top\mathbf{l} = \mathbf{l}^\top\mathbf{x} = 0$ is satisfied. The intersection point $\mathbf{x} = (x, y, 1)^\top$ of the two lines $\mathbf{l}_1 = (a_1, b_1, c_1)^\top$ and $\mathbf{l}_2 = (a_2, b_2, c_2)^\top$ can be determined from the cross product of the lines according to $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2 = [\mathbf{l}_1]_\times \mathbf{l}_2$. The $3 \times 3$ matrix $[\mathbf{l}_1]_\times$ is the skew symmetric matrix corresponding to $\mathbf{l}_1 = (a_1, b_1, c_1)^\top$ according to

$$[\mathbf{l}_1]_\times = \begin{bmatrix} 0 & -c_1 & b_1 \\ c_1 & 0 & -a_1 \\ -b_1 & a_1 & 0 \end{bmatrix}. \tag{1.11}$$

This matrix can be used to express the cross product. The skew symmetric matrix will appear in the following chapters in different coherences.

A line $\mathbf{l} = (a, b, c)^\top$ joining two points $\mathbf{x}_1 = (x_1, y_1, 1)^\top$ and $\mathbf{x}_2 = (x_2, y_2, 1)^\top$ can be computed as the cross product of the points according to $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2 = [\mathbf{l}_1]_\times \mathbf{l}_2$.
It is recommended to normalise lines represented as $\mathbf{l} = (a, b, c)^\top$ such that the norm of the first two elements $\|(a, b)^\top\|$ is equal to one. Homogeneous points represented as $\mathbf{x} = (x, y, w)^\top$ should be normalised such that the last element $w$ is equal to 1.

### 1.3.8 Points and Lines in 3D

A point **X** in the euclidean 3-space $\mathbb{R}^3$ is simply represented by its euclidean coordinates $\mathbf{X} = (X, Y, Z)^\top$. In the projective 3-space $\mathbb{P}^3$ the homogeneous representation is appropriate. Thus, $\mathbf{X} = (X, Y, Z, 1)^\top$.

A suitable representation of a line in the euclidean 3-space $\mathbb{R}^3$ is the Plücker matrix **L**. The basic concepts of the Plücker matrix are borrowed from Hartley and Zisserman in [25], chapter 3.3.2, page 68 ff.
The Plücker matrix is a $4 \times 4$ skew-symmetric homogeneous matrix

$$\mathbf{L} = \begin{bmatrix} 0 & l_1 & l_2 & l_3 \\ -l_1 & 0 & l_4 & l_5 \\ -l_2 & -l_4 & 0 & l_6 \\ -l_3 & -l_5 & -l_6 & 0 \end{bmatrix} \tag{1.12}$$

with 6 independent elements not equal to zero, from which their 5 ratios are significant. Since $det(\mathbf{L}) = 0$ the number of degrees of freedom is 4. A line in the euclidean 3-space can be defined by the intersection of two orthogonal planes. Each of the intersection points on the planes has two degrees of freedom. Thus, in total the line has 4 degrees of freedom. This is visualised in figure 1.2.

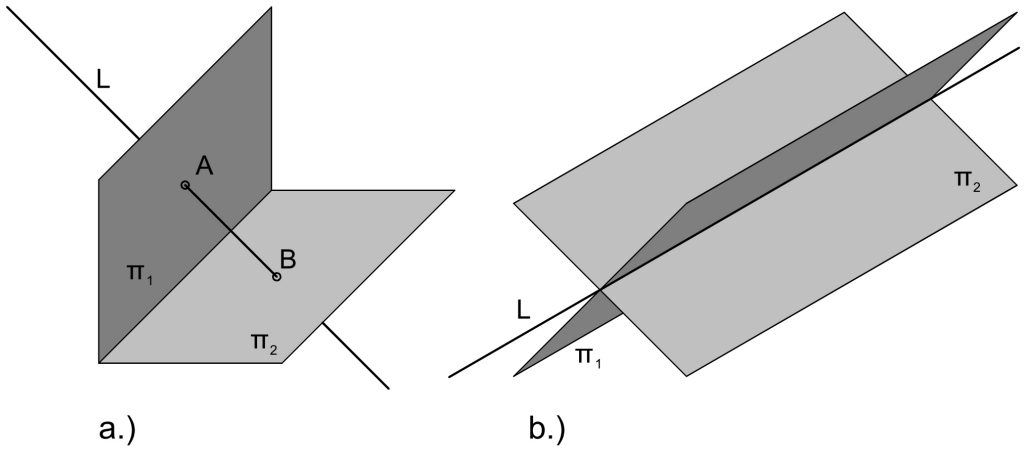



Figure 1.2: ***Definition of a line in the euclidean 3-space:*** *In (a) the line* **L** *is defined by its points of intersection* **A**, **B** *on the two orthogonal planes* $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$. *In (b) the line* **L** *is specified as the intersection of the two arbitrary planes* $\boldsymbol{\pi}_1$ *and* $\boldsymbol{\pi}_2$.

Suppose **A** and **B** are two points in 3-space, then

$$\mathbf{L} = \mathbf{A}\mathbf{B}^\top - \mathbf{B}\mathbf{A}^\top. \tag{1.13}$$

The Plücker Matrix **L** can be understood as the line in 3D joining the two points $\mathbf{A}, \mathbf{B}$ in space. It is independent of the choice of the points on the line.
A line in 3D can be defined also as intersection of two planes in 3-space. Thus, a dual Plücker representation $\mathbf{L}^*$ is obtained. Suppose **P** and **Q** are two planes in 3-space, then

$$\mathbf{L}^* = \mathbf{P}\mathbf{Q}^\top - \mathbf{Q}\mathbf{P}^\top. \tag{1.14}$$

$\mathbf{L}^*$ has similar properties as **L**. **L** can be converted to $\mathbf{L}^*$ and vice versa by the rewrite rule

$$l_{12} : l_{13} : l_{14} : l_{23} : l_{42} : l_{34} = l^*_{34} : l^*_{42} : l^*_{23} : l^*_{14} : l^*_{13} : l^*_{12}. \tag{1.15}$$

To project a line **L** in the image plane of a camera **P**, the transformation

$$[\hat{\mathbf{l}}]_x = \mathbf{P}\mathbf{L}\mathbf{P}^\top \tag{1.16}$$

can be applied. $[\hat{\mathbf{l}}]_x$ is a $3 \times 3$ skew-symmetric matrix.

$$[\hat{\mathbf{l}}]_x = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \tag{1.17}$$

It is defined as the corresponding skew-symmetric matrix of the line vector $\hat{\mathbf{l}} = \left(\hat{a}, \hat{b}, \hat{c}\right)^\top$. Thus, the line vector $\hat{\mathbf{l}}$ is

$$\hat{\mathbf{l}} = \begin{bmatrix} \hat{a} \\ \hat{b} \\ \hat{c} \end{bmatrix} = \begin{bmatrix} -\mathbf{P}^2\mathbf{L}\mathbf{P}^{3\top} \\ \mathbf{P}^1\mathbf{L}\mathbf{P}^{3\top} \\ -\mathbf{P}^1\mathbf{L}\mathbf{P}^{2\top} \end{bmatrix}. \tag{1.18}$$

The vector $\hat{\mathbf{l}} = \left(\hat{a}, \hat{b}, \hat{c}\right)^\top$ is the reprojection of the line **L** in the image of camera **P**, the 4-vector $\mathbf{P}^k$ denotes the $k^{th}$ row of the matrix **P**.
The line vector $\hat{\mathbf{l}}$, that is defined up to a scale factor, represents the parameters of the equation $\hat{a}x + \hat{b}y + c = 0$ that describes a line in the 2D space. The line vector is normalised such that $\|(\hat{a}, \hat{b})^\top\| = 1$.
The line $\mathbf{l} = (a, b, c)^\top$ is the 2D line corresponding to the 3D lines **L** measured in the image plane of the camera **P**. It is normalised in the same way as the reprojected line according to $\|(a, b)^\top\| = 1$.

### 1.3.9 Reconstruction Theory

From several views of a scene the scene can be reconstructed. However, independent of the number of images the scene can not be determined uniquely. Only a relative reconstruction of the scene can be obtained. Its absolute position in the world cannot be determined. Depending on additional knowledge about the scene or the camera, different stages of reconstruction can be obtained. With no further knowledge a perspective reconstruction is possible. This reconstruction can be updated up to a true reconstruction incorporation informations about the scene of the cameras. In [25], chapter 1, Hartley and Zisserman give an overview of the different reconstructions. In chapter 10 they explain how reconstructions can be upgraded using further informations about the camera and the scene.

#### Projective Reconstruction

Without any knowledge about the scene or the internal camera parameters, the scene can be reconstructed up to a perspective ambiguity. Imagine, the camera $\mathbf{P}$ is reconstructed from point correspondences. Let $\mathbf{X}$ be a point in space. Then the point $\mathbf{x}$, the projection of point $\mathbf{X}$ in the image plane of camera $\mathbf{P}$ is given by the coherence $\mathbf{x} = \mathbf{PX}$. Let $\mathbf{H}$ be a $4 \times 4$ transformation matrix that represents a projective transformation. This matrix can be applied to the points $\mathbf{X}$ and cameras $\mathbf{P}$ without any influence on the projected points $\mathbf{x}$ according to $\mathbf{x} = \mathbf{PX} = (\mathbf{PH}^{-1})(\mathbf{HX})$. The matrix $\mathbf{H}$ is arbitrary. This ambiguity is called a projective ambiguity.

#### Euclidean Reconstruction

If the camera calibration is known, an euclidean reconstruction can be obtained from image correspondences. In an euclidean reconstruction the shape of the scene is determined exactly up to a common scale factor. This was desired in the present application. Since the camera calibration is known as described in section 1.3.4, the reconstruction from image correspondences is computed up to an overall scale factor. This reconstruction is also known as similarity reconstruction.

#### True Reconstruction

In a true reconstruction, the original scene is recovered without the ambiguities mentioned before. It can be seen as an upgrade of the euclidean reconstruction. If additionally to the camera calibration a reference length in the scene is known, the scale factor can be determined and the scene can be upgraded to a true reconstruction.

## 1.4 Bundle Adjustment

Bundle Adjustment is a technique that can be used to refine reconstructions of three-dimensional scenes obtained from images. The 3D structure as well as the parameters of the cameras are estimated. The name bundle is related to the bundle of light rays going from a three-dimensional object through the camera centres. These rays are adjusted adapting the object as well as the camera centre. In [63] an introduction on various techniques and algorithms on the field of bundle adjustment is given. In [25] optimisation algorithms for computer vision applications are provided.
In the late $18^{th}$ century and at the beginning of the $19^{th}$ century Gauss and Legendre independently developed the theory of minimising the least squares. These methods where then used for estimation problems in astronomy and geodesy by Gauss (refer to [63], appendix A, for the history of bundle adjustment). Basically this was the basis for modern bundle adjustment techniques. The goal is to minimising a cost function that describes the error of the system. Such a cost function can be the sum of squares of the reprojection errors in the images, that is defined in chapter 3.4. Assuming that the error is normally distributed, the sum of squares is the Maximum Likelihood solution. Bundle Adjustment is often used as a final step in pose estimation processes in computer vision. Bundle adjustment is a very flexible technique. It can not only be used for quadric cost functions. Since

in the problem described in this thesis the quadric model was used, the other models will not be dealt with. In [63] bundle adjustment is described in a more general way. Because of the flexibility of bundle adjustment a lot of minimisation problems can be formulated. To obtain good results, a good initialisation is required. Since the concept is not restricted in the number of parameters to be optimised, it can become a extremely large minimisation problem.
Bundle adjustment is an iterative minimisation problem. Depending on the purpose various algorithms such the Levenberg-Marquardt algorithm or the Gauss newton iteration can be used for the minimisation. In chapter 2 the cameras and the scene consisting of points and lines are estimated. These results are used as initialisation for the bundle adjustment procedure, described in chapter 3. There the algorithm is explained in detail and the mathematical steps for parametrising the points, lines and cameras are pointed out.

# 2 Pose Estimation and Reconstruction

Suppose, an image stream of a scene, recorded by a camera, is provided. If the intrinsic parameters of the camera are known and correspondences across the camera views can be obtained, an euclidean reconstruction of the scene, consisting of the point and line features as well as the camera poses can be computed. If only point correspondences across the images are provided, a reconstruction can be obtained from two images. If point and line correspondences across the images are given, at least three images are required to obtain a reconstruction of the scene. This is the case in the present problem as already stated in chapter 1. From an image stream a reconstruction of the scene, consisting of points, lines as well as cameras is required. This chapter explains a method to compute the trifocal tensor from line and point correspondences across three views. From this tensor the scene consisting of points and lines as well as the three cameras can be reconstructed up to a scale factor. In a next step, the single reconstructions will be concatenated to obtain a large scene that consists of all the points, lines and a large number of cameras.

## 2.1 Pose Estimation

### 2.1.1 The Trifocal Tensor

To estimate the Camera Poses and the structure of the scene, the trifocal tensor $\mathcal{T}$ is used. $\mathcal{T}$ is a $3 \times 3 \times 3$ Matrix and encapsulates the geometric relations between three cameras in the same way as the fundamental matrix does for 2 views.
Hartley and Zisserman in [25], chapter 15 and Hartley in [26] provide a good introduction in the basics of the trifocal tensor. Furthermore several methods for the computation of the tensor and for recovering the scene structure are described. Here only the tasks which are relevant for the present problem are considered.
The trifocal tensor can be obtained from a set of point and line correspondences across three images. From the trifocal tensor, the three camera matrices and the fundamental matrices of every two camera pairs can be recovered. Without any knowledge of the scene and the internal camera calibration, the camera matrices can be recovered up to a perspective ambiguity. With further knowledge about the intrinsic camera parameters an euclidean reconstruction can be obtained. This is the case in the present situation. It can be assumed that the camera is well calibrated and the intrinsic camera parameters are known.
Let the first camera be chosen as $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$. Then a consistent triplet of cameras with the second camera $\mathbf{P}'$ and the third camera $\mathbf{P}''$ can be obtained.
Since $\mathcal{T}$ is a $3 \times 3 \times 3$ Matrix, it has 27 entries. The trifocal tensor consists of 26 independent ratios except for the common scale factor of the matrix.
Even though the tensor has 27 entries with 26 independent ratios and an overall scaling of the matrix, there are only 18 independent degrees of freedom. Each of the three camera matrices has 11 degrees of freedom. From this 33 degrees of freedom the 15 degrees of

freedom for the projective world frame must be subtracted. Therefore the tensor has $33 - 15 = 18$ degrees of freedom. Thus, there are $26 - 18 = 8$ independent algebraic constraints. The trifocal tensor $\mathcal{T}$ is said to be geometrically valid or satisfies all internal constraints if there exist three camera matrices

$$\mathbf{P} = [\mathbf{I} \mid \mathbf{0}], \quad \mathbf{P}' = \left[\mathbf{M}' \mid \mathbf{p}'_4\right] \quad \text{and} \quad \mathbf{P}'' = \left[\mathbf{M}'' \mid \mathbf{p}''_4\right] \tag{2.1}$$

such that $\mathcal{T}$ corresponds to the three camera matrices according to

$$\mathbf{T}_i = \mathbf{m}'_i \mathbf{p}''^{\top}_4 - \mathbf{p}'_4 \mathbf{m}''^{\top}_i. \tag{2.2}$$

$\mathbf{m}'_i$ and $\mathbf{m}''_i$ respectively is the $i^{th}$ column of $\mathbf{M}'$ and $\mathbf{M}''$. The set of three matrices $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$ forms the trifocal tensor $\mathcal{T}$.

### Point and Line Correspondences Across Images

Point and line correspondences provide the set of equations that forms the linear system of equations of the form $\mathbf{At} = \mathbf{0}$. The vector $\mathbf{t}$ is composed of the 27 entries of trifocal tensor $\mathcal{T}$. Equations of point correspondences may be combined with equations of lines as well as with equations of a mixture of point and line correspondences. In table 2.1 the available equations are summarised. The relations are illustrated in figure 2.1.

| **Correspondence** | **Relation** | **Number of Equations** |
|---|---|---|
| point-point-point ($\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$) | $[\mathbf{x}']_\times \left(\sum_i \mathbf{x}(i)\mathbf{T}_i\right) [\mathbf{x}'']_\times = \mathbf{0}_{3\times 3}$ | 4 |
| point-point-line ($\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{l}''$) | $[\mathbf{x}']_\times \left(\sum_i \mathbf{x}(i)\mathbf{T}_i\right) \mathbf{l}'' = \mathbf{0}_{3\times 1}$ | 2 |
| point-line-line ($\mathbf{x} \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$) | $\mathbf{l}'^{\top} \left(\sum_i \mathbf{x}(i)\mathbf{T}_i\right) \mathbf{l}'' = 0$ | 1 |
| line-line-line ($\mathbf{l} \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$) | $\mathbf{l}'^{\top} [\mathbf{T}_1\ \mathbf{T}_2\ \mathbf{T}_3] \mathbf{l}'' = \mathbf{l}^{\top}$ | 2 |

Table 2.1: ***Trifocal tensor relations between point and line correspondences***. *Pure correspondences of points or lines are possible as well as any mixture of point and line correspondences. In the last column the number of linear independent equations are denoted. The table is borrowed from [25], chapter 16.1, page 391 with slightly variations.*

To define the trifocal tensor up to scale, at least 26 independent equations are required. If more than 26 equations are present, a solution is obtained by a linear least-squares minimisation. One minimises $\|\mathbf{At}\|$ subject to the constraint $\|\mathbf{t}\| = 1$.

In the present problem, pure point correspondences $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$ as well as pure line correspondences $\mathbf{l} \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$ in the images are considered. Since a point-point-point correspondence provides 4 independent equations, at least 7 point correspondences are required. On the other hand, a line-line-line correspondence provides two independent equations. Thus, at least 13 line correspondences are necessary. Any mixture of point correspondence and

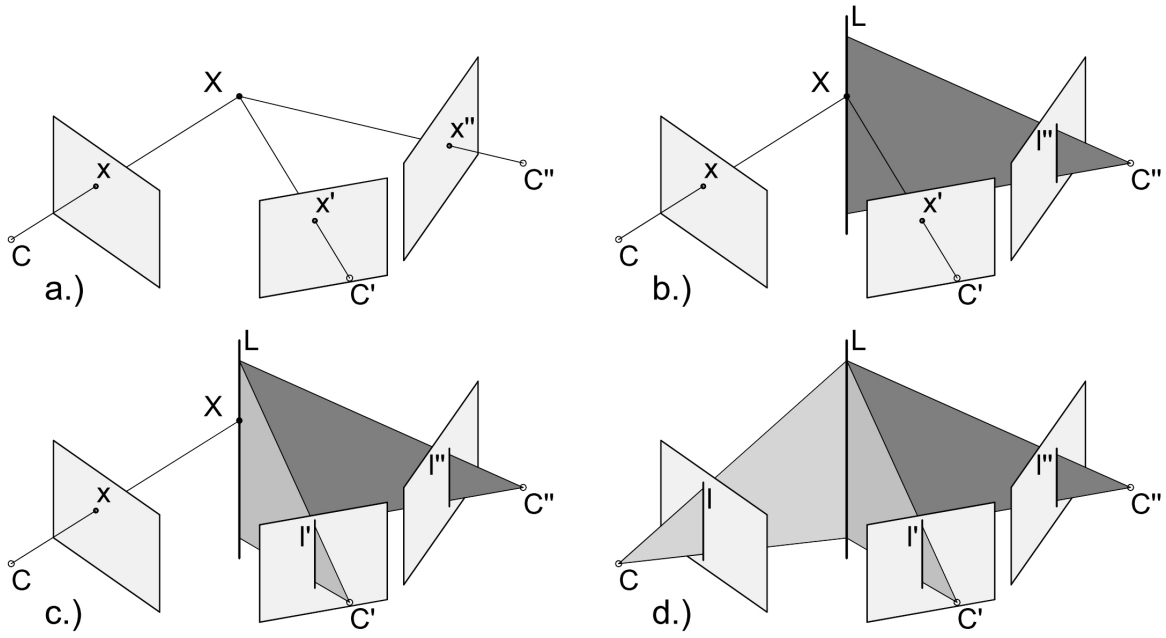



Figure 2.1: ***Point and line correspondences across three views.*** *The space point* $\boldsymbol{X}$ *lies on the space line* $\boldsymbol{L}$*. The cameras are represented by* $\boldsymbol{C}, \boldsymbol{C}', \boldsymbol{C}''$ *that indicates the camera centrers of the three views. (a) point-point-point correspondence (*$\boldsymbol{x} \leftrightarrow \boldsymbol{x}' \leftrightarrow \boldsymbol{x}''$*). (b) point-point-line correspondence (*$\boldsymbol{x} \leftrightarrow \boldsymbol{x}' \leftrightarrow \boldsymbol{l}''$*). (c) point-line-line correspondence (*$\boldsymbol{x} \leftrightarrow \boldsymbol{l}' \leftrightarrow \boldsymbol{l}''$*). (d) line-line-line correspondence (*$\boldsymbol{l} \leftrightarrow \boldsymbol{l}' \leftrightarrow \boldsymbol{l}''$*).*

line correspondence is possible. The only constraint is that at least 26 independent equations are provided.

**Retrieve Epipoles from the Trifocal Tensor**

The projection of the camera center of the first image into the second and third image are the epipoles $\mathbf{e}' = \mathbf{P}'\mathbf{C}$ and $\mathbf{e}'' = \mathbf{P}''\mathbf{C}$, as illustrated in figure 2.2.
Suppose plane $\boldsymbol{\pi}'$, that is the back projection from line $\mathbf{l}'$ is an epipolar plane regarding to the first and the second camera. Thus, it passes trough the camera centrer $\mathbf{C}$ and $\mathbf{C}'$. Let $\mathbf{X}$ be a point in 3D space lying on plane $\boldsymbol{\pi}'$. Then the ray trough point $\mathbf{X}$ and camera centre $\mathbf{C}$ lies in the plane $\boldsymbol{\pi}'$. Therefore the line $\mathbf{l}'$ in the second image is the epipolar line of the point $\mathbf{x}'$, the projection of the world point $\mathbf{X}$ in the second image.
The plane $\boldsymbol{\pi}''$ is the back projection from line $\mathbf{l}'$. This plane intersects the plane $\boldsymbol{\pi}'$ in the 3D line $\mathbf{L}$. The ray through $\mathbf{x}$ and $\mathbf{C}$ lies in the plane $\boldsymbol{\pi}'$ and must intersect the line $\mathbf{L}$. Thus, the ray trough $\mathbf{x}$ and the planes $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$ intersect in the same point. This gives a point-line-line correspondence $\mathbf{x} \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$. This satisfies the equation $\mathbf{l}'^{\top}(\sum_i \mathbf{x}(i)\mathbf{T}_i)\mathbf{l}'' = 0$. Since this is true for every line $\mathbf{l}''$ and $\mathbf{l}'$ respectively, the following relation holds:

$$\mathbf{l}'^{\top}\left(\sum_i \mathbf{x}(i)\mathbf{T}_i\right) = \mathbf{0}^{\top} \quad \text{and} \quad \left(\sum_i \mathbf{x}(i)\mathbf{T}_i\right)\mathbf{l}'' = \mathbf{0} \tag{2.3}$$

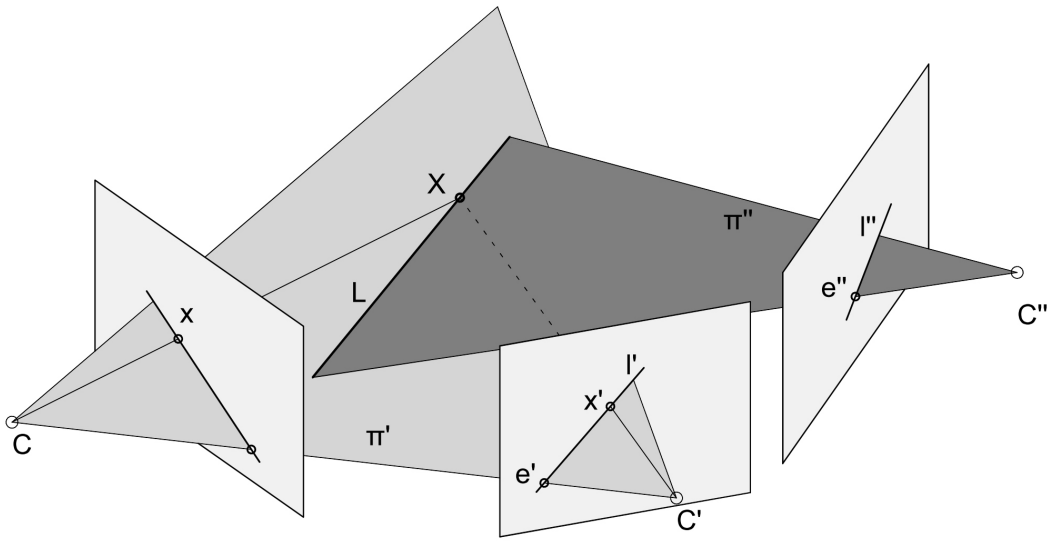



Figure 2.2: ***Epipoles and the trifocal tensor:*** *Scene consisting of three cameras with camera centres **C**, **C**′ and **C**″. It describes the epipolar relations across three views.*

Thus, $\mathbf{l}'^{\top}$ is the left null space of $\sum_i \mathbf{x}(i)\mathbf{T}_i$ and $\mathbf{l}''$ is the right null space of $\sum_i \mathbf{x}(i)\mathbf{T}_i$.
If the point $\mathbf{x}$ changes, the epipolar lines $\mathbf{l}'$,$\mathbf{l}''$ changes similarly. The epipoles $\mathbf{e}$,$\mathbf{e}'$ however remains unchanged because the epipolar lines in one image intersect in the same point, the epipole.
Consider now the three different choices $(1,0,0)^{\top}$, $(0,1,0)^{\top}$ and $(0,0,1)^{\top}$ of the homogeneous point $\mathbf{x}$. The equation $\sum_i \mathbf{x}(i)\mathbf{T}_i$ is equal to $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$ for the three different choices of $\mathbf{x}$. Since the epipole $\mathbf{e}'$ is the common intersection of all epipolar lines $\mathbf{l}'_i$, it can be represented as the common intersection of the left null-vectors of the matrices $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$.
In the same way, the epipole $\mathbf{e}''$ is the common intersection of the right null-spaces of the matrices $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$. Thus, the epipoles can be computed as the null vectors of the $3 \times 3$ matrices composed of the epipolar lines corresponding to the three choices of $\mathbf{x}$.

$$\mathbf{e}'^{\top}\left[\mathbf{l}'_1, \mathbf{l}'_2, \mathbf{l}'_3\right] = \mathbf{0} \quad \text{and} \quad \mathbf{e}''^{\top}\left[\mathbf{l}''_1, \mathbf{l}''_2, \mathbf{l}''_3\right] = \mathbf{0} \tag{2.4}$$

**Retrieve Fundamental Matrices from the Trifocal Tensor**

From the trifocal tensor the fundamental matrices between the first and the other views can be obtained easily. In [25] the following equations are provided:

$$\mathbf{F}_{21} = [\mathbf{e}']_{\times}\left[\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\right]\mathbf{e}'' \tag{2.5}$$

$$\mathbf{F}_{31} = [\mathbf{e}'']_{\times}\left[\mathbf{T}_1^{\top}, \mathbf{T}_2^{\top}, \mathbf{T}_3^{\top}\right]\mathbf{e}' \tag{2.6}$$

The epipoles can be obtained according to equation 2.4. $[\mathbf{e}']_{\times}$ and $[\mathbf{e}'']_{\times}$ respectively are the skew symmetric matrices corresponding to $\mathbf{e}'$ and $\mathbf{e}''$.
The fundamental Matrix $\mathbf{F}_{21}$ satisfies the relation $\mathbf{x}^{\top}\mathbf{F}_{21}\mathbf{x}'$ for the point correspondence $\mathbf{x} \leftrightarrow \mathbf{x}'$. The similar coherence is valid for $\mathbf{F}_{31}$.

#### Retrieve Camera Matrices from the Trifocal Tensor

Up to a perspective ambiguity, a consistent triplet of camera matrices $\mathbf{P}, \mathbf{P}', \mathbf{P}''$ can be obtained directly by the trifocal tensor. The first camera is defined as $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$. Then the second camera can be computed according to

$$\mathbf{P}' = \left[ [\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3]\, \mathbf{e}'' \mid \mathbf{e}' \right]. \tag{2.7}$$

The camera pair $\{\mathbf{P}, \mathbf{P}'\}$ now is consistent with the fundamental matrix $\mathbf{F}_{21}$. Choosing the third camera in the same way building a camera pair $\{\mathbf{P}, \mathbf{P}''\}$ consistent with the fundamental matrix $\mathbf{F}_{31}$, gives an inconsistent camera triplet $\{\mathbf{P}, \mathbf{P}', \mathbf{P}''\}$, because the two camera pairs do no necessarily define the same projective world frame. A third camera, that is consistent to the camera pair $\{\mathbf{P}, \mathbf{P}'\}$ can be obtained according to

$$\mathbf{P}'' = \left[ (\mathbf{e}''\mathbf{e}''^{\top} - \mathbf{I}) \left[ \mathbf{T}_1^{\top}, \mathbf{T}_2^{\top}, \mathbf{T}_3^{\top} \right] \mathbf{e}' \mid \mathbf{e}'' \right]. \tag{2.8}$$

### 2.1.2 Computation of the Trifocal Tensor from Three Images

There are several methods for computing the trifocal tensor. The following algorithms are described and evaluated by Hartley and Zisserman in [25], chapter 16.
Minimising the geometric distance with the maximum Likelihood solution is the *Gold Standard procedure* and provides the best results. Another iterative method where the geometric distance is minimised is the *Sampson geometric approximation*, which is a good approximation to the gold standard method. Since both methods are iterative, they can poorly be applied in a real time procedure. The *normalised linear algorithm* is a linear algorithm that computes the trifocal tensor directly from a set of linear equations. This algorithm does not enforce the internal constraints of the trifocal tensor. Nevertheless it can be used to initialise the other algorithms that enforce the internal constraints. The *algebraic minimisation algorithm* will be initialised with the *normalised algorithm*. Then it corrects the trifocal tensor such that it corresponds to a geometric configuration. A geometrically valid tensor can be found non-iteratively, this tensor can be optimised iteratively with a Levenberg-Marquardt algorithm. This algorithm, that is used here to determine the trifocal tensor $\mathcal{T}$ is described next.

#### The algebraic minimisation algorithm

The algebraic minimisation algorithm consists of two phases. In a first step a tensor is computed with a linear algorithm. This tensor does not necessarily satisfies the internal constraints. In a second step, based on the former result, a geometrically valid tensor is computed. This tensor then will be optimised iteratively.
From the equations in table 2.1 one can form the linear system of equations $\mathbf{At} = \mathbf{0}$. Each correspondence provides a number of equations, but only a subset of them are linearly independent. For instance the point-point-point correspondence has a $3 \times 3$ null space, thus, 9 equations are provided. From those equations only four are linearly independent. Thus, only the four linearly independent equations must be considered. The line-line-line correspondence has a $3 \times 1$ null space. From these three equations only two are linearly independent.

**Point Correspondences:** From the equation for point-point-point correspondences of table 2.1 one can obtain four linearly independent equations. These equations are of the form

$$\mathbf{x}(k)\left(\mathbf{x}'(i)\mathbf{x}''(l)\mathbf{T}_k(3,3) - \mathbf{x}''(l)\mathbf{T}_k(i,3) - \mathbf{x}'(i)\mathbf{T}_k(3,l) + \mathbf{T}_k(i,l)\right) = 0 \tag{2.9}$$

for $i, l = 1, 2$ and $k = 1 - 3$ for each combination of $i$ and $l$. Each of the four equations, evaluated for $k = 1 - 3$ leads to one equation of the linear system of equations $\mathbf{At} = \mathbf{0}$.
$\mathbf{x}(k)$ is the $k^{th}$ element of the homogeneous 3 vector $\mathbf{x}$ that is a point coordinate in the image. The image frame is specified by the number of primes. $\mathbf{T}_k(i,l)$ is the element in row $i$, column $l$ of matrix $\mathbf{T}_k$. The point-point-point correspondence is visualised in figure 2.3.

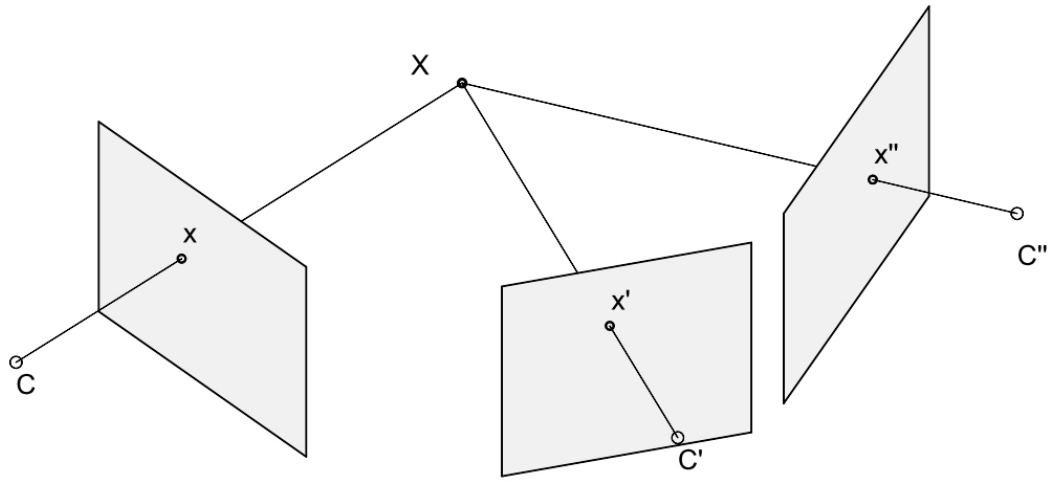



Figure 2.3: ***Point correspondence across three views***

**Line Correspondences:** A line $\mathbf{l} = (a, b, c)^\top$ in the euclidean 2-space is represented as a vector with 3 elements. This vector represents the line $ax + by + c = 0$.
Suppose, $\mathbf{l}_1 = (0.01, 0, 1)$ and $\mathbf{l}_2 = (0, 0.01, 1)$. These two vectors are algebraically very similar, since $\|\mathbf{l}_1 - \mathbf{l}_2\|$ is small. Geometrically these two vectors represent two completely different lines, particularly $\mathbf{l}_1$ represents the vertical line $x = 1$, the line $\mathbf{l}_2$ represents the horizontal line $y = 1$. Thus, an appropriate scaling for line vector is necessary. If the lines are normalised such that $\|(a, b)^\top\| = 1$ then the lines $\mathbf{l}_1, \mathbf{l}_2$ become $\mathbf{l}_1 = (1, 0, 100)$ and $\mathbf{l}_2 = (0, 1, 100)$. Those two line vectors, that geometrically represent the same lines as before the scaling, now algebraically are very different. Thus, when dealing with lines it is recommended to be careful in the representation.
If a line-line-line correspondence $\mathbf{l} \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$ is given, one can substitute the line $\mathbf{l}$ by two points $\mathbf{x}_1, \mathbf{x}_2$ lying on the line $\mathbf{l}$. This is shown in figure 2.4. Thus, the correspondences $\mathbf{x}_1 \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$ and $\mathbf{x}_2 \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$ arise.

Table 2.1 provides an equation for such a point-line-line correspondence. From each for those two point-line-line correspondences one linearly independent equation can be ob-

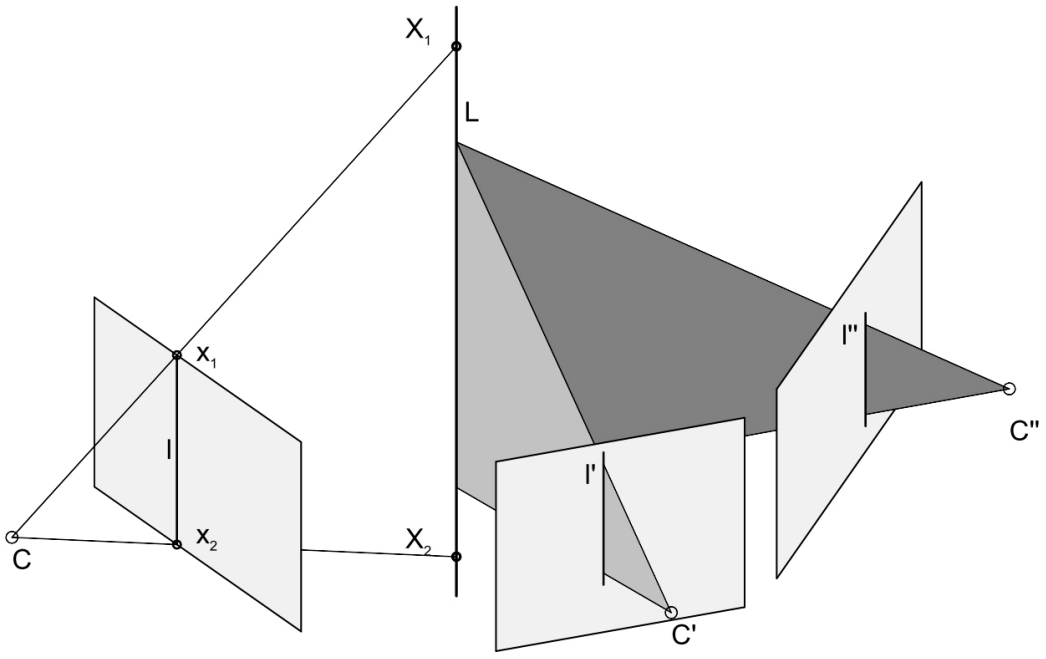



Figure 2.4: ***Line correspondence across three views:*** *The line in the first image is substituted by the two points along the line, that intersect the image border.*

tained. The obtained equation has the form

$$\begin{aligned} \mathbf{l'}^{\top} \left(\mathbf{x}_1(1)\mathbf{T}_1 + \mathbf{x}_1(2)\mathbf{T}_2 + \mathbf{x}_1(3)\mathbf{T}_3\right) \mathbf{l''} = 0 \\ \mathbf{l'}^{\top} \left(\mathbf{x}_2(1)\mathbf{T}_1 + \mathbf{x}_2(2)\mathbf{T}_2 + \mathbf{x}_2(3)\mathbf{T}_3\right) \mathbf{l''} = 0 \end{aligned} \tag{2.10}$$

where $\mathbf{x}_i(j)$ is the $j^{th}$ element of the homogeneous 3 vector $\mathbf{x}_i$.
Since the lines are considered as infinite lines without endpoints, one must make an appropriate choice of the points $\mathbf{x}_1$, $\mathbf{x}_2$ on the line $\mathbf{l}$ in the first image. For the present application, the intersection points of line $\mathbf{l}$ with the image borders where selected.
In this manner, the use of lines in the first image can be avoided. In the second and third image, the lines must still be used, because no corresponding points along the lines through the images are known.

**Data normalisation:** Before solving the linear system of equations $\mathbf{At} = \mathbf{0}$ to obtain the trifocal tensor $\mathcal{T}$, the input data of points and lines must be normalised. Thus, a transformation is applied to the points and lines. The transformation, that is applied to each image, consists of a translation and a scaling of the data. The translation transforms the image such that the centroid of the points lies at the origin. The scaling transforms the entities in the image such that the average distance of the points to the centroid is $\sqrt{2}$. For the normalisation, lines are represented by two points. Here the intersection points with the image borders are considered.

Therefore three $3 \times 3$ transformation matrices $\mathbf{H}, \mathbf{H}', \mathbf{H}''$ of the form

$$\mathbf{H} = \begin{bmatrix} s & 0 & t_x s \\ 0 & s & t_y s \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

are obtained, one for each image. $s$ stands for the scaling factor, $t_x$,$t_y$ represent the translations in $x$ and $y$ direction. These matrices are applied to the points and lines according to $\hat{\mathbf{x}} = \mathbf{H}\mathbf{x}$ and $\hat{\mathbf{l}} = \mathbf{H}^{-\top}\mathbf{l}$ for each image. $\hat{\mathbf{x}}$ and $\hat{\mathbf{l}}$ stands for the normalised points and lines respectively.
In a final step, after the normalised trifocal tensor $\hat{\mathcal{T}}$ has been computed, the trifocal tensor $\mathcal{T}$ can be obtained by denormalisation according to

$$\mathbf{T}_i = \mathbf{H}'^{-1} \left( \sum_{j=1} \mathbf{H}^{\top}(i,j) \hat{\mathbf{T}}_j \right) \mathbf{H}''^{-\top}. \tag{2.12}$$

$\mathbf{T}_i$ is the $i^{th}$ submatrix of the tensor $\mathcal{T}$, $\hat{\mathbf{T}}_j$ is the $j^{th}$ submatrix of the tensor $\hat{\mathcal{T}}$.

**Solving for the tensor:** From equation 2.9 for point correspondences and equation 2.10 for lines correspondences one can obtain a set of equations of the form $\mathbf{At} = \mathbf{0}$. Since it is recommended to consider a great number of correspondences, the system ordinarily is overdetermined. The simple solution $\mathbf{t} = \mathbf{0}$ does not suffice. A non-zero solution of $\mathbf{t}$ for the set of equations is required. In general there is no exact solution. Thus, a least squares solution of $\mathbf{t}$ is wanted that minimises $\|\mathbf{At}\|$ subject to the constraint $\|\mathbf{t}\| = 1$. A solution can be found by the Singular Value Decomposition (SVD). The matrix $\mathbf{A}$ can be decomposed into $\mathbf{A} = \mathbf{UDV}^{\top}$. Then $\mathbf{t}$ is the last column of $\mathbf{V}$. This method is described in the appendix A5.3 of [25], page 593.
Up to here a not necessarily geometrical valid tensor is computed. In the next steps a tensor will be found that satisfies the internal constraints.

**Retrieving the epipoles:** As illustrated in figure 2.2, $\mathbf{e}'$ and $\mathbf{e}''$ are the epipoles in the second and third image of the first camera centre. The epipoles $\mathbf{e}'$ and $\mathbf{e}''$ respectively are the common perpendicular of the left and right null space of the matrices $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$, as indicated in equation 2.4.
With noisy data, no exact solution can be obtained. Thus, the epipoles can be computed from the trifocal tensor using the SVD. Algorithm 2.1 explains the procedure to determine the epipoles $\mathbf{e}''$ and $\mathbf{e}'$ .

**Algebraic minimisation:** After having computed the epipoles $\mathbf{e}'$,$\mathbf{e}''$, the remaining elements of the camera matrices $\mathbf{P}'$ and $\mathbf{P}''$ must be determined. From the camera matrices the geometrically valid tensor can be computed according to equation 2.2. From the equations 2.7 and 2.8 it can be seen that $\mathbf{e}' = \mathbf{p}'_4$ and $\mathbf{e}'' = \mathbf{p}''_4$. With the known epipoles computed in the previous steps, the trifocal tensor can be expressed with respect to the remaining entries of the camera matrices $\mathbf{P}'$ and $\mathbf{P}''$, denoted by $\mathbf{M}'$ and $\mathbf{M}''$ according to the equations 2.1 and 2.2. This coherence can be reformulated as the linear system of equations $\mathbf{t} = \mathbf{Ea}$ where

**Algorithm summary:**
The algorithm describes the computation of the epipole $\mathbf{e}''$ from from the trifocal tensor $\mathcal{T}$, composed of the matrices $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$. In the same way the epipole $\mathbf{e}'$ can be obtained using $\mathbf{T}_i^\top$ instead of $\mathbf{T}_i$. Thus, in the first step the epipolar lines $\mathbf{l}_i'$ are computed. In the second step the epipole $\mathbf{e}'$ is obtained.

**Algorithm:**

1. Minimise the equation $\left\|\mathbf{T}_i\mathbf{l}_i''\right\|$ for $i = 1, 2, 3$. Let $\mathbf{T}_i = \mathbf{UDV}^\top$ be the SVD of $\mathbf{T}_i$. Then $\mathbf{l}_i''$ is the last column of the matrix $\mathbf{V}$.
2. Determine $\mathbf{e}''$ by minimising the equation $\|\mathbf{We}''\|$ where $\mathbf{W} = [\mathbf{l}_1'', \mathbf{l}_2'', \mathbf{l}_3'']$. Let $\mathbf{W} = \mathbf{UDV}^\top$ be the SVD of $\mathbf{W}$. The epipole $\mathbf{e}''$ is then the last column of $\mathbf{V}$.

**Algorithm 2.1:** ***Computation of the epipoles:*** *Algorithm to compute the epipoles from the trifocal tensor. This algorithm is a repeated utilisation of algorithm A5.4, described in [25], appendix A5.3.*

$\mathbf{t}$ is the vector of elements of the trifocal tensor, $\mathbf{a}$ is a vector that contains the remaining elements of the matrices $\mathbf{M}'$ and $\mathbf{M}''$. The matrix $\mathbf{E}$ contains the entries of $\mathbf{e}'$ and $\mathbf{e}''$.
The set of equations that is said to be minimised is the algebraic error $\|\mathbf{At}\|$, subject to the constraint $\|\mathbf{t}\| = 1$. Since $\mathbf{t} = \mathbf{Ea}$ the problem can be reformulated as $\|\mathbf{At}\| = \|\mathbf{AEa}\|$ with the constraint $\|\mathbf{Ea}\| = 1$. This minimisation problem can be solved according to algorithm 2.2. Thus, a geometrically valid tensor is obtained.

**Iterative optimisation** For computing the geometrically valid tensor, the epipoles $\mathbf{e}'$ and $\mathbf{e}''$ where used. Those epipoles where determined from the estimation of the tensor, that does not satisfy the internal constraints. Thus, the epipoles may be inaccurate. Since the geometrically valid tensor $\mathcal{T}$ is computed based on inaccurate epipoles, $\mathcal{T}$ does not describe exactly the relations of the camera triplet $\{\mathbf{P}, \mathbf{P}', \mathbf{P}''\}$. Finding the optimal epipoles, $\mathcal{T}$ can be improved. For the optimisation one can use the Levenberg-Marquardt algorithm to optimise the estimated trifocal tensor $\mathcal{T}$. The procedure is outlined in algorithm 2.3.
The mapping $(\mathbf{e}', \mathbf{e}'') \mapsto \mathbf{AEa}$ is a mapping $\mathbb{R}^6 \mapsto \mathbb{R}^{27}$. In the iteration procedure 6 parameters, that are the homogeneous coordinates of the epipoles, are involved. This is a relatively small number compared to the gold standard algorithm, where the camera parameters of all three cameras as well as the coordinates of all the points and lines must be estimated.

The steps for computing the trifocal tensor are summarised in algorithm 2.4.

### 2.1.3 Extracting Camera Matrices from the Trifocal Tensor

From the trifocal tensor $\mathcal{T}$, the three camera matrices $\mathbf{P}$,$\mathbf{P}'$ and $\mathbf{P}''$ can be obtained. Without any knowledge about the camera calibration or the scene, the camera matrices can be recovered only up to a projective ambiguity. In the present case, the intrinsic camera parameters of the cameras are known. Thus, an euclidean reconstruction of the three cameras can be obtained. A true reconstruction can be obtained only with any knowledge about the scene or the motion of the camera. This however is not relevant for the present application.

**Algorithm summary:**
The algorithm finds the vector $\mathbf{t}$ that minimises $\|\mathbf{At}\|$ subject to the constraint $\|\mathbf{t}\| = 1$. $\mathbf{t} = \mathbf{Ea}$, where $\mathbf{E}$ has rank $r$. $\mathbf{t}$ is a vector that contains the 27 elements of the trifocal tensor $\mathcal{T}$, the matrix $\mathbf{A}$ is obtained from the equations in table 2.2 that form the linear system of equations $\mathbf{At} = \mathbf{0}$, the matrix $\mathbf{E}$ is composed of the epipoles $\mathbf{e}'$ and $\mathbf{e}''$ and the vector $\mathbf{a}$ contains the remaining elements of the camera matrices $\mathbf{P}'$ and $\mathbf{P}''$ according to equation 2.1.

**Algorithm:**

1. Compute the SVD of $\mathbf{E} = \mathbf{UDV}^\top$, where the non-zero values of $\mathbf{D}$ appear first down the diagonal.
2. Let $\mathbf{U}'$ be the matrix comprising the first $r$ columns of $\mathbf{U}$.
3. Find the unit vector $\mathbf{t}'$ that minimises $\left\|\mathbf{AU}'\mathbf{t}'\right\|$. Let $\mathbf{AU}' = \mathbf{XYZ}^\top$ be the SVD of $\mathbf{AU}'$, then $\mathbf{t}'$ is the last column of $\mathbf{Z}$.
4. The required solution is $\mathbf{t} = \mathbf{U}'\mathbf{t}'$. Thus, the trifocal tensor $\mathcal{T}$ can be composed of the elements of vector $\mathbf{t}$.
5. For the iterative refinement of the trifocal tensor the vector $\mathbf{a}$ is required. It can be computed $\mathbf{a}$ as $\mathbf{V}'\mathbf{D}'^{-1}\mathbf{t}'$, where $\mathbf{V}'$ consists of the first $r$ columns of $\mathbf{V}$ and $\mathbf{D}'$ is the upper $r \times r$ block of $\mathbf{D}$.

**Algorithm 2.2:** ***Computation of the trifocal tensor:*** *Algorithm to compute the trifocal tensor from three images. This algorithm is borrowed from [25], appendix 5.4.1, algorithm A5.6.*

**Determining Essential Matrices:** From the trifocal tensor, the fundamental matrices $\mathbf{F}_{21}$ and $\mathbf{F}_{31}$ can be recovered according to equation 2.5 and 2.6. From those fundamental matrices, the essential matrices can be obtained. Suppose, $\mathbf{K} = \mathbf{K}_1 = \mathbf{K}_2 = \mathbf{K}_3$ is the camera calibration matrix equal for all three camera matrices, that contains the intrinsic camera parameter. Then the essential matrices can be computed according to

$$\mathbf{E}_{21} = \mathbf{K}^\top \mathbf{F}_{21} \mathbf{K} \tag{2.13}$$

$$\mathbf{E}_{31} = \mathbf{K}^\top \mathbf{F}_{31} \mathbf{K}. \tag{2.14}$$

H.C. Longuet-Higgins [36] was the first who presented the essential matrix in the context of computer vision. Hatley and Zissermann in [25] and J.Philip in [45] provide a good introduction in the essential matrix. Huang and Faugeras in [29] give a deeper look in the role of the fundamental matrix in determining 3D structure from two views. The following explanations and equations are mostly obtained form [25], chapter 9.6, page 257 ff.
The essential matrix is a special case of the fundamental matrix for the case of known camera calibration. Compared to the fundamental matrix it has some additional properties.
Suppose, $\mathbf{x} = \mathbf{PX}$ is the projection of a point $\mathbf{X}$ from the three-dimensional space to the image space of the camera $\mathbf{P}$. Let $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]$ be the decomposed camera matrix. If the calibration matrix $\mathbf{K}$ is known, it can be removed from the camera matrix to obtain the normalised camera matrix according to $\mathbf{K}^{-1}\mathbf{P} = [\mathbf{R} \mid \mathbf{t}]$. Thus, $\hat{\mathbf{x}} = [\mathbf{R} \mid \mathbf{t}]\mathbf{X}$ is the image point expressed in normalised coordinates.
Assume now two normalised camera frames $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}' = [\mathbf{R} \mid \mathbf{t}]$ are given. Then the essential matrix for the two camera frames is

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}. \tag{2.15}$$

**Algorithm summary:**
Iterative optimisation of the trifocal tensor finding the optimal epipoles $\mathbf{e}'$ and $\mathbf{e}''$ with the Levenberg-Marquardt algorithm.

**Algorithm:**

1. Compute the error vector $\boldsymbol{\epsilon}_0 = \mathbf{At} = \mathbf{AEa}$.
2. Compute the Jacobian $\mathbf{J} = \mathbf{A}\frac{\partial \mathbf{t}}{\partial(\mathbf{e}',\mathbf{e}'')} = \mathbf{A}\frac{\partial \mathbf{E}}{\partial(\mathbf{e}',\mathbf{e}'')}\mathbf{a}$.
3. Compute the $6 \times 1$ delta vector $\boldsymbol{\delta} = -(\mathbf{J}^\top\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^\top\boldsymbol{\epsilon}$ that contains the correction factors for the epipoles. $\lambda$ is initialised at the beginning with $10^{-3}$ times the average of the diagonal elements of $\mathbf{J}^\top\mathbf{J}$.
4. Compute the corrected epipoles: $(\mathbf{e}'_1, \mathbf{e}''_1)^\top = (\mathbf{e}'_0, \mathbf{e}''_0)^\top + \boldsymbol{\delta}$.
5. Construct the new matrix $\mathbf{E}$ from the epipoles.
6. Compute the new error vector $\boldsymbol{\epsilon}_1 = \mathbf{AEa}$.
7. If the new error vector is greater than the old one, increase $\lambda$ by the factor 10 and go to the step 3. Else decrease $\lambda$ by the factor 10, update the error vector and the epipoles and go to the step 2. Repeat until the algorithm converges.

**Algorithm 2.3:** ***Iterative optimisation of the trifocal tensor*** *Levenberg-Marquardt implementation to improve the trifocal tensor. The Levenberg-Marquardt algorithm is borrowed from [25], appendix A6.2, page 600.*

The essential matrix has 5 degrees of freedom. The roation $\mathbf{R}$ as well as the translation $\mathbf{t}$ has 3 degrees of freedom, but since the essential matrix is a homogeneous quantity, there is an overall scale ambiguity. In contrast the fundamental matrix has 7 degrees of freedom. Because of the reduced number of degrees of freedom of the essential matrix, additional internal constraints has to be satisfied related to the fundamental matrix. The fundamental matrix is a homogenous matrix that satisfies the singularity constraint. Thus, its determinant has to be equal to zero. A way of ensuring this is to use the SVD and setting the smallest singualr value equal to zero. The essential matrix however satisfies the constraint that two singular values are equal and the third is zero. Suppose $\mathbf{E} = \mathbf{UDV}^\top$ is the singular value decomposition of the essential matrix. The diagonal matrix $\mathbf{D}$ contains the singular values of $\mathbf{E}$. Thus, the constraint can be satisfied according to

$$\mathbf{E} = \mathbf{U}\begin{bmatrix} \frac{d_1+d_2}{2} & 0 & 0 \\ 0 & \frac{d_1+d_2}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{V} \tag{2.16}$$

where $d_1$ and $d_2$ are the first and second element of the diagonal matrix $\mathbf{D}$ that contains the singular values of the essential matrix. In this way, the essential matrices $\mathbf{E}_1$ and $\mathbf{E}_2$ may be obtained.

**Recovering a Camera Pair from the Essential Matrix:** Once the essential matrices $\mathbf{E}_{21}$ and $\mathbf{E}_{31}$ that satisfy the internal constraints are computed, two pairs of normalised camera

**Algorithm summary:**
Computation of a geometrically valid Trifocal tensor $\mathcal{T}$ from point and line correspondences across three images.

**Algorithm:**

1. normalise the image entities finding the matrices $\mathbf{H}$,$\mathbf{H}'$ and $\mathbf{H}''$ and applying them to all the points and lines in the three images according to $\hat{\mathbf{x}} = \mathbf{H}\mathbf{x}$ and $\hat{\mathbf{l}} = \mathbf{H}^{-\top}\mathbf{l}$. Transform points and lines in the second and third image in the same way.
2. Form the set of equations $\mathbf{A}\mathbf{t} = \mathbf{0}$ from point and line correspondences.
3. Solve the linear system of equations for the tensor $\hat{\mathcal{T}}$ by minimising $\|\mathbf{A}\mathbf{t}\|$ subject to the constraint $\|\mathbf{t} = 1\|$. This tensor is not necessarily geometrically valid.
4. Find the epipoles $\mathbf{e}'$ and $\mathbf{e}''$ as the common intersection of the left respectively right null-space of the matrices $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$.
5. construct the matrix $\mathbf{E}$ such that $\mathbf{t} = \mathbf{E}\mathbf{a}$. $\mathbf{t}$ is the vector containing the elements of $\mathcal{T}$, $\mathbf{a}$ is the vector containing the remaining elements of the camera matrices $\mathbf{P}'$ and $\mathbf{P}''$ according to equation 2.1. $\mathbf{E}$, that is composed of the epipoles $\mathbf{e}'$ and $\mathbf{e}''$ respectively expresses the linear relationship of equation 2.2.
6. Minimise the expression $\|\mathbf{A}\mathbf{E}\mathbf{a}\|$ subject to the constraint $\|\mathbf{E}\mathbf{a}\| = 1$ using algorithm 2.2.
7. Perform the iterative optimisation of the trifocal tensor $\mathcal{T}$ according to algorithm 2.3.
8. Since the data where normalised at the beginning, the tensor must be denormalised such that it corresponds to the original data. Thus, $\mathcal{T}$ is computed according to $\mathbf{T}_i = \mathbf{H}'^{-1}\left(\sum_{j=1} \mathbf{H}^{\top}(i,j)\hat{\mathbf{T}}_j\right)\mathbf{H}''^{-\top}$ for $i = 1, 2, 3$. The set of the three matrices $\{\mathbf{T}_i\}$ forms the tensor $\mathcal{T}$.

**Algorithm 2.4:** ***algebraic minimisation algorithm*** *The Algorithm finds a geometrically valid tensor that minimises the algebraic error. The last iteration step can be dropped to provide a non iterative algorithm that finds a geometrically valid tensor $\mathcal{T}$. The algorithm is composed of two algorithms borrowed from [25], namely algorithm 16.1, page 394 and algorithm 16.2, page 396.*

matrices $\{\mathbf{P}, \mathbf{P}'\}$ and $\{\mathbf{P}, \mathbf{P}''\}$ can be obtained. The following procedure describes the computation of the camera pair $\{\mathbf{P}, \mathbf{P}'\}$. The camera pair $\{\mathbf{P}, \mathbf{P}''\}$ can be obtained in a similar way.
The first camera is defined as $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$. The second camera $\mathbf{P}'$ can be expressed as $\mathbf{P}' = [\mathbf{R} \mid \mathbf{t}]$. The same holds also for the second camera pair $\{\mathbf{P}, \mathbf{P}''\}$. The retrieving of the camera matrices from the essential matrix follows the advices from [25], chapter 9.6.2 and 9.6.3 on pages 258 ff.
The camera matrices can be recovered from the essential matrix up to a scale factor. Four solution will be found. In a further step the correct solution can be selected.
Suppose, $\mathbf{E} = [\mathbf{t}]_\times\mathbf{R} = \mathbf{S}\mathbf{R}$ where $\mathbf{S}$ is a skew symmetric matrix and the matrix $\mathbf{R}$ is orthonormal. Consider the two matrices

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.17}$$

The matrix **W** is orthonormal and the matrix **Z** is skew symmetric. According to the block decomposition of a skew symmetric matrix in [25], appendix A4.2, page 580, the matrix **S** may be written as $\mathbf{S} = \mathbf{U}\mathbf{Z}\mathbf{U}^\top$, where **U** is orthogonal. Since $\mathbf{Z} = d \cdot diag(1,1,0)\mathbf{W}$ with $d$ as arbitrary scale factor, $\mathbf{S} = \mathbf{U}\,diag(1,1,0)\,\mathbf{W}\mathbf{U}^\top$. Thus, $\mathbf{E} = \mathbf{S}\mathbf{R} = \mathbf{U}\,diag(1,1,0)\,\left(\mathbf{W}\mathbf{U}^\top\mathbf{R}\right)$ is a SVD of **E** with two singular values equal and one equal to zero.
Let $\mathbf{E} = \mathbf{U}\,diag(1,1,0)\,\mathbf{V}^\top$ be the SVD of the essential matrix. Ignoring the signs, there are two possible factorisations for $\mathbf{E} = \mathbf{S}\mathbf{R}$.

$$\mathbf{S} = \mathbf{U}\mathbf{Z}\mathbf{U}^\top \tag{2.18}$$

$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^\top \quad \text{or} \quad \mathbf{R} = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \tag{2.19}$$

From these factorisations, the vector **t** of camera matrix $\mathbf{P}'$ can be determined from **S**, because $\mathbf{S} = [\mathbf{t}]_\times$. Thus, $\mathbf{S}\mathbf{t} = \mathbf{0}$. Since **U** is an orthonormal matrix, $\mathbf{U}\mathbf{U}^\top = \mathbf{I}$ gives the identity matrix. Thus, $\mathbf{t} = \mathbf{U}(0,0,1)^\top = \mathbf{u}_3$, where $\mathbf{u}_3$ is the third column of matrix **U**. Since the essential matrix is defined up to an arbitrary scale, the sign of **t** is unknown. From equation 2.19 also two possibilities for the rotational part **R** had been determined. This leads to four possible choices of $\mathbf{P}'$.

$$\begin{aligned}
\mathbf{P}' &= [\mathbf{U}\mathbf{W}\mathbf{V}^\top \mid +\mathbf{u}_3] \quad \text{or} \\
\mathbf{P}' &= [\mathbf{U}\mathbf{W}\mathbf{V}^\top \mid -\mathbf{u}_3] \quad \text{or} \\
\mathbf{P}' &= [\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \mid +\mathbf{u}_3] \quad \text{or} \\
\mathbf{P}' &= [\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top \mid -\mathbf{u}_3]
\end{aligned} \tag{2.20}$$

Figure 2.5 illustrates the geometrical meaning of the four possible solutions.
The correct solution is those, in which the points lie in front of both cameras. It is sufficient to perform the test for one single point correspondence for all four solutions. In one of the four configurations the point would lie in front of both cameras **P** and $\mathbf{P}'$, this solution will be selected. Thus, it is necessary to reconstruct the selected point **X** from a point correspondence $\mathbf{x} \Leftrightarrow \mathbf{x}'$ for each configuration as stated in chapter 2.2.1. This leas to four points $\mathbf{X}_i$ for $i = 1...4$, one for each configuration. Now a depth test must be performed for each system to test whether the point lies in front of or behind the camera.
The depth test is performed according to result 6.1 of [25] on page 162.

$$d(\mathbf{X},\mathbf{P}) = \frac{\sigma(det(\mathbf{R}))w}{T\,\|\mathbf{r}^3\|} \tag{2.21}$$

$d(\mathbf{X},\mathbf{P})$ is the depth of point **X** in frame $\mathbf{P} = [\mathbf{R} \mid \mathbf{t}]$, the expression $\sigma(det(\mathbf{R}))$ is the sign of the argument $det(\mathbf{R})$, that is $+1$ for a positive determinant and $-1$ for a negative one. $w$ is the $3^{rd}$ element of vector of **x** that is determined according to $\mathbf{x} = \mathbf{P}\mathbf{X}$, $T$ is the fourth element of the homogeneous space point vector **X**. Finally, $\mathbf{r}^3$ is the $3^{rd}$ row of the matrix **R**. For right hand frames, that are present here, the depth test leads to a positive result if the point $X$ lies in front of the camera and to a negative result if the point lies behind the camera. Thus, the correct solution for the camera pair can be obtained.
By this means an euclidean reconstruction of the camera pairs $\{\mathbf{P} = [\mathbf{I} \mid \mathbf{0}], \mathbf{P}'\}$ and $\{\mathbf{P} = [\mathbf{I} \mid \mathbf{0}], \mathbf{P}''\}$ can be obtained. Since each reconstruction is obtained up to a scale factor, this is not a geometrically valid camera triplet $\{\mathbf{P} = [\mathbf{I} \mid \mathbf{0}], \mathbf{P}', \mathbf{P}''\}$.

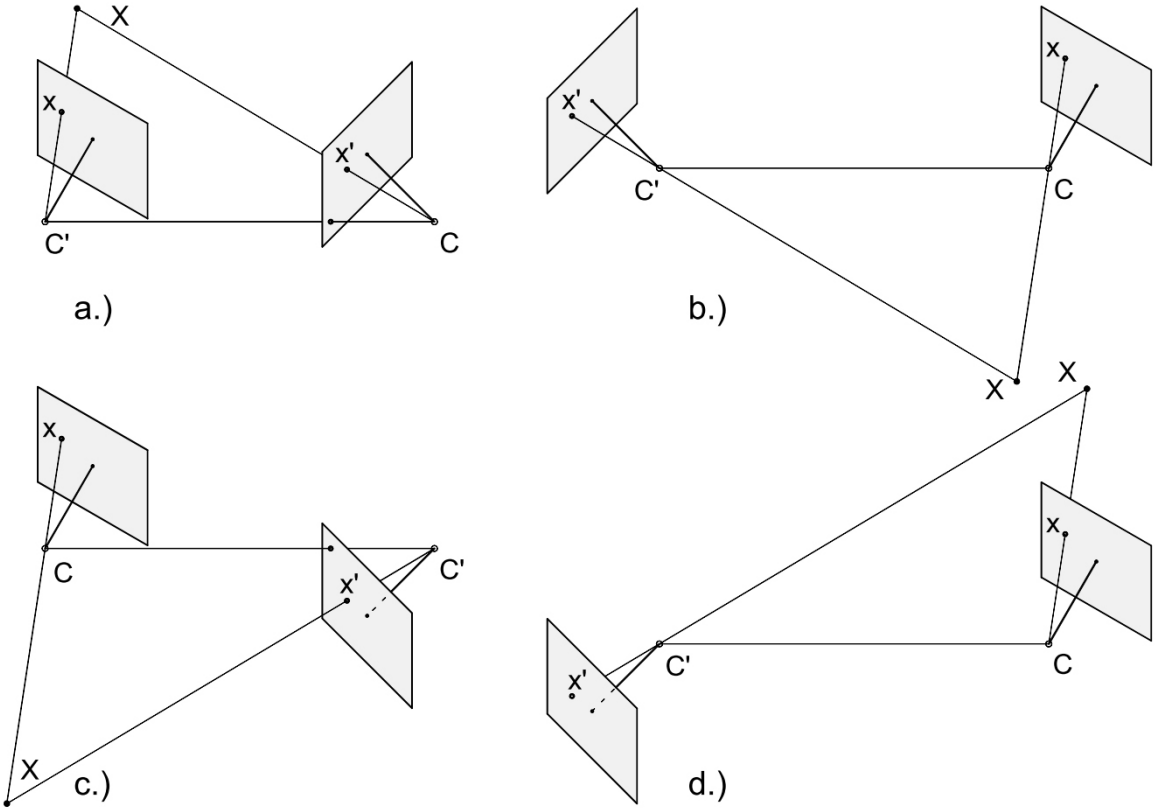



Figure 2.5: ***The four possible solutions for the camera pair obtained from the essential matrix:*** *In (a) the point* $\boldsymbol{X}$ *lies in front of both images. In (b) and (d) the camera* $\boldsymbol{P}'$ *lies in the opposite direction relative to the configuration in (a). In (c) and (d) the camera* $\boldsymbol{P}'$ *rotates by 180° around the baseline. Thus, in (b),(c) and (d) the point* $\boldsymbol{X}$ *lies at least behind on the cameras.*

**Projective reconstruction of a Camera Triplet from the Trifocal Tensor:** As already noted in chapter 2.1.1, a consistent triplet of camera matrices $\{\mathbf{P}, \mathbf{P}', \mathbf{P}''\}$ can be obtained directly from the trifocal tensor $\mathcal{T}$ up to a perspective ambiguity. The first camera is defined at $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$, the second camera $\mathbf{P}'$ and the third camera $\mathbf{P}''$ can be obtained according to equation 2.7 and equation 2.8. In a next step, this perspective reconstruction of the camera triplet may be updated to a euclidean one.

**Euclidean Reconstruction of Camera Triple:** Csurka and Horaud in [14] provide a method of finding the collineation between two projective reconstructions. A homography matrix can be found that maps the 3D projective space onto the 3D euclidean space. Let $\hat{\mathbf{X}}$ be a set of points in the projective space where $\hat{\mathbf{X}}_i$ is the $i^{th}$ column of $\hat{\mathbf{X}}$ and represents the 4-vector of a homogeneous point coordinate. $\mathbf{X}$ is the same set of points in the euclidean space. Thus, the $4 \times 4$ homography matrix $\mathbf{H}$ maps the points $\hat{\mathbf{X}}_i$ onto $\mathbf{X}_i$ according to $\mathbf{X}_i = \mathbf{H}\hat{\mathbf{X}}_i$

Thus, a set of points reconstructed from the perspective system $\hat{\mathbf{X}}$ obtained from the trifocal

tensor as well as the same set of points reconstructed from the euclidean system **X** obtained by the fundamental matrix is required. The reconstruction is done according to chapter 2.2.1. In the present application the complete set of correspondence points was used.
Since two euclidean systems with camera pairs {**P**,**P'**} and {**P**,**P''**} are available, one has to decide which one is considered as euclidean reference system for the upgrade of the perspective reconstruction. A reasonable decision would be to choose the system with the greater baseline between the two camera centres. The influence of the baseline for the quality of the reconstruction is illustrated in figure 2.6.

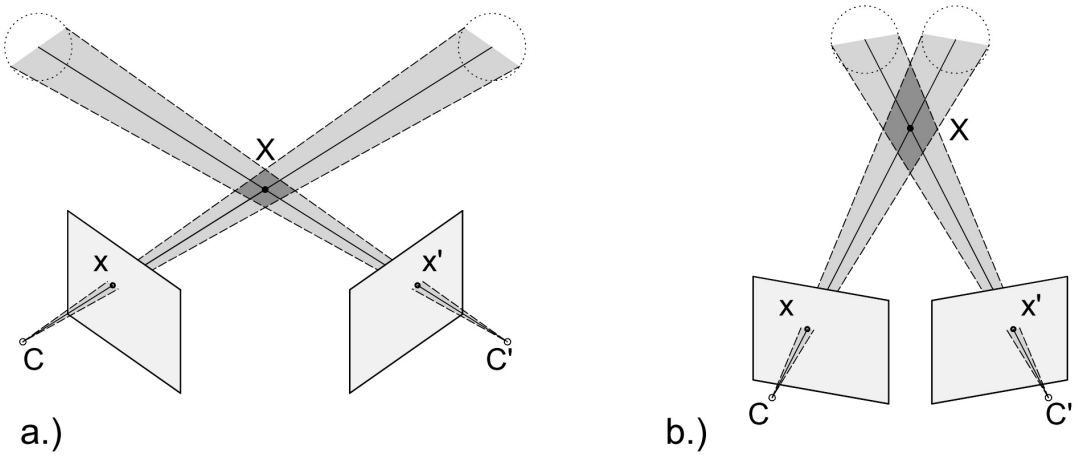



Figure 2.6: ***Influence of the baseline in the reconstruction:*** *In (a) the baseline is long and the angle between the two rays going through the corresponding points* $\boldsymbol{x}$ *and* $\boldsymbol{x}'$ *in the image is large. In (b) the baseline is short and the angle between the rays is small. The dark shaded area indicates the region of insecurity of the point* $\boldsymbol{X}$*. The more parallel the rays are, the bigger is the region of insecurity. With a longer baseline, the angle between the rays becomes bigger and the reconstruction of the point is more precise.*

In a linear algorithm the the euclidean distance **d** between $\hat{\mathbf{X}}_i$ and $\mathbf{H}\hat{\mathbf{X}}_i$ will be minimised in **H**, as illustrated in figure 2.7

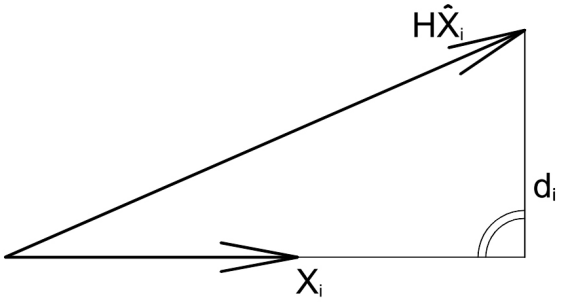



Figure 2.7: ***Minimising the euclidean distance:*** $d_i$ *is an approximation of the euclidean distance between the vectors* $\boldsymbol{X}_i$ *and* $\boldsymbol{H}\hat{\boldsymbol{X}}_i$

The euclidean distance is defined in 4D space associated with the euclidean space as a subspace of the projective space. This distance is computed by the theorem of Pythagoras according to

$$d_i^2 = \left(\mathbf{H}\hat{\mathbf{X}}_i\right)^\top \left(\mathbf{H}\hat{\mathbf{X}}_i\right) - \frac{\left(\mathbf{X}_i^\top \mathbf{H}\hat{\mathbf{X}}_i\right)^2}{\left(\mathbf{X}_i^\top \mathbf{X}_i\right)}. \tag{2.22}$$

The scale factor of $d$ is associated with the homogeneous vector $\mathbf{H}\hat{\mathbf{X}}_i$ and is computed as the distance of the vector $\mathbf{H}\hat{\mathbf{X}}_i$ to its projection onto the vector $\mathbf{X}_i$. Equation 2.22 can be converted according to

$$\begin{aligned} d_i^2 &= \hat{\mathbf{X}}_i^\top \mathbf{H}^\top \mathbf{I}_{4\times4} \mathbf{H}\hat{\mathbf{X}}_i - \left(\frac{1}{\mathbf{X}_i^\top \mathbf{X}_i}\right) \hat{\mathbf{X}}_i^\top \mathbf{H}^\top \mathbf{X}_i \mathbf{X}_i^\top \mathbf{H}\hat{\mathbf{X}}_i \\ &= \hat{\mathbf{X}}_i^\top \mathbf{H}^\top \left(\mathbf{I}_{4\times4} - \frac{\mathbf{X}_i \mathbf{X}_i^\top}{\mathbf{X}_i^\top \mathbf{X}_i}\right) \mathbf{H}\hat{\mathbf{X}}_i \\ &= \hat{\mathbf{X}}_i^\top \mathbf{H}^\top \mathbf{A}_i \mathbf{H}\hat{\mathbf{X}}_i \end{aligned} \tag{2.23}$$

where the matrix $\mathbf{A}_i = \left(\mathbf{I}_{4\times4} - \frac{\mathbf{X}_i \mathbf{X}_i^\top}{\mathbf{X}_i^\top \mathbf{X}_i}\right)$.
The expression $\mathbf{H}\hat{\mathbf{X}}_i$ can be written as linear system of equations according to

$$\mathbf{H}\hat{\mathbf{X}}_i = \begin{bmatrix} \hat{\mathbf{X}}_i^\top & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \hat{\mathbf{X}}_i^\top & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} & \hat{\mathbf{X}}_i^\top & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} & \hat{\mathbf{X}}_i^\top \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_{16} \end{bmatrix} = \mathbf{E}_i \mathbf{h} \tag{2.24}$$

where $\mathbf{h}$ contains the 16 elements $h_i$ of the $4 \times 4$ matrix $\mathbf{H}$. The equation for the euclidean distance can now be written as

$$d_i^2 = \mathbf{h}^\top \mathbf{E}_i^\top \mathbf{A}_i \mathbf{E}_i \mathbf{h}. \tag{2.25}$$

Thus, the error function $d$ can be obtained according to

$$d^2 = \sum_i d_i^2 = \mathbf{h}^\top \sum_i \left(\mathbf{E}_i \mathbf{A}_i \mathbf{E}_i\right) \mathbf{h} = \mathbf{h}^\top \mathbf{A}\mathbf{h} \tag{2.26}$$

where $\mathbf{A} = \sum_i \left(\mathbf{E}_i \mathbf{A}_i \mathbf{E}_i\right)$. This function is minimised performing the singular value decomposition on $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. The solution that minimises this function in the elements of $\mathbf{H}$ is the vector corresponding to the smallest singular vale of $\mathbf{A}$. Thus, $\mathbf{h}$ is the last column of $\mathbf{V}$. The received homography matrix is applied to the camera matrices of the triplet $\{\hat{\mathbf{P}}, \hat{\mathbf{P}}', \hat{\mathbf{P}}''\}$ obtained by the trifocal tensor in the example of the first camera according to

$$\mathbf{P} = \hat{\mathbf{P}}\mathbf{H} \tag{2.27}$$

The same is done for the other camera matrices $\mathbf{P}'$ and $\mathbf{P}''$. In a next step it is necessary to enforce the constraints of the camera matrices to obtain valid cameras. This is done according to chapter 1.3.3. Thus, the camera matrix $\mathbf{P}$ can be decomposed as $\mathbf{P} = \tilde{\mathbf{K}}\mathbf{R}[\mathbf{I} \mid -\mathbf{C}]$ where

$\mathbf{R}$ is the orthogonal rotation matrix, $\mathbf{C}$ is the camera centrer and $\tilde{\mathbf{K}}$ the camera calibration matrix. The matrix $\tilde{\mathbf{K}}$ is substituted by the intrinsic camera matrix $\mathbf{K}$ obtained by the calibrations of the camera. The camera matrix is computed again according to $\mathbf{P} = \mathbf{KR}[\mathbf{I} \mid -\mathbf{C}]$ Thus, a valid camera matrix is obtained. The same also holds for camera matrix $\mathbf{P}'$ and $\mathbf{P}''$.

The steps for retrieving the camera matrices are summarised in algorithm 2.5.

**Algorithm summary:**
The algorithm computes a projective reconstruction of the camera triplet $\{\mathbf{P}, \mathbf{P}', \mathbf{P}''\}$ and updates this system to an euclidean reconstruction using the knowledge of the camera calibration.

**Algorithm:**

1. Compute the fundamental matrices $\mathbf{F}_{21}$ and $\mathbf{F}_{31}$ from the trifocal tensor $\mathcal{T}$ according to equation 2.5 and equation 2.6.
2. Compute the essential matrices $\mathbf{E}_{21}$ and $\mathbf{E}_{31}$ according to equation 2.13 and equation 2.14. The internal constraints of the essential matrices can be satisfied according to equation 2.16.
3. Recover the camera pair $\{\mathbf{P}, \mathbf{P}'\}$ from $\mathbf{E}_{21}$ and the camera pair $\mathbf{P}, \mathbf{P}''$ from $\mathbf{E}_{31}$ retrieving four possible solutions for the camera pair from the essential matrix. Select the solution where the points lie in front of the camera pair.
4. Recover a consistent camera triplet $\{\mathbf{P}, \mathbf{P}', \mathbf{P}''\}$ from the trifocal tensor. Define $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$, the second and third camera matrix $\mathbf{P}'$ and $\mathbf{P}''$ can be obtained according to equation 2.7 and 2.8.
5. Upgrade the projective camera triplet to an euclidean system computing the $4 \times 4$ homography matrix that maps the projective points $\hat{\mathbf{X}}$ onto the euclidean points $\mathbf{X}$ according to $\mathbf{X}_i = \mathbf{H}\hat{\mathbf{X}}_i$. Apply this homography to the camera matrices $\mathbf{P} = \hat{\mathbf{P}}\mathbf{H}$, $\mathbf{P}' = \hat{\mathbf{P}}'\mathbf{H}$, $\mathbf{P}'' = \hat{\mathbf{P}}''\mathbf{H}$. Enforce the internal constraint of the camera matrix substituting the matrix $\tilde{\mathbf{K}}$ of the decomposition $\mathbf{P} = \tilde{\mathbf{K}}\mathbf{R}[\mathbf{I} \mid -\mathbf{C}]$ by the intrinsic camera matrix $\mathbf{K}$ obtained by the calibration. Recompute the camera matrix $\mathbf{P} = \mathbf{KR}[\mathbf{I} \mid -\mathbf{C}]$. This has to be done also for the camera matrices $\mathbf{P}'$ and $\mathbf{P}''$.

**Algorithm 2.5:** ***Retrieving camera matrices from the trifocal tensor***

## 2.2 Structure Computation

From the point correspondences $\mathbf{x}$ and the line correspondences $\mathbf{l}$ across the images and the camera matrices $\mathbf{P}$ points and lines in 3-space can be reconstructed. The camera matrices of the camera triplet $\{\mathbf{P}, \mathbf{P}', \mathbf{P}''\}$ are known up to a scaling factor. Thus, the points $\mathbf{X}$ and lines $\mathbf{L}$ in the euclidean 3D space can be recovered.

### 2.2.1 Point Reconstruction

Let $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{x}''$ be the homogeneous image points in the first, second and third image. Thus, the projection of the homogeneous 3D points $\mathbf{X}$ satisfies the equations $\mathbf{x} = \mathbf{PX}$, $\mathbf{x}' = \mathbf{P}'\mathbf{X}$ and $\mathbf{x}'' = \mathbf{PX}$. If the data are perfect without any noise, a reconstruction can be obtained by simply back projecting the rays from the image points $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{x}''$. The point $\mathbf{X}$ in 3-space lies exactly in the intersection point of the rays. This holds for perfect data without any noise. Since in practice image data are noisy, this triangulation does not work. It is

assumed that the noise in the image is Gaussian. Hartley and Zisserman in [25], chapter 12, provide several linear and non linear methods for computing three dimensional points $\mathbf{X}$ from image correspondences. For the present algorithm the linear triangulation method is applied to reconstruct the points. This linear method can be used to reconstruct points in 3 space from image correspondences from an arbitrary number of images. This is important because other methods allow the reconstruction from correspondences in an image pair only. In the present case three images are present.
The equation for the points in the first image can be written as

$$\mathbf{PX} = \begin{bmatrix} \mathbf{P}^1\mathbf{X} \\ \mathbf{P}^2\mathbf{X} \\ \mathbf{P}^3\mathbf{X} \end{bmatrix} \tag{2.28}$$

where $\mathbf{P}^i$ denotes the $i^{th}$ row of the camera matrix. From the cross product $\mathbf{x} \times \mathbf{PX} = \mathbf{0}$ a solution for the equation $\mathbf{A}_i\mathbf{X} = \mathbf{0}$ can be obtained. Let $\mathbf{x} = (x, y, 1)$, then

$$\mathbf{x} \times \mathbf{PX} = \begin{bmatrix} y\left(\mathbf{P}^3\mathbf{X}\right) - \left(\mathbf{P}^2\mathbf{X}\right) \\ x\left(\mathbf{P}^3\mathbf{X}\right) - \left(\mathbf{P}^1\mathbf{X}\right) \\ x\left(\mathbf{P}^2\mathbf{X}\right) - y\left(\mathbf{P}^1\mathbf{X}\right) \end{bmatrix} = \mathbf{0} \tag{2.29}$$

This expression can be reformulated in terms of $\mathbf{A}_i\mathbf{X} = \mathbf{0}$ according to

$$\begin{bmatrix} y\mathbf{P}^3 - \mathbf{P}^2 \\ x\mathbf{P}^3 - \mathbf{P}^1 \\ x\mathbf{P}^2 - y\mathbf{P}^1 \end{bmatrix} \mathbf{X} = \mathbf{0}. \tag{2.30}$$

Only two of this equations are linearly independent, thus only two equations must be considered. Thus, the matrix linear system of equations can be rewritten as

$$\mathbf{A}_i\mathbf{X} = \begin{bmatrix} y\mathbf{P}^3 - \mathbf{P}^2 \\ x\mathbf{P}^3 - \mathbf{P}^1 \end{bmatrix} \mathbf{X} = \mathbf{0}. \tag{2.31}$$

Doing this for the other two images, the matrix $\mathbf{A}$ of the system of equations $\mathbf{AX} = \mathbf{0}$ can be obtained stacking the single matrices $\mathbf{A}_i$ according to

$$\mathbf{AX} = \begin{bmatrix} y\mathbf{P}^3 - \mathbf{P}^2 \\ x\mathbf{P}^3 - \mathbf{P}^1 \\ y'\mathbf{P}'^3 - \mathbf{P}'^2 \\ x'\mathbf{P}'^3 - \mathbf{P}'^1 \\ y''\mathbf{P}''^3 - \mathbf{P}''^2 \\ x''\mathbf{P}''^3 - \mathbf{P}''^1 \end{bmatrix} \mathbf{X} = \mathbf{0}. \tag{2.32}$$

Thus, the vector $\mathbf{X}$, that minimises this equation, can by obtained performing the singular value decomposition on $\mathbf{A}$ according to $\mathbf{A} = \mathbf{UDV}^\top$. Then the vector $\mathbf{X}$, that minimises the equation $\mathbf{AX} = \mathbf{0}$ is the last column of the matrix $\mathbf{V}$. The steps for reconstructing a point from three views are summarised in algorithm 2.6.

**Algorithm summary:**
Computes the point $\mathbf{X}$ in 3-space from the correspondence $\mathbf{x} \leftrightarrow \mathbf{x}' \leftrightarrow \mathbf{x}''$ over three images provided the camera matrices $\mathbf{P}, \mathbf{P}', \mathbf{P}''$ are known.

**Algorithm:**

1. Make the cross product $\mathbf{x} \times \mathbf{PX} = \mathbf{0}$ for each of the three images.
2. Form the system of equations $\mathbf{A}_i\mathbf{X} = \mathbf{0}$ from the cross product of each of the images $i = 1 \dots 3$.
3. Concatenate the matrices $\mathbf{A}_i$ for $i = 1...3$ such that the system of equations $\mathbf{AX} = \mathbf{0}$ arises.
4. Perform the SVD on $\mathbf{A} = \mathbf{UDV}^\top$. The homogeneous vector $\mathbf{X}$ is the last column of the matrix $\mathbf{V}$.

**Algorithm 2.6:** ***Point reconstruction:*** *Linear algorithm to compute points in 3-space from image correspondences across three images.*

### 2.2.2 Line Reconstruction

From a line correspondence across three images and the known camera matrices, the line in 3 space can be computed. In [25], chapter 12.7, page 312 ff. a linear method is provided. Suppose two images are given with line correspondences. Then the line in 3 space can be computed simply by computing the intersection of the planes that goes trough the image centrer and the image line. This is also possible when noise is present in the images, because the planes always intersect in a specified line. Compared to a point correspondence $\mathbf{x} \leftrightarrow \mathbf{x}'$, where the correlation is overdetermined because of 4 measurements in $\mathbf{x}$ and $\mathbf{x}'$ on three degrees of freedom in the space point $\mathbf{X}$, the line correspondence $\mathbf{l} \leftrightarrow \mathbf{l}'$ is exactly determined. The two image lines provides two measurements a time, the line in 3 space has 4 degrees of freedom. With three images however, in the presence of noise, the line in 3 space cannot be determined uniquely, because from the three image lines in total 6 measurements are provided for the 4 degrees of freedom of the line in 3 space. Thus a minimisation is required.
With a linear minimisation algorithm the line L can be recovered. Thus, the 4 vector of the plane $\boldsymbol{\pi}$ that goes trough the camera centrer $\mathbf{C}$ and the line $\mathbf{l}$ must be computed according to $\boldsymbol{\pi} = \mathbf{P}^\top\mathbf{l}$. The same has to be done in the other images to obtain $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$. From those planes, the matrix $\mathbf{A}$ can be formed according to $\mathbf{A} = \left(\boldsymbol{\pi}^\top, \boldsymbol{\pi}'^\top, \boldsymbol{\pi}''^\top\right)^\top$. From $\mathbf{A}$ the singular value decomposition $\mathbf{A} = \mathbf{UDV}^\top$ is computed. The first two columns of $\mathbf{V}$, that correspond to the largest singular values, define the line $\mathbf{L}$ as intersection of two planes. The Plücker matrix can be obtained as described in chapter 1.3.8. In algorithm 2.7 the steps for reconstructing lines from three images are summarised.

## 2.3 Connection of Single Reconstructions

Suppose, two reconstructions of the same scene from in total 4 views are provided. The first reconstruction was made from the views $1, 2, 3$, the second reconstruction is provided from the views $2, 3, 4$. Let the reconstructed points from the first system be denoted by $\mathbf{X}_1$ and the reconstructed points from the second system be denoted by $\mathbf{X}_2$. The cameras in the first

**Algorithm summary:**
Computes the reconstruction of a line $\mathbf{L}$ in 3 space from the correspondence $\mathbf{l} \leftrightarrow \mathbf{l}' \leftrightarrow \mathbf{l}''$ over three images, provided that the camera matrices $\mathbf{P}, \mathbf{P}', \mathbf{P}''$ are known.

**Algorithm:**

1. compute the plane $\boldsymbol{\pi} = \mathbf{P}^\top \mathbf{l}$ of the first image. Do the same in the other images to obtain $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$.
2. Form the matrix $\mathbf{A} = \left(\boldsymbol{\pi}^\top, \boldsymbol{\pi}'^\top, \boldsymbol{\pi}''^\top\right)^\top$ from the three planes.
3. Perform the SVD on $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. The line $\mathbf{L}$ corresponds to the first two columns of $\mathbf{V}$. Thus $\mathbf{L}$ is defined as the intersection of two planes.
4. compute the Plücker matrix of the line $\mathbf{L}$ according to chapter 1.3.8.

**Algorithm 2.7:** ***Line reconstruction:*** *Linear algorithm to compute lines in 3-space from image correspondences.*

system are denoted as $\mathbf{P}_1, \mathbf{P}'_1, \mathbf{P}''_1$, the cameras of the second system as $\mathbf{P}_2, \mathbf{P}'_2, \mathbf{P}''_2$. Since the first camera $\mathbf{P}_i$ of any reconstruction $i$ is defined as $\mathbf{P}_i = [\mathbf{I} \mid \mathbf{0}]$ and the other two cameras $\mathbf{P}'_i$ and $\mathbf{P}''_i$ are computed relative to $\mathbf{P}_i$, no information about the position and orientation relative to a common world frame is provided. Thus, the frame of the first reconstruction is defined as the world frame, with the camera $\mathbf{P}_1$ positioned in the origin. The other reconstructions will be aligned relative to this frame.
The concatenation of two reconstructions is illustrated in figure 2.8 and is explained next.

In a first step the scaling factor between the two reconstructions will be computed. Since the reconstructions are euclidean and defined up to an arbitrary scale factor, a similarity transformation must be applied. Determine the centroid $\mathbf{S}_1$ of all points $\mathbf{X}_1$. Then compute the average distance $d_1$ of the camera centrers $\mathbf{C}'_1$ and $\mathbf{C}''_1$ relative to $\mathbf{S}_1$. Do the same for the second reconstructions and obtain $\mathbf{S}_2$ and $d_2$. The scaling factor $s$ is computed according to $s = {}^{d1}/_{d2}$. In the next step the rotation of the second system relative to the base system will be computed. For this process the rotation matrices $\mathbf{R}''_1$ and $\mathbf{R}'_2$ of the cameras $\mathbf{P}''_1$ and $\mathbf{P}'_2$ are required. They can be obtained from the decomposition $\mathbf{P} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\mathbf{C}\right]$ according to chapter 1.3.3. The relative orientation of the second reconstruction system to the first system is computed according to $\mathbf{R} = \mathbf{R}''^\top_1 \mathbf{R}'_2$. Thus, the transformation matrix $\hat{\mathbf{H}}$ is computed.

$$\hat{\mathbf{H}} = \left[\mathbf{I} \mid \mathbf{C}'_2\right] \mathbf{S}\mathbf{R} \left[\mathbf{I} \mid -\mathbf{C}'_2\right] \tag{2.33}$$

$\mathbf{S}$ is the homogeneous scaling matrix composed of $\mathbf{S} = s\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. $\mathbf{C}'_2$ is the camera centrer of the camera $\mathbf{P}'_2$, obtained form the decomposition in chapter 1.3.3. The Homography $\hat{\mathbf{H}}$ performs a translation from the camera centrer $\mathbf{C}'_2$ to the origin of base frame, applies the rotation and the scaling and retranslates to the starting point $\mathbf{C}'_2$.
This homography is applied to the centroid $\mathbf{S}_2$ of the points of the second system according to $\hat{\mathbf{S}}_2 = \hat{\mathbf{H}}\mathbf{S}_2$. Thus, the translation can be obtained from the distance between $\mathbf{S}_1$ and $\hat{\mathbf{S}}_2$ according to $\hat{\mathbf{t}} = \mathbf{S}_1 - \hat{\mathbf{S}}_2$. From the inhomogeneous translation vector $\hat{\mathbf{t}}$ the homogeneous translation matrix $\mathbf{T}$ is constructed according to

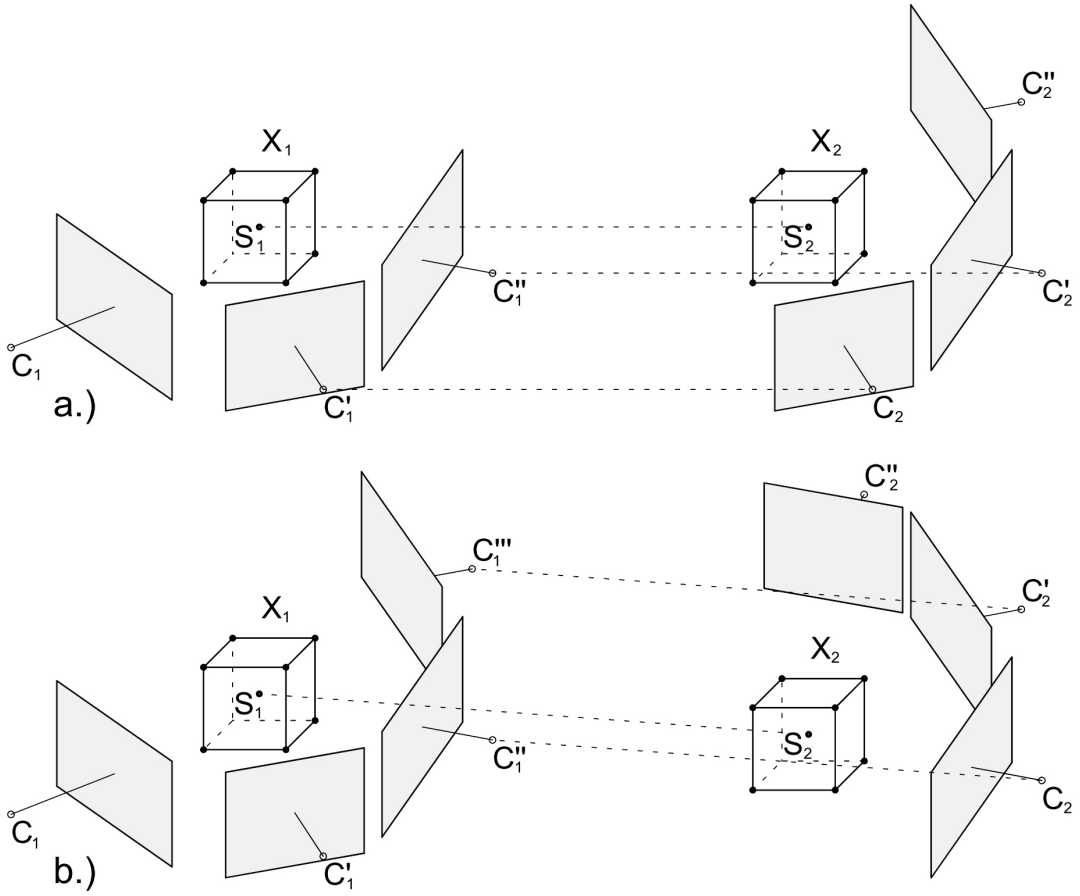



Figure 2.8: ***Connection of single reconstructions*** *(a) shows two euclidean reconstructions of the same scene, each consisting of three cameras. The first two cameras $\boldsymbol{P}_2, \boldsymbol{P}'_2$ of the second reconstruction correspond to the last two cameras $\boldsymbol{P}'_1, \boldsymbol{P}''_1$ of the first reconstruction. Thus, the two reconstructions will be concatenated to a system consisting of four cameras as illustrated in the left image of (b). In (b) a third euclidean reconstruction will be attached to the existing system. This is done in the same way as in (a).*

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & \hat{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.34}$$

The transformation matrix $\mathbf{H}$ that is used to concatenate the single reconstructions is obtained from $\mathbf{H} = \mathbf{T}\hat{\mathbf{H}}$. Applying this matrix to cameras, points and lines of the second reconstruction translates the system into the base frame of the first reconstruction and concatenates the two systems. Selecting all three cameras from the first system and the last camera from the second system an augmented system consisting of 4 cameras is obtained. The best reconstruction of points and lines determining the reprojection error over all four cameras is selected.
Suppose now many reconstruction systems are provided, each consisting of three frames $\mathbf{P}_i, \mathbf{P}'_i, \mathbf{P}''_i$ where the cameras $\mathbf{P}'_i, \mathbf{P}''_i$ of the system $i$ overlap with the cameras $\mathbf{P}_{i+1}, \mathbf{P}'_{i+1}$ of the

system $i+1$. Then the base frame of the first system is considered as global world frame. The system $i+1$ is concatenated to the system $i$ as described previously. Doing this for $i = 1...n$ with $n$ systems, a coherent system consisting of $n+2$ cameras is obtained.

The process of connecting the single reconstructions is summarised in algorithm 2.8.

**Algorithm summary:**
Given two similarity reconstructions of a scene, they may be concatenated to receive a connected scene that contains the elements of both the first and the second scene. Each reconstruction must consist of a triplet of cameras $\mathbf{P}_1, \mathbf{P}'_1, \mathbf{P}''_1$ and $\mathbf{P}_2, \mathbf{P}'_2, \mathbf{P}''_2$ respectively. The points and lines of the scene must be present in both reconstructions as $\mathbf{X}_1$,$\mathbf{L}_1$ in the first system and $\mathbf{X}_2$,$\mathbf{L}_2$ in the second system respectively. The first two cameras $\mathbf{P}_2, \mathbf{P}'_2$ of the second reconstruction system must overlap with the last two cameras $\mathbf{P}'_1, \mathbf{P}''_1$ of the first reconstruction system.

**Algorithm:**

1. Determine the scale factor $s$ between the two reconstructions. It is computed as the ratio $s = {}^{d_1}/_{d_2}$ where $d_1$ is the average distance of the centroid $\mathbf{S}_1$ of the points $\mathbf{X}_1$ to the camera centrer $\mathbf{C}'_1$ and $\mathbf{C}''_1$ respectively. The distance $d_2$ is the average distance of the centroid $\mathbf{S}_2$ of the points $\mathbf{X}_2$ to the camera centrer $\mathbf{C}_2$ and $\mathbf{C}'_2$ respectively.
2. Determine the rotation of the second reconstruction relative to the first one. This is done according to $\mathbf{R} = \mathbf{R}''^{\top}_1 \mathbf{R}'_2$. The rotation matrix $\mathbf{R}''_1$ is received from the decomposition $\mathbf{P}''_1 = \mathbf{K}\mathbf{R}''_1 \left(\mathbf{I} \mid -\mathbf{C}''_1\right)$, $\mathbf{R}'_2$ from the decomposition $\mathbf{P}'_2 = \mathbf{K}\mathbf{R}'_2 \left(\mathbf{I} \mid -\mathbf{C}'_2\right)$.
3. Compute the transformation matrix $\hat{\mathbf{H}} = \left[\mathbf{I} \mid \mathbf{C}'_2\right] \mathbf{S}\mathbf{R} \left[\mathbf{I} \mid -\mathbf{C}'_2\right]$. $\mathbf{S}$ is the similarity Matrix obtained from $s$.
4. Determine the translation $\mathbf{t}$ from the distance between $\mathbf{S}_1$ and $\hat{\mathbf{S}}_2$. $\hat{\mathbf{S}}_2$ is the centroid of the points $\mathbf{X}_2$, transformed by the homography $\hat{\mathbf{H}}$ according to $\hat{\mathbf{S}}_2 = \mathbf{H}\mathbf{S}_2$. From the translation vector $\mathbf{t}$, form the homogeneous translation matrix $\mathbf{T}$.
5. Compute the homography $\mathbf{H} = \mathbf{T}\hat{\mathbf{H}}$.
6. Apply this homography matrix to points $\mathbf{X}_2$, lines $\mathbf{L}_2$ and cameras $\mathbf{P}_2, \mathbf{P}'_2, \mathbf{P}''_2$ of the second reconstruction.

**Algorithm 2.8:** ***Concatenation of reconstructions***

In this chapter the reconstruction of a scene from image correspondences, consisting of points, lines and cameras, was described. An important criterion in choosing the methods for computing the structure of the scene was the real time capability. For the minimisation problems, that are solved here with linear least square methods, also iterative non-linear least square methods are provided. Hartley and Zisserman in [25], where most of the reconstruction algorithms are obtained, compare the linear algorithms with the iterative one. In all cases the linear algorithms used here where recommended as a good choice, when optimality is not required, but good results lying close to the optimal solution, compared with a relatively low computation cost are desired.
In the present problem, optimal results in the reconstruction are not required. The reconstructed entities are used only as initialisation for the optimisation process, that will be

described in the next chapter. In this process, the reconstruction will be refined iteratively. Since this refinement is thought to be executed in a separate process that does not influence the tracking procedure, no real time behaviour is expected. Thus, the initialisation has to be computed fast but as accurate as possible, the optimisation process however performs iteratively and is not real time capable.

In algorithm 2.9 the complete reconstruction process, starting with the computations of the trifocal tensor up to the concatenation of the single reconstructions will be summarised.

**Algorithm summary:**
Given point and line correspondences across multiple images of an image stream, recorded by a calibrated camera, the algorithm computes a similarity reconstruction of the scene consisting of points, lines and an arbitrary number of camera poses.

**Algorithm:**

1. From the first three images of the stream a geometrically valid tensor $\mathcal{T}$ can be obtained according to algorithm 2.4.
2. Determine the camera triplet $\mathbf{P}, \mathbf{P}', \mathbf{P}''$ according to algorithm 2.5.
3. Compute the euclidean reconstruction of the points $\mathbf{X}$ from algorithm 2.6.
4. Compute the euclidean reconstruction of the lines $\mathbf{L}$ from algorithm 2.7.
5. repeat the steps for the camera triplet $\mathbf{P}', \mathbf{P}'', \mathbf{P}'''$. Thus, two reconstructions are available where the first two cameras of the second reconstruction correspond to the second and third camera of the first reconstruction.
6. according to algorithm 2.8 concatenate the two reconstructions.
7. repeat the last two steps for the complete stream of a subset of views from the stream.

**Algorithm 2.9:** ***Reconstruction process of an image stream***

# 3 Bundle Adjustment

In chapter 2 the reconstruction of points and lines from images was described. Since the data in the images are noisy, the reconstructions are not perfectly determined, however each reconstruction has a certain error in the pose of cameras, points and lines. When the single reconstructions, that consist of the scene composed of points and lines and a camera triplet, are concatenated, points and lines of one reconstruction will be projected into the images of the other cameras. This causes in an increased error, because the error of the single reconstructions accumulate. After a certain number of reconstructions are merged together, A minimisation of the error of the complete set of points, lines and cameras is executed. This technique is called Bundle Adjustment. The idea behind the Bundle Adjustment concept is summarised in chapter 1.4. For this process a suitable representation of cameras, points and lines is essential. The following chapter provides convenient parametrisations of this entities and gives an appropriate algorithm to perform the minimisation process using an implementation of the Levenberg-Marquardt algorithm.

## 3.1 Parametrisation Camera Parameter

A camera matrix $\hat{\mathbf{P}}$ is composed of

$$\hat{\mathbf{P}} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\tilde{\mathbf{C}}\right] \tag{3.1}$$

where $\mathbf{K}$ is a upper triangular $3 \times 3$ calibration matrix, $\mathbf{R}$ is a $3 \times 3$ rotation Matrix and $\tilde{\mathbf{C}}$ is the inhomogeneous vector of the camera-centre in the world frame.
For the bundle adjustment process normalised camera matrices are used. The concept of normalised cameras and normalise coordinates is explained in [25], chapter 9.6, page257. With known calibration matrix, the normalised camera is computed according to

$$\mathbf{P} = \mathbf{K}^{-1}\hat{\mathbf{P}} = \mathbf{R}\left[\mathbf{I} \mid -\tilde{\mathbf{C}}\right]. \tag{3.2}$$

Thus $\mathbf{P}$ is a $3 \times 4$ projective camera-matrix that can be used to map features from the euclidean 3-space $\mathbb{R}^3$ to the projective 2-space $\mathbb{P}^2$. The matrix $\mathbf{P}$ has 12 elements, but only 6 degrees of freedom. The three translational parameters are present in the three elements of the vector $\tilde{\mathbf{C}}$, that represents the camera centre in the world frame. The three rotational parameters forms the $3 \times 3$ rotation matrix $\mathbf{R}$.
There are several ways to represent a rotation in the three-dimensional space, such as the rotation-matrix, Euler-angles or the unit Quaternions. A detailed overview about the most important mathematical concepts to represent rotations in the three-dimensional space is given by Diebel in [16]. In [55] Slabaugh discusses a technique to find all possible Euler-angles from a rotation matrix. A suitable representation is required that represents the camera matrices with the minimum number of parameters. From the rotation matrices, that are

described in chapter 1.3.2, the axis-angle representation can be obtained. This method, that is described by Hartley and Zisserman in [25], appendix A4.3.2 is used here with some modifications.

A rotation in 3-space, given as rotation Matrix **R**, can be represented as an axis of rotation **v** and the corresponding angle $\theta$, that is the magnitude of the 3-vector **v**. This is called the axis angle representation.

$$\theta = arccos\left(\frac{trace(\mathbf{R}) - 1}{2}\right) \tag{3.3}$$

$$\mathbf{w} = \frac{1}{2sin(\theta)}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{3.4}$$

**w** is the unity vector in direction of **v** with $\|\mathbf{w}\| = 1$. Thus, the 3-vector **v** is obtained according to

$$\mathbf{v} = \theta\mathbf{w}. \tag{3.5}$$

Additionally, the Rotation matrix **R** can be obtained from the rotation axis **v** and the rotation angle $\theta$ according to

$$\mathbf{R} = \mathbf{I} + [\mathbf{w}]_\times sin(\theta) + [\mathbf{w}]_\times^2 (1 - cos(\theta)). \tag{3.6}$$

$[\mathbf{w}]_\times$ is the skew-symmetric matrix corresponding to the vector **w**.

$$[\mathbf{w}]_\times = \begin{bmatrix} 0 & -w(z) & w(y) \\ w(z) & 0 & -w(x) \\ -w(y) & w(x) & 0 \end{bmatrix} \tag{3.7}$$

Thus, the six degrees of freedom of a camera-pose can be expressed as a 6-vector composed of the translation vector **t** and the rotation axis **v**. This Parameter Vector is defined as $\mathbf{s}_p = (t_x, t_y, t_z, v_x, v_y, v_z)^\top$.

Imagine now a pose update of a camera. Let $\tilde{\mathbf{P}}$ be a normalised $3 \times 4$ camera pose matrix of the last time step and **E** the homogeneous $4 \times 4$ Pose matrix that represents the update. Then the pose matrix of the new camera is $\mathbf{P} = \tilde{\mathbf{P}}\mathbf{E}$. Taking the Jacobian of **P** with respect to the parameter vector $\mathbf{s}_p$ leads to

$$\mathbf{J}_P = \frac{\partial \mathbf{P}}{\partial \mathbf{s}_p} = \tilde{\mathbf{P}}\frac{\partial \mathbf{E}}{\partial \mathbf{s}_p}. \tag{3.8}$$

The update matrix **E** is composed of the incremental rotation $\mathbf{R}_m$, the incremental translation $\mathbf{t}_m$ and the last row for homogenisation.

$$\mathbf{E} = \begin{bmatrix} \mathbf{P}_m & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_m & & & \mathbf{t}_m \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Taking the derivative of **E** with respect to the translation parameters evaluated at the point $\mathbf{t}_m = (0,0,0)^\top$ leads to the matrices

$$\left.\frac{\partial \mathbf{E}}{\partial t_x}\right|_{t_x=0} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad \left.\frac{\partial \mathbf{E}}{\partial t_y}\right|_{t_y=0} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad \left.\frac{\partial \mathbf{E}}{\partial t_z}\right|_{t_z=0} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.10}$$

Thus, the derivative of the camera pose matrix **P** with respect to the translational parameters is

$$\left.\frac{\partial \mathbf{P}}{\partial t_x}\right|_{t_x=0} = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \tilde{\mathbf{P}}_1 \end{bmatrix} \tag{3.11}$$

$$\left.\frac{\partial \mathbf{P}}{\partial t_y}\right|_{t_y=0} = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \tilde{\mathbf{P}}_2 \end{bmatrix} \tag{3.12}$$

$$\left.\frac{\partial \mathbf{P}}{\partial t_z}\right|_{t_z=0} = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \tilde{\mathbf{P}}_3 \end{bmatrix}. \tag{3.13}$$

The derivative of **E** with respect to the rotation axis **v** evaluated at $\theta = 0$ results in

$$\left.\frac{\partial \mathbf{E}}{\partial v_x}\right|_{\theta=0} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad \left.\frac{\partial \mathbf{E}}{\partial v_y}\right|_{\theta=0} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad \left.\frac{\partial \mathbf{E}}{\partial v_z}\right|_{\theta=0} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.14}$$

Thus, the derivative of the camera pose matrix **P** with respect to the rotational parameters is

$$\left.\frac{\partial \mathbf{P}}{\partial v_x}\right|_{\theta=0} = \begin{bmatrix} \mathbf{0}_{3\times1} & \tilde{\mathbf{P}}_3 & -\tilde{\mathbf{P}}_2 & \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.15}$$

$$\left.\frac{\partial \mathbf{P}}{\partial v_y}\right|_{\theta=0} = \begin{bmatrix} -\tilde{\mathbf{P}}_3 & \mathbf{0}_{3\times1} & \tilde{\mathbf{P}}_1 & \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.16}$$

$$\left.\frac{\partial \mathbf{P}}{\partial v_z}\right|_{\theta=0} = \begin{bmatrix} \tilde{\mathbf{P}}_2 & -\tilde{\mathbf{P}}_1 & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.17}$$

$$\tag{3.18}$$

where $\tilde{\mathbf{P}}_i$ is the $i^{th}$ column of the matrix $\tilde{\mathbf{P}}$. This results are used later to determine the Jacobian of the reprojection error.

## 3.2 Parametrisation Point features

Suppose, the scene consists of a set of points $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n\}$ and a set of Cameras $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_m\}$. The point $\mathbf{x}_{ij}$ in the image plane of the camera $\mathbf{P}_j$ is a homogeneous vector $\mathbf{x}_{ij} = (x_{ij}, y_{ij}, 1)^\top$ in the projective 2-space $\mathbb{P}^2$ and correspond to the scene point $\mathbf{X}_i$,

that is a homogeneous vector $\mathbf{X}_i = (X_i, Y_i, Z_i, 1)^\top$ in the euclidean 3-space $\mathbb{R}^3$.
Projecting the estimated Point $\mathbf{X}_i$ back to the image plane using the camera matrix $\mathbf{P}_j$ leads to the reprojected Point $\hat{\mathbf{x}}_{ij}$ according to

$$\hat{\mathbf{x}}_{ij} = \mathbf{P}_j \mathbf{X}_i. \tag{3.19}$$

A point $\mathbf{X}_i$ in 3-space can be represented by the homogeneous vector $\mathbf{X}_i = (X_i, Y_i, Z_i, 1)$. Since the homogeneous 4-vector is defined up to scale, it has 3 degrees of freedom. Thus, the point depends on three euclidean coordinates $\{X_i, Y_i, Z_i, \}$. Let the vector $\mathbf{s}_x$ be the parameter vector of the point $\mathbf{X}_x$. Thus, it is composed according to $\mathbf{s}_x = (X_i, Y_i, Z_i)$. The 6 parameters of the camera can be obtained from chapter 3.1. Thus, the parameter vector for the camera parameter $\mathbf{s}_p$ is composed according to $\mathbf{s}_p = (t_x, t_y, t_z, v_x, v_y, v_z)$.
The reprojected point $\mathbf{x}_{ij}$ depends on the parameters in $\mathbf{s}_x$ as well as on the parameters $\mathbf{s}_p$.

Since the homogeneous vector of the reprojected point is $\hat{\mathbf{x}}_{ij} = \mathbf{P}_j \mathbf{X}_i$, the inhomogeneous vector of $\hat{\mathbf{x}}_{ij}$ is

$$\begin{pmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \end{pmatrix} = \begin{pmatrix} \frac{\mathbf{P}_j^1 \mathbf{X}_i}{\mathbf{P}_j^3 \mathbf{X}_i} \\ \frac{\mathbf{P}_j^1 \mathbf{X}_i}{\mathbf{P}_j^3 \mathbf{X}_i} \end{pmatrix}. \tag{3.20}$$

$\mathbf{P}_j^k$ denotes the $k^{th}$ row of the camera matrix $\mathbf{P}_j$.

For the minimisation process, the derivative of the point $\hat{\mathbf{x}}_{ij}$ with respect to the parameter in $\mathbf{s}_x$ and the parameters in $\mathbf{s}_p$ are required. This derivatives are the Jacobians, denoted here as $\mathbf{J}_x$ and $\mathbf{J}_p$.
The Jacobian $\mathbf{J}_x$ with respect to the point parameters is computed according to

$$\mathbf{J}_x = \frac{\partial \begin{pmatrix} \hat{x}(\mathbf{s}) \\ \hat{y}(\mathbf{s}) \end{pmatrix}}{\partial \mathbf{s}_x} = \begin{bmatrix} \frac{\partial \hat{x}(\mathbf{s})}{\partial X_i} & \frac{\partial \hat{x}(\mathbf{s})}{\partial Y_i} & \frac{\partial \hat{x}(\mathbf{s})}{\partial Z_i} \\ \frac{\partial \hat{y}(\mathbf{s})}{\partial X_i} & \frac{\partial \hat{y}(\mathbf{s})}{\partial Y_i} & \frac{\partial \hat{y}(\mathbf{s})}{\partial Z_i} \end{bmatrix} \tag{3.21}$$

Deriving the inhomogeneous point vector with respect to the parameter vector $\mathbf{s}_p$, the Jacobian $\mathbf{J}_p$ is obtained. To compute the Jacobian, the equations 3.11 to 3.17 are required for the derivative with respect to the translational parameter and the equation 3.15 to 3.15 are needed for the derivative with respect to the rotational parameter. This equations provide the derivative of the camera matrix $\mathbf{P}$ with respect to the 6 extrinsic parameters. Thus, the Jacobian

$$\mathbf{J}_p = \frac{\partial \begin{pmatrix} \hat{x}(\mathbf{s}) \\ \hat{y}(\mathbf{s}) \end{pmatrix}}{\partial \mathbf{s}_p} = \begin{bmatrix} \frac{\partial \hat{x}(\mathbf{s})}{\partial t_x} & \frac{\partial \hat{x}(\mathbf{s})}{\partial t_y} & \frac{\partial \hat{x}(\mathbf{s})}{\partial t_z} & \frac{\partial \hat{x}(\mathbf{s})}{\partial v_x} & \frac{\partial \hat{x}(\mathbf{s})}{\partial v_y} & \frac{\partial \hat{x}(\mathbf{s})}{\partial v_z} \\ \frac{\partial \hat{y}(\mathbf{s})}{\partial t_x} & \frac{\partial \hat{y}(\mathbf{s})}{\partial t_y} & \frac{\partial \hat{y}(\mathbf{s})}{\partial t_z} & \frac{\partial \hat{y}(\mathbf{s})}{\partial v_x} & \frac{\partial \hat{y}(\mathbf{s})}{\partial v_y} & \frac{\partial \hat{y}(\mathbf{s})}{\partial v_z} \end{bmatrix}. \tag{3.22}$$

is obtained. The derivations can be computed using the results from chapter 3.1.

## 3.3 Parametrisation Line features

In contrast to the representation of points in 3D, the representation of lines is more complex. An suitable description of lines in 3D is required to handle the projection of lines from the euclidean 3-space $\mathbb{R}^3$ to the projective 2-space $\mathbb{P}^2$. Furthermore a parametrisation is needed that represents the lines with the minimal number of parameters for the Bundle Adjustment process.

### 3.3.1 Line Representation in 3D

Suppose, only line features are considered. Thus the scene consists of a set of lines $\mathbf{L} = \{\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_n\}$ and a set of Cameras $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_m\}$. In chapter 1.3.8 the Plücker Lines $\mathbf{L}$ where introduced. The matrix $\mathbf{L}$ is a $4 \times 4$ skew symmetric matrix as described in equation 1.12. It can be computed as the join of two points in 3-space according to equation 1.13. This representation of a line in 3D has 6 parameters, but only 4 degrees of freedom, as visualised in figure 1.2.
Another way of representing the Plücker lines are the Plücker Line coordinates. Suppose, again, $\mathbf{A}$ and $\mathbf{B}$ are points in 3-space, then the two 3-vectors $\mathbf{a}$, $\mathbf{b}$ can be computed according to

$$\mathbf{a} = \mathbf{A} \times \mathbf{B} \qquad \mathbf{b} = \mathbf{B} - \mathbf{A}. \tag{3.23}$$

The 6-Vector $\boldsymbol{\mathcal{L}}$ is composed of the 3-vectors $\mathbf{a}$ and $\mathbf{b}$.

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \\ b_1 - a_1 \\ b_2 - a_2 \\ b_3 - a_3 \end{bmatrix} \tag{3.24}$$

The 6 elements of the vector $\boldsymbol{\mathcal{L}}$ correspond to the 6 elements of the Plücker matrix $\mathbf{L}$ from equation 1.12 according to

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} l_{23} \\ -l_{13} \\ l_{12} \\ -l_{14} \\ -l_{24} \\ -l_{34} \end{bmatrix} = \begin{bmatrix} l_4 \\ -l_2 \\ l_1 \\ -l_3 \\ -l_5 \\ -l_6 \end{bmatrix}. \tag{3.25}$$

Thus, the Plücker line coordinate representation $\boldsymbol{\mathcal{L}}$ can be converted easily into the Plücker matrix $\mathbf{L}$ and vice versa.
To project the Plücker line coordinates $\boldsymbol{\mathcal{L}}$ to the image plane, an adapted $3 \times 6$ projection matrix $\boldsymbol{\mathcal{P}}$ can be formed according to

$$\boldsymbol{\mathcal{P}} = \left[ det\left(\mathbf{M}\right) \mathbf{M}^{-\top} \mid [\mathbf{p}_4]_\times \mathbf{M} \right] \tag{3.26}$$

where the camera matrix $\mathbf{P} = [\mathbf{M} \mid \mathbf{p}_4]$ and $[\mathbf{p}_4]_\times$ is the skew-symmetric matrix of the vector $\mathbf{p}_4$. Thus the projected line $\mathbf{l}$ in the image is determined according to

$$\mathbf{l} = \boldsymbol{\mathcal{P}} \boldsymbol{\mathcal{L}}. \tag{3.27}$$

As well as the camera matrix $\mathbf{P}$ the projection matrix $\boldsymbol{\mathcal{P}}$ has six parameters, three for translation and three for rotation. Taking the derivative of $\boldsymbol{\mathcal{P}}$ with respect to the parameter vector $\mathbf{s}_p = (t_x, t_y, t_z, v_x, v_y, v_z)^\top$ leads to the Jacobian

$$\mathbf{J}_{\boldsymbol{\mathcal{P}}} = \frac{\partial \boldsymbol{\mathcal{P}}}{\partial \mathbf{s}_p}. \tag{3.28}$$

The pose update takes place in a similar way as for the camera matrix $\mathbf{P}$, described in chapter 3.1. The $3 \times 4$ matrix $\tilde{\mathbf{P}}$ is the camera matrix of the last time step. $\mathbf{E}$ is of size $4 \times 4$ and represents the pose update. Thus, $\mathbf{P} = \tilde{\mathbf{P}}\mathbf{E}$. Now $\boldsymbol{\mathcal{P}}$ is computed according to equation 3.26. The derivative of $\boldsymbol{\mathcal{P}}$ with respect to the translational and rotational parameters is

$$\begin{aligned}
\left.\frac{\partial \boldsymbol{\mathcal{P}}}{\partial t_x}\right|_{t_x=0} &= \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_3 & -\boldsymbol{\mathcal{P}}_2 \end{bmatrix} \\
\left.\frac{\partial \boldsymbol{\mathcal{P}}}{\partial t_y}\right|_{t_y=0} &= \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & -\boldsymbol{\mathcal{P}}_3 & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_1 \end{bmatrix} \\
\left.\frac{\partial \boldsymbol{\mathcal{P}}}{\partial t_z}\right|_{t_z=0} &= \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_2 & -\boldsymbol{\mathcal{P}}_1 & \mathbf{0}_{3\times1} \end{bmatrix} \\
\left.\frac{\partial \boldsymbol{\mathcal{P}}}{\partial v_x}\right|_{v_x=0} &= \begin{bmatrix} \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_3 & -\boldsymbol{\mathcal{P}}_2 & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_6 & -\boldsymbol{\mathcal{P}}_5 \end{bmatrix} \\
\left.\frac{\partial \boldsymbol{\mathcal{P}}}{\partial v_x}\right|_{v_x=0} &= \begin{bmatrix} -\boldsymbol{\mathcal{P}}_3 & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_1 & -\boldsymbol{\mathcal{P}}_6 & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_4 \end{bmatrix} \\
\left.\frac{\partial \boldsymbol{\mathcal{P}}}{\partial v_x}\right|_{v_x=0} &= \begin{bmatrix} \boldsymbol{\mathcal{P}}_2 & -\boldsymbol{\mathcal{P}}_1 & \mathbf{0}_{3\times1} & \boldsymbol{\mathcal{P}}_5 & -\boldsymbol{\mathcal{P}}_4 & \mathbf{0}_{3\times1} \end{bmatrix}
\end{aligned} \tag{3.29}$$

$\boldsymbol{\mathcal{P}}_i$ denotes the $i^{th}$ column of the matrix $\boldsymbol{\mathcal{P}}$.

### 3.3.2 Orthonormal Representation Lines

The Orthonormal Representation makes it possible to describe a line in 3D with $4$ parameters, equal to the number of degrees of freedom. This is important for the Bundle Adjustment procedure. This approach follows Bartoli and Sturm in [4] and [57]. They propose a method of representing a line in 3D as the combination of a rotation in 3-space and a rotation in 2-space is proposed. Since a rotation in 3-space has 3 parameters and a rotation in 2-space has $1$ parameter,the line can be represented with $4$ parameters, that is a representation with the minimal number of parameters for a line in the three dimensional space.
In Lie algebra a rotation is represented by the $SO(3)$ group, a rotation in 2D is represented by the $SO(2)$ group. Thus, the line can represented as combination of Lie groups according to $(\mathbf{U}, \mathbf{W}) \in SO(3) \times SO(2)$.

#### Rotation in 2-space

The orthonormal representation for the $SO(2)$ Group represents a rotation in 2D and depends on one single parameter $\theta$. Thus, a matrix that corresponds to the $SO(2)$ Group is an

orthogonal matrix of dimensions $2 \times 2$. Let the homogeneous 2-vector $\boldsymbol{\sigma}$ be defined up to a scale factor, thus it has one degree of freedom. Suppose $\mathbf{W}$ is a matrix corresponding to the $SO(2)$ group, then

$$\mathbf{W} = \frac{1}{\|\boldsymbol{\sigma}\|} \begin{bmatrix} \sigma_1 & -\sigma_2 \\ \sigma_2 & \sigma_1 \end{bmatrix} \tag{3.30}$$

where $\boldsymbol{\sigma} \in \mathbb{P}^1$ is the first column of the matrix $\mathbf{W}$, normalised such that $\|\boldsymbol{\sigma}\| = 1$.
A local update step of the parameter $\theta$ is $\mathbf{W} = \mathbf{W}_0\mathbf{R}(\theta)$ where $\mathbf{W}_0$ is the rotation matrix of the last time step and the update matrix $\mathbf{R}(\theta)$ is a rotation of $\theta$ in the 2D space. The Jacobi Matrix of $\boldsymbol{\sigma}$ with respect to the parameter $\theta$ evaluated at $\theta_0 = 0$ is

$$\left.\frac{\partial \boldsymbol{\sigma}}{\partial \theta}\right|_{\theta_0=0} = \left.\frac{\partial \mathbf{w}_1}{\partial \theta}\right|_{\theta_0=0} = \begin{bmatrix} -\sigma_2 \\ \sigma_1 \end{bmatrix} = \mathbf{w}_2. \tag{3.31}$$

$\mathbf{w}_i$ is the $i^{th}$ column of the matrix $\mathbf{W}$. This result is needed later for the representation of a line in 3-space.

**Rotation in 3-space**

The orthonormal representation for the $SO(3)$ Group represents a rotation in 3D and depends on the parameter vector $\mathbf{v}$ that is composed of three parameters $v_x, v_y, v_z$. The vector $\mathbf{v}$ denotes the rotation axis, the norm $\|\mathbf{v}\|$ of the vector is the rotation angle. This axis angle representation of a rotation in 3-space is described in chapter 3.1. Suppose $\mathbf{U} \in SO(3)$, that is a $3 \times 3$ Matrix and denotes a rotation in the 3D space, then $\mathbf{U} = \mathbf{U}_0\mathbf{R}(v)$ where $\mathbf{U}_0$ is the rotation matrix that describes the orientation of the previous step and the rotation matrix $\mathbf{R}$ is the rotation update. The rotation matrix $\mathbf{R}$ introduced in equation 3.6 is determined according to $\mathbf{R} = \mathbf{I} + [\mathbf{w}]_\times sin(\theta) + [\mathbf{w}]_\times^2 (1 - cos(\theta))$ where $\mathbf{w}$ is the unit vector of $\mathbf{v}$ with $\|w\| = 1$. The $3 \times 3$ matrix $[\mathbf{w}]_\times$ is the skew symmetric matrix of $\mathbf{w}$. $\theta$ is the norm of the vector $\mathbf{v}$ and represents the rotation angle around the axis $\mathbf{w}$.
The Jacobi Matrix of $\mathbf{U}$ with respect to the parameter vector $\mathbf{v}$, evaluated at $\mathbf{v}_0 = \mathbf{0}_{3\times1}$ is given by

$$\left.\frac{\partial \mathbf{U}}{\partial \mathbf{v}}\right|_{v_0=0} = \left( \left.\frac{\partial \mathbf{U}}{\partial v_x}\right|_{v_{x_0}=0} \quad \left.\frac{\partial \mathbf{U}}{\partial v_y}\right|_{v_{y_0}=0} \quad \left.\frac{\partial \mathbf{U}}{\partial v_z}\right|_{v_{z_0}=0} \right) \tag{3.32}$$

where

$$\left.\frac{\partial \mathbf{U}}{\partial v_x}\right|_{v_{x_0}=0} = \left.\frac{\partial \mathbf{U}_0\mathbf{R}(v)}{\partial v_x}\right|_{v_{x_0}=0} = \mathbf{U}_0 \left.\frac{\partial \mathbf{R}(v)}{\partial v_x}\right|_{v_{x_0}=0} = \mathbf{U}_0 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{3.33}$$

Thus,

$$\left.\frac{\partial \mathbf{U}}{\partial v_x}\right|_{v_{x_0}=0} = \begin{bmatrix} \mathbf{0}_{3\times1} & \mathbf{u}_3 & -\mathbf{u}_2 \end{bmatrix}. \tag{3.34}$$

the vector $\mathbf{u}_i$ denotes the $i^{th}$ column of the matrix $\mathbf{U}$. Thus, the derivative of $\mathbf{U}$ with respect to the other parameters is

$$\left.\frac{\partial \mathbf{U}}{\partial v_y}\right|_{v_{y_0}=0} = \begin{bmatrix} -\mathbf{u}_3 & \mathbf{0}_{3\times1} & \mathbf{u}_1 \end{bmatrix}. \tag{3.35}$$

$$\left.\frac{\partial \mathbf{U}}{\partial v_z}\right|_{v_{z_0}=0} = \begin{bmatrix}\mathbf{u}_2 & -\mathbf{u}_1 & \mathbf{0}_{3\times 1}\end{bmatrix}. \tag{3.36}$$

**Minimal representation lines**

As noted previously, the orthonormal representation of a line $\boldsymbol{\mathcal{L}}$ is $(\mathbf{U}, \mathbf{W}) \in SO(3) \times SO(2)$. Referring to equation 3.24, $\boldsymbol{\mathcal{L}} = \begin{bmatrix}\mathbf{a}^\top & \mathbf{b}^\top\end{bmatrix}^\top$ is the $6 \times 1$ vector that contains the Plücker coordinates. Then the matrix $\mathbf{C}_{3\times 2} = [\mathbf{a}\ \mathbf{b}]$ can be decomposed into

$$\mathbf{C} = \mathbf{U}_{3\times 3}\boldsymbol{\Sigma}_{3\times 2} = \begin{bmatrix}\frac{\mathbf{a}}{\|\mathbf{a}\|} & \frac{\mathbf{b}}{\|\mathbf{b}\|} & \frac{\mathbf{a}\times\mathbf{a}}{\|\mathbf{a}\times\mathbf{a}\|}\end{bmatrix}\begin{bmatrix}\|\mathbf{a}\| & \\ & \|\mathbf{b}\| \end{bmatrix} = \mathbf{U}\begin{bmatrix}\sigma_1 & \\ & \sigma_2\end{bmatrix} \tag{3.37}$$

applying the QR decomposition. $\mathbf{U} \in SO(3)$ is a orthogonal matrix and $\boldsymbol{\Sigma}$ is an upper triangular matrix. The matrix $\mathbf{W} \in SO(2)$ from equation 3.30 can be composed by the two non-zero elements of the matrix $\boldsymbol{\Sigma}$, where the vector $\boldsymbol{\sigma} = [\|\mathbf{a}\|\ \|\mathbf{b}\|]^\top$. Thus, the 4 parameters of the line $\mathbf{L}$ are defined by the 3 parameters of the matrix $\mathbf{U}$ and the one parameter of matrix $\mathbf{W}$.
The conversion of the orthonormal represented line $(\mathbf{U}, \mathbf{W})$ back to the Plücker coordinates $\boldsymbol{\mathcal{L}}$ can be done by

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix}w_{11}\mathbf{u}_1 \\ w_{21}\mathbf{u}_2\end{bmatrix} \tag{3.38}$$

where $\mathbf{u}_i$ is the $i^{th}$ column of the matrix $\mathbf{U}$.
According to the updates of $\mathbf{W} \in SO(2)$ and $\mathbf{U} \in SO(3)$, the line can be updated by the parameter vector $\mathbf{p} = [\mathbf{v}\quad \theta]$. The Jacobian $\mathbf{J}$ of the line $\boldsymbol{\mathcal{L}}$ with respect to the parameter vector $\mathbf{p}$ evaluated at $\mathbf{p}_0 = \mathbf{0}$ is a $6 \times 4$ matrix and can be determined according to

$$\begin{aligned}\mathbf{J} &= \begin{bmatrix}\left.\frac{\partial \boldsymbol{\mathcal{L}}}{\partial v_x}\right|_{\mathbf{p}_0} & \left.\frac{\partial \boldsymbol{\mathcal{L}}}{\partial v_y}\right|_{\mathbf{p}_0} & \left.\frac{\partial \boldsymbol{\mathcal{L}}}{\partial v_z}\right|_{\mathbf{p}_0} & \left.\frac{\partial \boldsymbol{\mathcal{L}}}{\partial \theta}\right|_{\mathbf{p}_0}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{0} & -\sigma_1\mathbf{u}_3 & \sigma_1\mathbf{u}_2 & -\sigma_2\mathbf{u}_1 \\ \sigma_2\mathbf{u}_3 & \mathbf{0} & -\sigma_2\mathbf{u}_1 & \sigma_1\mathbf{u}_2\end{bmatrix}\end{aligned} \tag{3.39}$$

## 3.4 Parameter Optimisation

In the process of bundle adjustment, the approaches derived in the previous chapters will be applied to improve the estimation of the points and lines as well as the camera poses of the reconstructed scene. The parameter vector $\mathbf{s}$ with the parameters that has to be refined consists of the 3 inhomogeneous coordinates of each scene point $\mathbf{X}_i$, the 4 parameters of each line $\mathbf{L}_k$ in the scene and the 6 parameters of each camera $\mathbf{P}_j$ involved in the scene. Thus, the parameter vector is of the size $3n_x + 4n_l + 6m$ where $n_x$ is the number or points, $n_l$ is the number of lines and $m$ stands for the number of cameras.
From algorithm 2.8 in chapter 2 an initial estimation of the points, lines and cameras is

obtained. Thus, the parameter vector **s** can be composed according to

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_x \\ \mathbf{s}_l \\ \mathbf{s}_p \end{bmatrix} \tag{3.40}$$

where $\mathbf{s}_x$ stands for the parameter vector of the points, $\mathbf{s}_y$ is the parameter vector of the lines and in $\mathbf{s}_p$ the parameters of the cameras are present. The vector $\mathbf{s}_x$ contains the euclidean coordinates $\{X, Y, Z\}$ of all the points $\mathbf{X}_i$ according to

$$\mathbf{s}_x = \left[X_1, Y_1, Z_1, ..., X_{n_x}, Y_{n_x}, Z_{n_x}\right]^\top. \tag{3.41}$$

The vector $\mathbf{s}_l$ is composed of the 4 parameters of each line $\mathbf{L}_k$. These 4 parameters $\{a, b, c, d\}$ can be obtained from equation 3.37 as described in chapter 3.3.2. Thus, the vector $\mathbf{s}_l$ is build according to

$$\mathbf{s}_l = \left[a_1, b_1, c_1, d_1, ..., a_{n_l}, b_{n_l}, c_{n_l}, d_{n_l}\right]^\top. \tag{3.42}$$

The vector for the camera parameters $\mathbf{s}_p$ is made up of the six parameters $\{t_x, t_y, t_z, v_x, v_y, v_z\}$ for each camera. These parameters are obtained from the camera matrix **R** applying the equation 3.3, 3.4 and 3.5 in chapter 3.1. Thus, $\mathbf{s}_p$ has the form

$$\mathbf{s}_p = \left[t_{x_1}, t_{y_1}, t_{z_1}, v_{x_1}, v_{y_1}, v_{z_1}, ..., t_{x_m}, t_{y_m}, t_{z_m}, v_{x_m}, v_{y_m}, v_{z_m}\right]^\top. \tag{3.43}$$

The error vector $\boldsymbol{\epsilon}$, that has to be minimised is composed of the reprojection error or points and lines in the images.

**Reprojection error points:** Let $\mathbf{x}_{ij} = (x_{ij}, y_{ij}, 1)^\top$ be a homogeneous vector of a point measured in the image and $\hat{\mathbf{x}}_{ij} = \mathbf{P}_j\mathbf{X}_i = (\hat{x}_{ij}, \hat{y}_{ij}, 1)^\top$ the corresponding reprojected point of the reconstruction. The reprojection error $d_{ij}$, that is visualised in figure 3.1 is the euclidean distance between $\hat{\mathbf{x}}_{ij}$ and $\mathbf{x}_{ij}$ and is computed according to

$$d_{ij}(\hat{\mathbf{x}}_{ij}, \mathbf{x}_{ij}) = \sqrt{(\hat{x}_{ij} - x_{ij})^2 + (\hat{y}_{ij} - y_{ij})^2}. \tag{3.44}$$

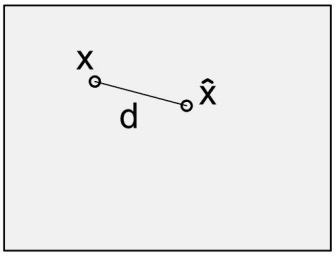



Figure 3.1: ***Reprojection error points:*** *The point* $\boldsymbol{x}$ *is the measured point in the image,* $\hat{\boldsymbol{x}}$ *is the reprojected point. The reprojection error* $d$ *is the euclidean distance between* $\boldsymbol{x}$ *and* $\hat{\boldsymbol{x}}$*.*

Thus, the error vector $\boldsymbol{\epsilon}_{x_{ij}}$ for this point is composed of the two euclidean coordinates of $\mathbf{x}_{ij}$ and $\hat{\mathbf{x}}_{ij}$ according to

$$\boldsymbol{\epsilon}_{x_{ij}} = \begin{bmatrix} \hat{x}_{ij} - x_{ij} \\ \hat{y}_{ij} - y_{ij} \end{bmatrix} \tag{3.45}$$

where the norm of the vector $\boldsymbol{\epsilon}_{ij}$ is the reprojection error $d_{ij}(\hat{\mathbf{x}}_{ij}, \mathbf{x}_{ij})$. Thus, the euclidean distence can be rewritten as

$$d_{ij}(\hat{\mathbf{x}}_{ij}, \mathbf{x}_{ij}) = \|\boldsymbol{\epsilon}_{ij}\| . \tag{3.46}$$

The total reprojection error $e_x$ is computed as the sum of squares of the single reprojection errors of all correspondences over all images according to

$$e_x = \sum_{i=1}^{n_x} \sum_{j=1}^{m} d_{ij} \left(\hat{\mathbf{x}}_{ij}, \mathbf{x}_{ij}\right)^2 . \tag{3.47}$$

Let $\mathbf{x}$ be the vector that contains the inhomogeneous coordinates of the measured points $\mathbf{x}_{ij}$ according to

$$\mathbf{x} = \left(x_{1,1}, y_{1,1}, ..., x_{n,m}, y_{n,m}\right)^\top \tag{3.48}$$

Since the homogeneous point vector $\hat{\mathbf{x}}_{ij}$ of the reprojected points is obtained according to $\hat{\mathbf{x}}_{ij} = \mathbf{P}_j \mathbf{X}_i = \left(\mathbf{P}_j^1 \mathbf{X}_i, \mathbf{P}_j^2 \mathbf{X}_i, \mathbf{P}_j^3 \mathbf{X}_i\right)^\top$, the inhomogeneous coordinates of the reprojected points are of the form $\left(\hat{x}_{1,1}, \hat{y}_{1,1}\right)^\top = \left(\mathbf{P}_j^1 \mathbf{X}_i / \mathbf{P}_j^3 \mathbf{X}_i, \mathbf{P}_j^2 \mathbf{X}_i / \mathbf{P}_j^3 \mathbf{X}_i\right)^\top$ where $\mathbf{P}_j^k$ denotes the $k^{th}$ row of the $j^{th}$ camera matrix. Thus, the vector that contains the reprojected point coordinates is

$$\hat{\mathbf{x}}(s) = \left(\hat{x}_{1,1}, \hat{y}_{1,1}, ..., \hat{x}_{n,m}, \hat{y}_{n,m}\right)^\top . \tag{3.49}$$

The error vector $\boldsymbol{\epsilon}_x$ can be assembled by the vectors $\hat{\mathbf{x}}(s)$ and $\mathbf{x}$ according to

$$\boldsymbol{\epsilon}_x = \left[\hat{\mathbf{x}}(\mathbf{s}) - \mathbf{x}\right] . \tag{3.50}$$

Thus, the total reprojection error $e_x$ from equation 3.47 can be reformulated as

$$e_x = \|\boldsymbol{\epsilon}_x\|^2 = \|\hat{\mathbf{x}}(\mathbf{s}) - \mathbf{x}\|^2 . \tag{3.51}$$

**Reprojection error lines:** Let $\mathbf{l}_{kj}$ be a line vector of a line measured in the image and $\hat{\mathbf{l}}_{kj}$ the corresponding reprojected line of the scene line $\boldsymbol{\mathcal{L}}_k$ in the camera $\boldsymbol{\mathcal{P}}_j$ according to $l = \boldsymbol{\mathcal{P}}_j \boldsymbol{\mathcal{L}}_k$. The vector $\boldsymbol{\mathcal{L}}_k$ is the $k^{th}$ line containing the 6 Plücker parameter according to equation 3.25 and $\boldsymbol{\mathcal{P}}_j$, that is obtained from equation 3.26 is the $j^{th}$ projection matrix that projects the Plücker parameter vector into the image plane. To compute the reprojection error between $\mathbf{l}_{kj}$ and $\hat{\mathbf{l}}_{kj}$ the intersection points of line $\mathbf{l}_{kj}$ with the image borders must be obtained. This homogeneous point vectors are denoted as $\mathbf{a}_{kj} = \left(x_{a_{kj}}, y_{a_{kj}}, 1\right)^\top$ and $\mathbf{b}_{kj} = \left(x_{b_{kj}}, y_{b_{kj}}, 1\right)^\top$. The perpendicular distances $t_{a_{kj}}$ and $t_{b_{kj}}$ of the points $\mathbf{a}_{kj}$ and $\mathbf{b}_{kj}$ respectively to the line $\hat{\mathbf{l}}_{kj}$ can be computed according to the scalar products

$$t_{a_{kj}} = \mathbf{a}_{kj}^\top \hat{\mathbf{l}}_{kj} \tag{3.52}$$

$$t_{b_{kj}} = \mathbf{b}_{kj}^\top \hat{\mathbf{l}}_{kj} \tag{3.53}$$

for the line $\hat{\mathbf{l}}_{kj}$ normalised such that the norm of the first two elements of the line is equal to 1, as noted in chapter 1.3.7. Thus, the error vector for the line is represented by the 2-vector

$$\boldsymbol{\epsilon}_{l_{kj}} = \begin{bmatrix} t_{a_{kj}} \\ t_{b_{kj}} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{kj}^{\top} \hat{\mathbf{l}}_{kj} \\ \mathbf{b}_{kj}^{\top} \hat{\mathbf{l}}_{kj} \end{bmatrix}. \tag{3.54}$$

The reprojection error $e_{kj}(\hat{\mathbf{l}}_{kj}, \mathbf{l}_{kj})$ for a line $\mathbf{l}_{kj}$ in image $j$ is defined in a similar way as the reprojection error for points as the norm of the vector $\boldsymbol{\epsilon}_{l_{kj}}$ according to

$$e_{kj}(\hat{\mathbf{l}}_{kj}, \mathbf{l}_{kj}) = \left\| \boldsymbol{\epsilon}_{l_{kj}} \right\|. \tag{3.55}$$

The reprojection error for lines is illustrated in figure 3.2.

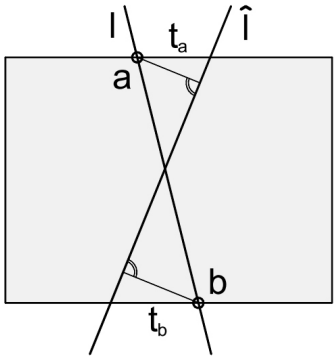



Figure 3.2: ***Reprojection error lines*** *The line* $\boldsymbol{l}$ *is the line measured in the image. The points* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *are the points where the line intersects the image border.* $\hat{\boldsymbol{l}}$ *is the reprojected line. The distance* $t_a$ *is the perpendicular distance of the point* $a$ *to line* $\hat{\boldsymbol{l}}$*, the distance* $t_b$ *is the perpendicular distance between the point* $\boldsymbol{b}$ *and the line* $\hat{\boldsymbol{l}}$*. The reprojection error of the lines is determined from the distances* $t_a$ *and* $t_b$*.*

The error vector $\boldsymbol{\epsilon}_l$ for all lines is assembled accumulating the error vectors $\boldsymbol{\epsilon}_{l_{kj}}$ of the single lines according to

$$\boldsymbol{\epsilon}_l = \left[ \boldsymbol{\epsilon}_{l_{11}}, ..., \boldsymbol{\epsilon}_{l_{n_l m}} \right]^{\top}. \tag{3.56}$$

Thus, the total reprojection error $e_l$ is computed as the sum of squares of the single reprojection errors of the lines over all images according to

$$e_l = \sum_{k=1}^{n_l} \sum_{j=1}^{m} t_{kj} \left( \hat{\mathbf{l}}_{kj}, \mathbf{l}_{kj} \right)^2 = \left\| \boldsymbol{\epsilon}_l \right\|^2. \tag{3.57}$$

**Combined reprojection error:** The two error vectors $\boldsymbol{\epsilon}_x$ and $\boldsymbol{\epsilon}_l$ for the reprojection error of points and lines can be combined to a single error vector $\boldsymbol{\epsilon}$.

$$\boldsymbol{\epsilon} = \begin{bmatrix} \boldsymbol{\epsilon}_x \\ \boldsymbol{\epsilon}_l \end{bmatrix} \tag{3.58}$$

Thus, the total reprojection error $e$ is computed as

$$e = \left\| \boldsymbol{\epsilon} \right\|^2. \tag{3.59}$$

**Levenberg-Marquardt Algorithm:** The error vector $\boldsymbol{\epsilon}$ from equation 3.58 and the parameter vector $\mathbf{s}$ from equation 3.40 are required to optimise the parameters of $\mathbf{s}$ minimising the error $e$. The minimisation is a non linear least squares problem and can be solved with a non linear minimisation method such as the Levenberg-Marquardt algorithm. Tordable in [61] describes how a Gauss-Newton algorithm is applied to a bundle adjustment problem. The concept for the algorithm is obtained from there. In a similar way Hartley and Zisserman in [25] describe a Bundle-Adjustment problem and propose the Levenberg-Marquardt algorithm, that is a slight variation on the Gauss-Newton method. From there the augmentation to a Levenberg-Marquardt algorithm is borrowed.
From the reconstruction process an initial estimation for the parameter vector $\mathbf{s}$ is obtained and represented by $\mathbf{s}_0$. Consider the function of the error $e = \|\boldsymbol{\epsilon}\|^2$ Since the square of the norm of a vector is equal to the square of a vector $e = \boldsymbol{\epsilon}^2$. This function has a minimum in the elements of the parameter vector $\mathbf{s}$. Thus

$$\frac{\partial e(\mathbf{s})}{\partial \mathbf{s}} = 2\boldsymbol{\epsilon}(\mathbf{s})\frac{\partial \boldsymbol{\epsilon}(\mathbf{s})}{\partial \mathbf{s}} = 0. \tag{3.60}$$

The derivation $\partial\boldsymbol{\epsilon}(\mathbf{s})/\partial\mathbf{s}$ is different for points and lines and must be considered separately.

**Minimisation of the point error:** Deriving the error function $e_x$ of the point error $\boldsymbol{\epsilon}_x = \hat{\mathbf{x}}(\mathbf{s}) - \mathbf{x}$ with respect to the point parameters $\mathbf{s}_x$ leads to

$$\frac{\partial e_x}{\partial \mathbf{s}_x} = \left(\hat{\mathbf{x}}(\mathbf{s}_x) - \mathbf{x}\right)^\top \frac{\partial \hat{\mathbf{x}}(\mathbf{s}_x)}{\partial \mathbf{s}_x} = 0. \tag{3.61}$$

To solve the equation, an approximation can be taken into consideration expressing $\hat{\mathbf{x}}(\mathbf{s})$ as a Taylor expansion

$$\hat{\mathbf{x}}(\mathbf{s}_x) \simeq \hat{\mathbf{x}}(\mathbf{s}_{x0}) + \frac{\partial \hat{\mathbf{x}}(\mathbf{s}_x)}{\partial \mathbf{s}_x}\boldsymbol{\delta}_x. \tag{3.62}$$

$\mathbf{s}_{x0}$ is the parameter vector at the last step, $\partial\hat{\mathbf{x}}(\mathbf{s}_{x0})/\partial\mathbf{s}_x$ is the Jacobian $\mathbf{J}_x$. Thus, the equation can be expressed in the form

$$\hat{\mathbf{x}}(\mathbf{s}_x) = \hat{\mathbf{x}}(\mathbf{s}_{x0}) + \mathbf{J}_x\boldsymbol{\delta}_x. \tag{3.63}$$

The vector $\boldsymbol{\delta}_x$ is the update-vector for the parameter vector $\mathbf{s}_x$ such that $\mathbf{s}_x = \mathbf{s}_{x0} + \boldsymbol{\delta}_x$. Applying equation 3.63 into equation 3.61 leads to

$$\begin{aligned}
\left(\hat{\mathbf{x}}(\mathbf{s}_{x0}) + \mathbf{J}_x\boldsymbol{\delta}_x - \mathbf{x}\right)\mathbf{J}_x &= 0 \\
\mathbf{J}_x^\top\left(\hat{\mathbf{x}}(\mathbf{s}_{x0}) + \mathbf{J}_x\boldsymbol{\delta}_x - \mathbf{x}\right) &= 0 \\
\mathbf{J}_x^\top\hat{\mathbf{x}}(\mathbf{s}_{x0}) + \mathbf{J}_x^\top\mathbf{J}_x\boldsymbol{\delta}_x - \mathbf{J}_x^\top\mathbf{x} &= 0 \\
\mathbf{J}_x^\top\left(\hat{\mathbf{x}}(\mathbf{s}_{x0}) - \mathbf{x}\right) + \mathbf{J}_x^\top\mathbf{J}_x\boldsymbol{\delta}_x &= 0.
\end{aligned} \tag{3.64}$$

Thus, the update vector $\boldsymbol{\delta}_x$ can be determined according to

$$\begin{aligned}
\boldsymbol{\delta}_x &= -\left(\mathbf{J}_x^\top\mathbf{J}_x\right)^{-1}\left(\mathbf{J}_x^\top\left(\hat{\mathbf{x}}(\mathbf{s}_{x0}) - \mathbf{x}\right)\right) \\
\boldsymbol{\delta}_x &= -\left(\mathbf{J}_x^\top\mathbf{J}_x\right)^{-1}\mathbf{J}_x^\top\boldsymbol{\epsilon}_{x0}.
\end{aligned} \tag{3.65}$$

**Minimisation of the line error:** The derivation of the error function of the line error $e_l$ with respect to the line parameters $\mathbf{s}_l$ is done in a similar way. From equation 3.57 the error vector $\boldsymbol{\epsilon}_l$ is obtained. Let the error vector consist of $2n$ elements of the form $\mathbf{a}_i^\top \mathbf{l}_i$ and $\mathbf{b}_i^\top \mathbf{l}_i$ respectively, thus $\boldsymbol{\epsilon}_l$ is of the form

$$\boldsymbol{\epsilon}_l = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{l}_1 \\ \mathbf{b}_1^\top \mathbf{l}_1 \\ \vdots \\ \mathbf{a}_i^\top \mathbf{l}_i \\ \mathbf{b}_i^\top \mathbf{l}_i \\ \vdots \\ \mathbf{a}_n^\top \mathbf{l}_n \\ \mathbf{b}_n^\top \mathbf{l}_n \end{bmatrix}. \tag{3.66}$$

The error, that has to be minimised is

$$e_l = \|\boldsymbol{\epsilon}\|_l^2 = \sum_{i=1}^{n} (\mathbf{a}_i^\top \mathbf{l}_i)^2 + (\mathbf{b}_i^\top \mathbf{l}_i)^2 = \sum_{j=1}^{2n} (\mathbf{x}_j^\top \mathbf{l}_j)^2 = \sum_{j=1}^{2n} e_j^2. \tag{3.67}$$

The point $\mathbf{x}_j$ denotes a point $\mathbf{a}_i$ or $\mathbf{b}_i$ of the $i^{th}$ element of vector. Thus, the derivative of $e_l$ with respect to $\mathbf{s}_l$ is

$$\frac{\partial e_l}{\partial \mathbf{s}_l} = 2 \sum_{j=1}^{2n} e_j \frac{\partial e_j}{\partial \mathbf{s}_l} = 0. \tag{3.68}$$

The expression $e_j$ may be replaced by the Taylor expansion

$$e_j = e_{0j} + \frac{\partial e_j}{\partial \mathbf{s}_l} \boldsymbol{\delta}_l \tag{3.69}$$

where $^{\partial e_j}/_{\partial \mathbf{s}_l}$ is the Jacobian $\mathbf{J}_l^j$, that is the $j^{th}$ row of the Jacobian $\mathbf{J}_l$. Thus, the equation can be rewritten as

$$e_j = e_{0j} + \mathbf{J}_l^j \boldsymbol{\delta}_l. \tag{3.70}$$

Substituting this equation into equation 3.68 lead to

$$\begin{aligned} \sum_{j=1}^{2n} \left( e_{0j} + \mathbf{J}_l^j \boldsymbol{\delta}_l \right) \mathbf{J}_l^j &= 0 \\ \sum_{j=1}^{2n} \mathbf{J}_l^j e_{0j} + \sum_{j=1}^{2n} \mathbf{J}_l^j \mathbf{J}_l^j \boldsymbol{\delta}_l &= 0 \\ \mathbf{J}_l^\top \boldsymbol{\epsilon}_{l0} + \mathbf{J}_l^\top \mathbf{J}_l \boldsymbol{\delta}_l &= 0 \end{aligned} \tag{3.71}$$

The update vector $\boldsymbol{\delta}_l$ can be determined according to

$$\boldsymbol{\delta}_l = - \left( \mathbf{J}_l^\top \mathbf{J}_l \right)^{-1} \mathbf{J}_l^\top \boldsymbol{\epsilon}_{l0} \tag{3.72}$$

The Jacobian $\mathbf{J}_l$ is the derivative of the error vector $\boldsymbol{\epsilon}_l$ with respect to the line parameter $\mathbf{s}_l$. Lets consider the Jacobian $\mathbf{J}_l^i$, the derivative of the error vector of one line $l_i$ with respect to $\mathbf{s}_l$.

$$\boldsymbol{\epsilon}_i = \begin{bmatrix} \mathbf{a}_i^\top \mathbf{l}_i \\ \mathbf{b}_i^\top \mathbf{l}_i \end{bmatrix}. \tag{3.73}$$

Thus, the Jacobian $\mathbf{J}_l^i$, that represents the row $i$ to $i+1$ of the Jacobian $\mathbf{J}_l$ can be computed according to

$$\mathbf{J}_l^i = \frac{\partial \boldsymbol{\epsilon}_i}{\partial \mathbf{s}_l} = \begin{bmatrix} \mathbf{a}_i^\top \mathbf{l}_i \\ \mathbf{b}_i^\top \mathbf{l}_i \end{bmatrix} \frac{\partial \mathbf{l}_i}{\partial \mathbf{s}_l} = \begin{bmatrix} \mathbf{a}_i^\top \mathbf{l}_i \\ \mathbf{b}_i^\top \mathbf{l}_i \end{bmatrix} \hat{\mathbf{J}}_l^i. \tag{3.74}$$

The derivative of the line $\mathbf{l}_i$ is denoted as $\hat{\mathbf{J}}_l^i$. Thus, the Jacobian $\hat{\mathbf{J}}_l$ is the derivative of all lines $\mathbf{l} = (\mathbf{l}_1, ..., \mathbf{l}_n)^\top$ with respect to the line parameter $\mathbf{s}_l$ and is composed of the single Jacobians $\hat{\mathbf{J}}_l^i$.

**Minimisation of the total error:** The equations 3.65 and 3.73 that represents the update vectors $\boldsymbol{\delta}_x$, $\boldsymbol{\delta}_l$ for the point and line parameters are quite similar and differ only in the index of the expressions such as error vector, Jacobian and update vector. Combining the error vectors $\boldsymbol{\epsilon}_x$,$\boldsymbol{\epsilon}_l$ and the update vectors $\boldsymbol{\delta}_x$,$\boldsymbol{\delta}_l$ for points and lines according to

$$\boldsymbol{\epsilon} = \begin{bmatrix} \boldsymbol{\epsilon}_x \\ \boldsymbol{\epsilon}_l \end{bmatrix}, \qquad \boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_x \\ \boldsymbol{\delta}_l \end{bmatrix} \tag{3.75}$$

leads to the total update equation

$$\boldsymbol{\delta} = -\left(\mathbf{J}^\top \mathbf{J}\right)^{-1} \mathbf{J}^\top \boldsymbol{\epsilon}_0. \tag{3.76}$$

The parameter vector $\mathbf{s}_0$ with the initial estimation can be updated to obtain the new estimation $\mathbf{s}$ according to

$$\mathbf{s} = \mathbf{s}_0 + \boldsymbol{\delta}. \tag{3.77}$$

The Jacobian $\mathbf{J}$, that is composed of the Jacobians $\mathbf{J}_x$ and $\mathbf{J}_l$ is obtained using the expressions of the equations 3.21 and 3.22 for the Jacobian $\mathbf{J}_x$ and equations 3.39 and 3.29 for the Jacobian $\mathbf{J}_l$. As a reminder from equation 3.40 the parameter vector $\mathbf{s}$ is composed of $\mathbf{s} = \left[\mathbf{s}_x^\top, \mathbf{s}_l^\top, \mathbf{s}_p^\top\right]^\top$. Thus, the error function $\boldsymbol{\epsilon}$ is derived with respect to the point, line and camera parameter. Deriving the point error with respect to the line parameters gives zero. Likewise the derivation of the line error with respect to the point parameters is zero as well. Thus, the Jacobian $\mathbf{J}$ that is used to determine the total update vector $\boldsymbol{\delta}$ according equation 3.76 is composed according to

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{xx} & \mathbf{0} & \mathbf{J}_{xp} \\ \mathbf{0} & \mathbf{J}_{ll} & \mathbf{J}_{lp} \end{bmatrix}. \tag{3.78}$$

$\mathbf{J}_{xx}$ is the Jacobian that corresponds to the derivative of the error vector of the point features $\boldsymbol{\epsilon}_x$ with respect to the vector of the point parameters $\mathbf{s}_x$. It is determined using the equation 3.21. $\mathbf{J}_{xp}$ corresponds to the derivative of the error vector of the point features $\boldsymbol{\epsilon}_x$ with respect to the camera parameters $\mathbf{s}_p$. $\mathbf{J}_{xp}$ is obtained applying the equation 3.22.
$\mathbf{J}_{ll}$ is the Jacobian that corresponds to the derivative of the error vector of the lines $\boldsymbol{\epsilon}_l$ with

respect to the line parameters $\mathbf{s}_l$. It can be computed according to equation 3.39. The Jacobian $\mathbf{J}_{lp}$ is the Jacobian that corresponds to the derivative of the line parameters $\boldsymbol{\epsilon}_l$ with respect to the camera parameters $\mathbf{s}_p$ and can be obtained from equation 3.29.

Up to here the update follows the Gauss-Newton algorithm. To augment the update to a Levenberg-Marquardt iteration, the total update equation $\boldsymbol{\delta} = -\left(\mathbf{J}^\top\mathbf{J}\right)^{-1}\mathbf{J}^\top\boldsymbol{\epsilon}_0$ is replaced by the augmented equation

$$\boldsymbol{\delta} = -\left(\mathbf{J}^\top\mathbf{J} + \lambda\mathbf{I}\right)^{-1}\mathbf{J}^\top\boldsymbol{\epsilon}_0. \tag{3.79}$$

The value of $\lambda$ varies from iteration to iteration. As a typical initial value of $\lambda$ the value $10^{-3}$ times the average of the diagonal elements of the matrix $\mathbf{J}^\top\mathbf{J}$ is proposed in [25]. Suppose, the value of $\lambda$ is very small. Then the method works basically in the same way as the Gauss newton Algorithm. If $\lambda$ is large, the equation 3.79 can be approximated by $\boldsymbol{\delta} = -\lambda^{-1}\mathbf{J}^\top\boldsymbol{\epsilon}_0$. Thus, the algorithm behaves like the gradient-descent method, where the update is in the direction of the most rapid local decrease of the function with a fixed step size. One can conclude that the Levenberg-Marquardt algorithm moves smoothly between the Gauss-Newton method that converges very quickly near the minimum and the gradient-descent algorithm that ensures a slow decreasing in difficult environments.
As the initial estimation of the parameters of points $\mathbf{X}$, lines $\mathbf{L}$ and cameras $\mathbf{P}$ the results of algorithm 2.8 are used. These parameters are collected in the parameter vector $\mathbf{s}$ according to equation 3.40. Furthermore a first estimation of the error vector $\boldsymbol{\epsilon}_0$ is computed according to equation 3.58, determining the reprojection errors of points and lines. Thus, the total reprojection error $e_0$ can be computed according to equation 3.59 as the sum of squares of the single reprojection errors. The Jacobian $\mathbf{J}$ can be obtained from equation 3.78. Thus, the update vector $\boldsymbol{\delta}$ can be obtained according to equation 3.79. Using equation 3.77, the estimation of the parameters $\mathbf{s}_0$ is updated to obtain the new estimation $\mathbf{s}$. This parameter vector contains new estimations of the points, lines and cameras. Thus, computing the reprojections of points and lines, a new error vector $\boldsymbol{\epsilon}$ can be obtained that contains the new reprojection errors.
If the total reprojection error $e$, that is obtained from the error vector $\boldsymbol{\epsilon}$ according to equation 3.59, is smaller than $e_0$, the update is accepted. The parameter vector $\mathbf{s}_0$ and the error vector $\boldsymbol{\epsilon}_0$ are updated according to $\mathbf{s}_0 \leftarrow \mathbf{s}$ and $\boldsymbol{\epsilon}_0 \leftarrow \boldsymbol{\epsilon}$, the step size for the next iteration step is increased dividing $\lambda$ by the factor 10.
If $e$ is bigger than $e_0$, the update is not accepted. $\lambda$ is multiplied by the factor 10 to reduce the step size and the update vector $\boldsymbol{\delta}$ is computed again. This is repeated until the update leads to a reduced error $e$ compared to the error $e_0$.
Repeating this process the error $e$ will decrease and the estimation of points, lines and cameras will be optimised. In algorithm 3.1 the process of Bundle adjustment is summarised.

**Convergence of the algorithm:** As convergence criterion several features can be considered. It is recommended to use a combination of various conditions. A possible condition is the absolute error $e$. If $e$ falls below a threshold, the algorithm terminates. Another termination condition may be the maximum number of iterations. One can consider the absolute change of the error, that can be computed as the difference between the initial error $e_0$ and the error after the iteration step $e$ according to $e_0 - e$ as a criterion. The relative change

**Algorithm summary:**
If the initial estimations of the points **X**, lines **L** and cameras **P** are available, the algorithm optimises this scene minimising the reprojection error of points and lines in the image.

**Algorithm:**

1. Take the initial estimations of the points **X**, lines **L** and cameras **P** to form the parameter vector $\mathbf{s}_0$ according to equation 3.40.
2. Compute the reprojected points $\hat{\mathbf{x}}$ and lines $\hat{\mathbf{l}}$ according equation 3.19 and 3.27. Thus, the error vector $\boldsymbol{\epsilon}_0$ can be obtained from equation 3.50 and equation 3.66. These two vectors are assembled according to equation 3.58. The total reprojection error $e_0$ is then computed as the sum of squares of the single reprojection errors according to equation 3.59.
3. Determine the Jacobian **J** according to equation 3.78.
4. Compute the update vector $\boldsymbol{\delta} = -\left(\mathbf{J}^\top\mathbf{J} + \lambda\mathbf{I}\right)^{-1}\mathbf{J}^\top\boldsymbol{\epsilon}_0$ as denoted in equation 3.79.
5. From equation 3.77, the estimation of the parameters $\mathbf{s}_0$ can be updated to the new estimation **s** according to $\mathbf{s} = \mathbf{s}_0 + \boldsymbol{\delta}$. From this new parameter vector the new estimations of the points **X**, lines **L** and cameras **P** can be obtained.
6. Computing again the reprojections of the points and lines, the new reprojection error $e$ can be obtained.
   If the new reprojection error $e$ increased referring to the initial error $e_0$, the update is not accepted. $\lambda$ is multiplied by the factor 10. Going back to step 4 the update vector is evaluated again, until the new error $e$ is smaller than $e_0$.
   If the new reprojection error $e$ is smaller than the initial error $e_0$, the update will be accepted. Thus, the parameter vector and the error vector are updated according to $\mathbf{s}_0 \leftarrow \mathbf{s}$ and $\boldsymbol{\epsilon}_0 \leftarrow \boldsymbol{\epsilon}$. The step size for the next iteration step is increased dividing $\lambda$ by the factor 10. Then on can go back to step 3 for a new iteration. This is done until the algorithm converges.

**Algorithm 3.1:** ***Bundle adjustment:*** *The bundle adjustment algorithm uses the Levenberg-Marquardt algorithm to improve the points, lines and cameras of the scene.*

of the update $\boldsymbol{\delta}$, that may also be a good criterion to quit the iterations, can be computed according to $\|\boldsymbol{\delta}\| \,/\, \|\mathbf{s}_0\|$. In the present algorithm, a combination of the different criteria is applied. The maximum number of iterations is limited, a threshold for the absolute error $e$ is used and the relative change of the update $\boldsymbol{\delta}$ is evaluated. It has shown that in most cases the relative change of the update leads to a reasonable termination of the algorithm. The other conditions are used as support criterion.

# 4 Evaluation

In this chapter the behaviour of the algorithm in different scenarios is evaluated. The outcomes are compared among themselves such that conclusions can be drawn. The Algorithm is evaluated using synthetic data as well as on real images.
The algorithm was implemented and tested in a Matlab environment, version 7.10.0.499 (R2010a) for 32 bit Windows systems. From points and lines, obtained from either real image data or generated data, a reconstruction is generated that is used as initialisation for the bundle adjustment procedure.
For testing with synthetic data a framework was implemented that generates the desired configuration of the scene. It provides the projections of the points and lines onto virtual image planes. The reconstruction process relies solely on these projections to estimate the scene.
The real images where made with the web cam that was already specified in chapter 1.3.4. Point and line correspondences where selected by hand. Points where first determined with sub-pixel precision using the Harris corner detector, borrowed from [33]. Afterwards, correspondences across the images where chosen by hand. Lines where selected by defining the endpoints in the image by hand. Subsequently, robust line fitting was performed. This way, line correspondences across the views where selected.

## 4.1 Evaluation with Synthetic Generated Data

Four different scenarios are examined. In the first configuration the scene does not change, but the noise on the points and lines in the image is changed in a controlled way. Thus, the influence of the image noise is validated. In the second scenario the influence of the baseline of the cameras is investigated. The third scenario provides a test on the influence of the number of the cameras involved in the scene. In the fourth test the algorithm is executed with different features. First only point features are considered, then only line features and in a last step both point and line features are used to check the influence of the feature type on the behaviour of the algorithm.
Evaluation with synthetic data has various advantages. Each desired configuration can be generated, the level of noise in the images can be controlled and the influence of single parameters can be surveyed by modifying only a single parameter while the rest of the scene remains unchanged. On the other hand, synthetic data is always only an approximation of the real world. The assumptions used, such as the camera model, may prove to be inaccurate or even wrong. Thus, the results of the evaluation with synthetic data should be interpreted with caution.
For the evaluation with synthetic data a camera model is used. The intrinsic parameters of the camera are an approximation of the parameters of the real camera used in the evaluation

with real data. The camera parameters are specified in chapter 1.3.4.

### 4.1.1 Influence of the Image Error

A scene consisting of 30 points, 30 lines and 6 cameras is generated. Since at least 7 point or 13 line correspondences are necessary in a minimal configuration, 30 lines and 30 points are a suitable number of correspondences. The points and lines are generated randomly in a specified area around the origin of the world frame. The cameras are arranged around the scene such that each of the point and line features is visible in each camera. In figure 4.1 the scene is visualised.

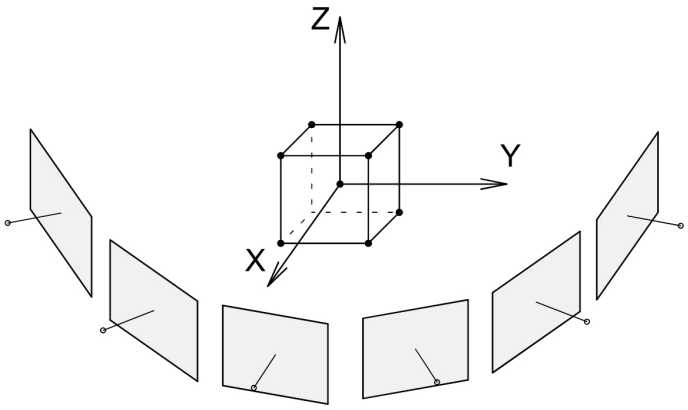



Figure 4.1: ***Schematic illustration of the scene used to examine the influence of the image error:*** *Six cameras are positioned around a scene. The scene consists of randomly generated entities, 30 points and 30 lines. These entities are positioned in an area around the origin of the world frame. The cameras are positioned such that each point and line is visible in each camera.*

Now the points and lines are projected into the camera images. Since the scene is generated synthetically, the data is exact which means o noise is present. Thus, a reconstruction of the whole scene from the image points and lines, called features, can be obtained perfectly, up to a common scale factor, as described in chapter 2. The bundle adjustment process, described in section 3, terminates after the first iteration because the data can not be improved.
The behaviour of the algorithm can be explored for different levels of noise by adding noise to the features of the image. For the specified configuration only the level of noise is adapted. The residual parameters such as the number and the position of the features and cameras in the scene remain unchanged. Starting from perfect data with a pixel error of 0 pixel, Gaussian noise will be added to the points and lines in the images. The noise is incremented in steps of $\sigma = 1$ up to a pixel error of $\sigma = 10$. This range covers a wide span of a possible image error. In real images a standard deviation of $\sigma = 10$ for the reprojection error (defined in chapter 3.4 for points and lines) would be a big distortion that should not crop up. For each examined error configuration 100 runs are executed. In each run new noise with the specified standard deviation $\sigma$ is generated. For each iteration of the bundle adjustment process, the minimisation error $e = \|\boldsymbol{\epsilon}\|$ on the normalised images

| $\sigma$ Image Noise | $\sigma_a$ error before BA | $\sigma_b$ error after BA |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1,916 | 0,412 |
| 2 | 3,673 | 0,787 |
| 3 | 6,108 | 1,215 |
| 4 | 8,284 | 1,645 |
| 5 | 9,718 | 1,992 |
| 6 | 12,122 | 2,649 |
| 7 | 13,906 | 3,345 |
| 8 | 16,673 | 3,580 |
| 9 | 17,831 | 4,515 |
| 10 | 20,127 | 4,722 |

Table 4.1: ***Standard deviation of the error for different noise levels:*** *In the left column the standard deviation $\sigma$ of the synthetic generated error in the images before the reconstruction process is shown. The second column contains the standard deviation $\sigma_a$ of the reprojection error after the reconstruction process. The bundle adjustment process is initialised with this data. The last column indicates the standard deviation $\sigma_b$ after the bundle adjustment process.*

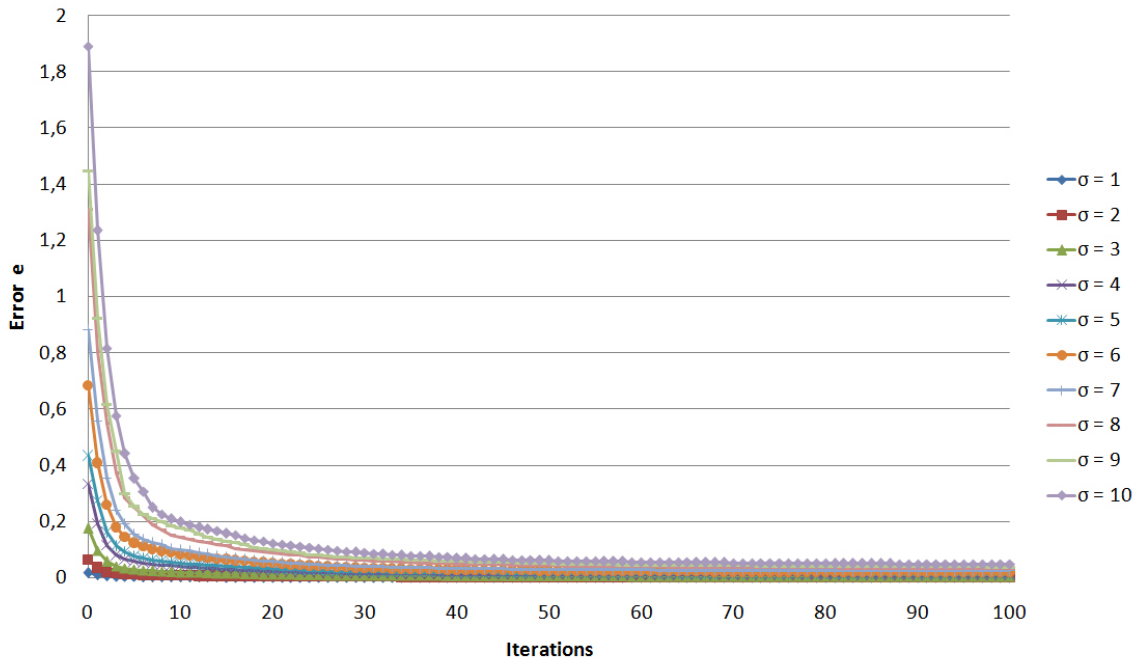



Figure 4.2: ***Different noise levels:*** *The chart contains 10 curves. Each curve stands for a noise level. The noise is specified with the standard deviation $\sigma$. The error $e$ for each noise level, averaged over 100 runs, is plotted against the iteration step of the bundle adjustment algorithm. It is determined as the sum of squares of the reprojection error in normalised images.*

is averaged over the total number of runs. Thus, the convergence behaviour of the bundle adjustment algorithm can be compared for different noise levels. The standard deviations of the reprojection error before and after the bundle adjustment are compared in table 4.1.
The standard deviation of the reprojection error before the bundle adjustment $\sigma_a$ is obviously different to the generated image noise $\sigma$, because each reconstruction process causes in an error of the reconstructed features. When the reconstructions are concatenated and the reprojection error of the concatenated system is computed, points and lines will be projected in all images. This also includes images that where not involved in the reconstruction of this specific feature. In the bundle adjustment process the total reprojection error will be reduced. It can be observed that the error is significantly reduced after the optimisation procedure when compared to the initial generated image noise.
The behaviour of the algorithm with changing noise conditions is visualised in the chart in figure 4.2.
It is obvious that the convergence of the algorithm depends on the magnitude of the noise in the images. In the presence of large image noise, the error converges very fast in the first iterations steps. As expected, the curve corresponding to the largest noise level of $\sigma = 10$ pixel provides the largest image error over the total minimisation process. Although it declines very quickly in the first steps compared to the other curves, the minimum to which it converges is larger than the minimum of the other curves. On the other hand, the curve corresponding to the smallest image noise level of $\sigma = 1$ is flat even in the first iterations steps, but converges to the smallest value. The curves in between for the image noise $\sigma = 2$ to $\sigma = 9$ behave in a similar way. The bigger the initial noise $\sigma$, the higher the initial error and the steeper the descent of the curve in the first steps of the minimisation. Furthermore, the smaller the image noise, the smaller the bottom limit to which the minimisation algorithm converges.

### 4.1.2 Influence of the Baseline

A scene consisting of 30 random points, 30 random lines and 6 cameras is generated. Since at least 7 point or 13 line correspondences are necessary in a minimal configuration, 30 lines and 30 points are a suitable number of correspondences. The points and lines lie in a specified area around the origin of the world frame. The six cameras are positioned on a circle around the $Z$-axis of the world frame, the circle lies in the $X$-$Y$ plane of the world frame. Their principal axis points towards the origin. The angle $\alpha$ on the circle between each two contiguous cameras is equal for each neighbouring camera pair. The noise of points and lines in the images is specified with $\sigma = 4$ pixels. For real images this noise level would be very high but still realistic. However it is important to note that the algorithm performs well even in the presence of poor image data. In figure 4.3 the configuration of the scene is illustrated schematically.

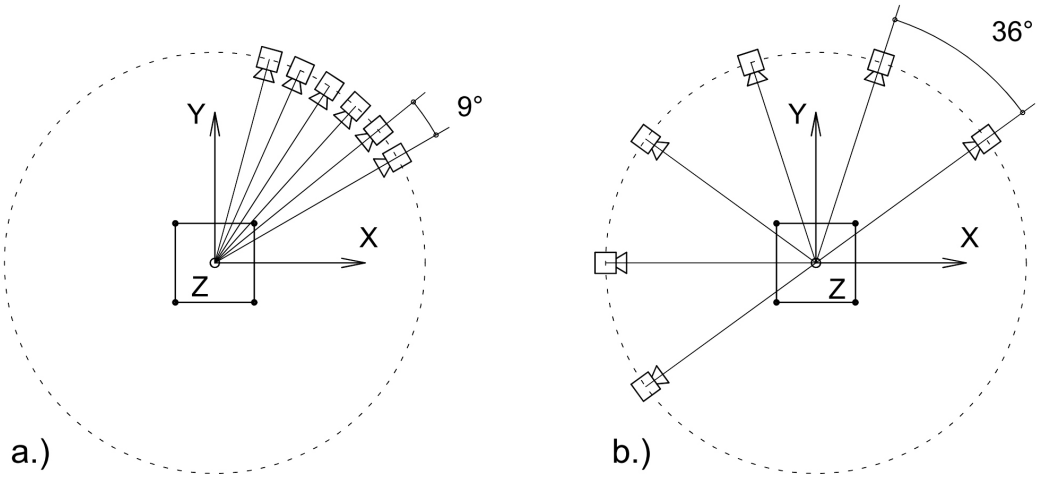



Figure 4.3: ***Schematic illustration of the scene for analysing the influence of the baseline between cameras:*** *The scene consists of 30 points and 30 lines, both generated randomly in an area around the origin of the world frame. Six cameras are positioned on a circle around the scene. The principal axis points towards the origin of the world frame. The angle $\alpha$ between two adjacent cameras is equal for each neighbouring camera pair. Starting from an angle $\alpha = 9°$ leads to a total angle of $45°$ between the first and the last camera as illustrated in (a). In each step the angle $\alpha$ is increased by $4.5°$. In the last step, the angle $\alpha$ between two adjacent cameras is $36°$. Thus, the total angle is $180°$. This case is illustrated in (b).*

In the first configuration the angle $\alpha$ between each two contiguous cameras, measured around the $Z$-axis of the world frame, is specified with $9°$. Thus, the $6$ cameras lie on the circle within $45°$. The algorithm is executed $100$ times for this configuration, in each run new Gaussian noise with the same standard deviation $\sigma = 4$ is generated on the point and line features. Next, the angle $\alpha$ is decreased by $4.5°$ such that $\alpha = 13.5°$. Again $100$ runs are executed. As in the first step new Gaussian noise with $\sigma = 2$ is generated. This procedure is repeated five times, increasing the angle $\alpha$ between two neighbouring cameras by $4.5°$ each time. In the last step the angle between two neighbouring cameras is $\alpha = 36°$. The

| $\gamma$ total angle | $\alpha$ angle | $\sigma_a$ error before BA | $\sigma_b$ error after BA |
|---|---|---|---|
| 45° | 9° | 4,925 | 1,820 |
| 67.5° | 13.5° | 5,121 | 1,904 |
| 90° | 18° | 6,295 | 2,190 |
| 112.5° | 22.5° | 6,221 | 2,106 |
| 135° | 27° | 9,510 | 3,399 |
| 157.5° | 31.5° | 6,029 | 3,008 |
| 180° | 36° | 6,204 | 3,383 |

Table 4.2: ***Standard deviation of the error for different baselines:*** *In the left column the total angle, meaning the angle between the first and the last camera, is shown. The second column contains the incremental angle that is the angle between two neighbouring cameras. In the last two columns the standard deviation of the reprojection error before the bundle adjustment $\sigma_a$ as well as the standard deviation of the error after the bundle adjustment procedure $\sigma_b$ is presented.*

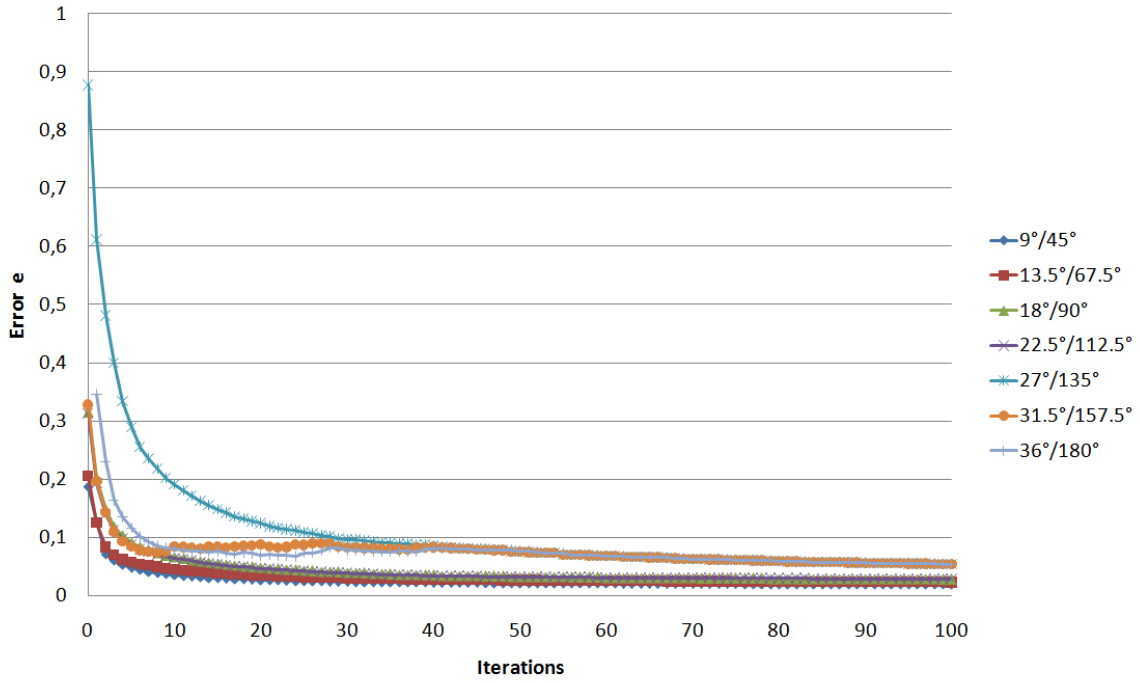



Figure 4.4: ***Varying baseline:*** *The chart contains 7 curves. Each curve stands for a different baseline between the cameras. The incremental angle between the cameras as well as the total angle of the cameras for each configuration is stated in the key on the right side of the chart. The noise in the images is assumed to be Gaussian and is equal for each image. It is specified with a standard deviation of $\sigma = 4$ pixel. The error $e$ for each configuration, averaged over 100 runs, is plotted against the iteration step of the bundle adjustment. It is determined as the sum of squares of the reprojection error in normalised images.*

total angle between the first and last camera is $180°$.
For each iteration of a run the minimisation error $e = \|\epsilon\|$ on normalised images is stored. Thus, the behaviour of the algorithm for different baselines is compared. The standard deviation of the reprojection error before and after the bundle adjustment are compared in table 4.2.
The algorithm behaves in a different manner than expected. The expectation was that with increasing angle and thus increasing baseline between two neighbouring cameras the initial error would decrease. In other words, the case with the smallest neighbouring angle of $\alpha = 9°$ was supposed to deliver the poorest results for the reprojection error of the reconstruction before the bundle adjustment. This is because the quality of the reconstruction depends on the baseline as explained in chapter 2.1.3 and illustrated in figure 2.6. The scenario where the neighbouring angle is maximal with $\alpha = 36°$ was expected to deliver the best results of the reconstruction because the baseline is largest.
However the algorithm behaves in a different way. The best results are achieved for the smallest baselines with an angle of $\alpha = 9°$ and $\alpha = 13.5°$ between two neighbouring cameras. The configuration with $\alpha = 18°$ and $\alpha = 22.5°$ performs slightly worse. In the case with $\alpha = 27°$ the reprojection error of the reconstruction, that is used as initialisation for the bundle adjustment process, is extremely high. The bundle adjustment procedure however performs well. In the last two configurations for $\alpha = 31.5°$ and $\alpha = 36°$ the reprojection error for the reconstructions before the bundle adjustment becomes better again. The minimisation algorithm however performs in a slightly bizarre way. After the error decreases in the first steps, it increases marginally. The curve for the configuration of $\alpha = 36°$ however seems to be the upper limit of the error $e$. Once the error level of this curve is reached, the error decreases again with exactly the same values for each iteration step.
Although the application was implemented carefully, it is not altogether impossible that an implementation error might be the reason for the behaviour of the algorithm. Another explanation for this behaviour might be the special configuration of the scene.

### 4.1.3 Influence of the Number of Cameras

A scene is generated synthetically that consists of 30 points and 30 cameras. Since at least 7 point or 13 line correspondences are necessary in a minimal configuration, 30 lines and 30 points are a suitable number of correspondences. The points and lines are generated randomly around the origin of the base frame. Furthermore cameras are positioned on a circle around the origin of the world frame. The principal axis of the cameras points toward the origin. The cameras are positioned such that the angle between the first and the last camera is $90°$. In figure 4.5 the scenario is illustrated schematically.

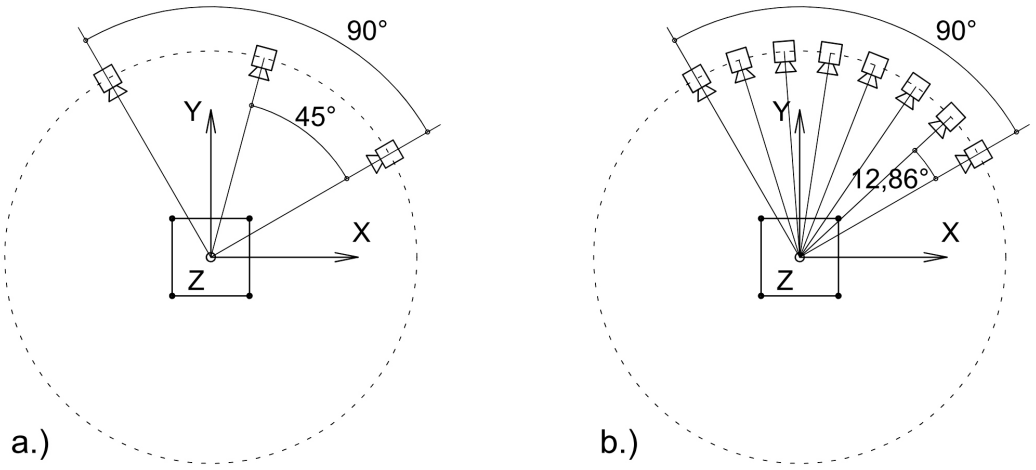



Figure 4.5: ***Schematic illustration of the scene to analyse the influence of the number of cameras on the minimisation process:*** *The scene consists of 30 points and 30 lines randomly generated in an area around the origin of the world frame. In the first step three cameras are positioned on a circle around the scene, the principal axis points towards the origin of the world frame as illustrated in (a). The angle between the first and last camera is $90°$. In each step an additional camera is added to the scene. The angle between the first and last camera remains unchanged and all cameras are arranged equidistantly. Thus, in number of cameras is increased to 8 in the last step. This is illustrated in (b).*

In a first step, three cameras are positioned in the scene. The angle between the first and second camera and between the second and third camera is $45°$. Camera images are created by projecting the points and lines onto the image planes of the cameras. In 100 runs, where each time a new generated noise level with $\sigma = 2$ is added to the images the convergence, the behaviour of the algorithm is surveyed. Afterwards, the same is done with four cameras, again with an angle of $90°$ between the first and the last camera. This causes in an angle of $30°$ between two neighbouring cameras. Repeating this two more times, the number of cameras is increased to six within an angle of $90°$. Thus, in the third step five cameras are positioned, the angle between two neighbouring cameras is $22.5°$. In the last step the angle between two collateral cameras is $18°$. The goal is to examine the influence of the number of cameras to the convergence behaviour in the bundle adjustment process. Thus, the image noise is equal for all images and the largest baseline in the bundle adjustment remains unchanged over the different scenarios. The standard deviation of the reprojec-

| number of cameras | $\alpha$ angle between neighbouring cameras | $\sigma_a$ error before BA | $\sigma_b$ error after BA |
|---|---|---|---|
| 3 | 45° | 1,270 | 0,574 |
| 4 | 30° | 1,833 | 0,674 |
| 5 | 22.5° | 2,230 | 0,757 |
| 6 | 18° | 2,672 | 0,869 |
| 7 | 15° | 2,968 | 0,925 |
| 8 | 12.86° | 3,250 | 0,972 |

Table 4.3: ***Standard deviation of the error for different numbers of cameras:*** *In the left column the number of cameras is shown. Since the total angle between the first and last camera is specified with 90° for each configuration, the angle between two neighbouring cameras decreases with increasing number of cameras. This angle is shown in the second column. The standard deviation of the reprojection error $\sigma_a$ before the bundle adjustment procedure and the standard deviation of the reprojection error $\sigma_b$ after the minimisation are outlined in the last two columns.*

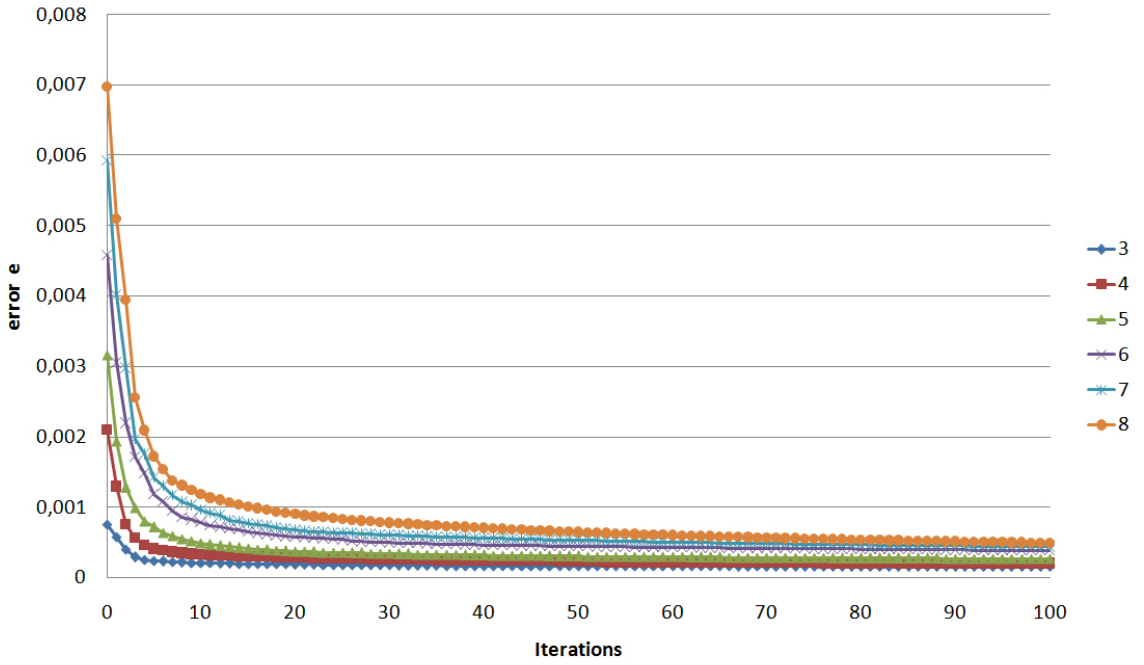



Figure 4.6: ***Different numbers of cameras:*** *The chart contains 6 curves. Each curve describes the convergence behaviour of the bundle adjustment algorithm with a specified number of cameras, as stated in the key on the right sight of the chart. The noise in the images is equal for each configuration and specified with the standard deviation $\sigma = 2$ pixel. The error $e$ for each curve is computed according to the sum of squares of the reprojection error for points and lines in each image divided by the number of cameras involved in the reconstruction and improvement procedure. The values of the error are averaged over 100 runs and plotted against the iteration steps of the bundle adjustment algorithm.*

tion error before and after the bundle adjustment for the different numbers of cameras are compared in table 4.3.
The Standard deviation $\sigma_a$ of the reprojection error in the images before the bundle adjustment procedure increases with increasing number of cameras involved in the reconstruction process. Each reconstruction has a separate error. This may be influenced by the noise in the image, the number of features and the baseline of the cameras involved in the reconstruction. The standard deviation of the reprojection error $\sigma_a$ before the minimisation in the first two steps, where 3 cameras are involved, lies below the initial image noise of $\sigma = 2$ pixel. When the trifocal tensor is determined, a minimisation over all image features is performed to find the best trifocal tensor for the image triplet. Thus, the result of the reprojected error of the reconstructed points and lines might be an improvement to the initial image noise. The bundle adjustment procedure then computes an improved result of the entities. Concatenating single reconstructions from each three images, the error increases. When reconstructions are concatenated and the reprojection error is computed, then points and lines will be projected to images that where computed in a different reconstruction procedure. This leads to an increased reprojection error. Thus, the reprojection error increases with each new camera involved in the process. In the chart of figure 4.6 the convergence behaviour of the bundle adjustment process on different numbers of cameras is illustrated. The curve corresponding to the set consisting of 3 cameras starts with the smallest error and converges to the smallest image error. The curve related to the set of 8 cameras starts with the largest error and converges to an error with the largest value compared to the sets consisting of fewer cameras. Large errors decrease very quickly in the first iteration steps. The shape of the curves is very similar. Besides the worse initialisation of the minimisation algorithm in the case of multiple cameras, no influence of the number of cameras to the quality of the minimisation procedure is recognisable.

### 4.1.4 Influence of the Feature Type

Three different scenarios are compared. In in the first scenario, the bundle adjustment is performed only for point features, in the second scenario only line features are present. In the last case both points and lines are present in the bundle adjustment process. In each case the scene consists of $6$ cameras, positioned equidistantly on a circle around the centre of the world frame. The circle lies in the $X$-$Y$ plane of the base frame. The circle has a radius of 2 units. Let one unit be defined as one meter. Thus, the centres of the cameras have a distance of $2m$ to the origin of the base frame. The total angle between the first and the last camera is $90°$, measured around the $Z$ axis of the base frame. The three scenarios are illustrated in figure 4.7.

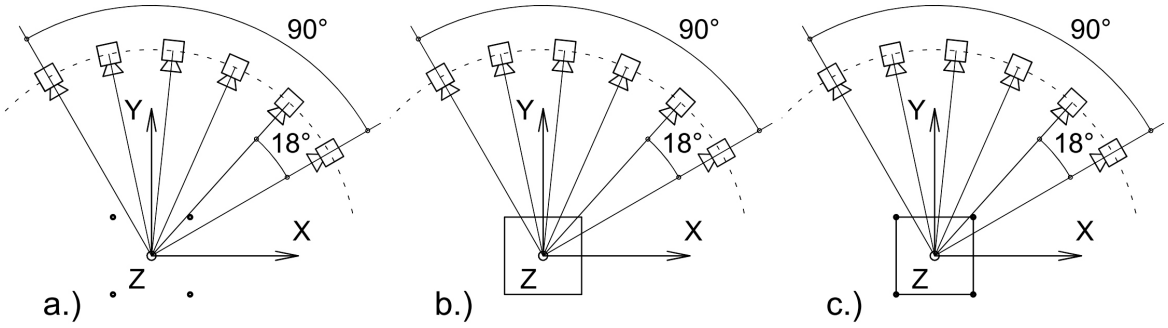



Figure 4.7: ***Schematic illustration of the three different scenes that are analysed and compared:*** *For all three cases the reconstruction is made from point and line features. The bundle adjustment process is executed only for point features, as illustrated in (a), only for line features as illustrated in (b) and for both point and line features as illustrated in (c). The three results are compared to investigate the influence of the kind of the features on the minimisation process.*

The reconstruction that is used as initialisation for the minimisation process is computed from points and lines in each of the three cases. This is because the number of features has an influence on the quality of the reconstruction. In this test only the the behaviour of the bundle adjustment is examined. Thus, the initialisation must be the same in all three cases. In each scenario $500$ runs are performed. The quality of the reconstruction is measured in the 3D data. The camera poses of the reconstructions before and after the bundle adjustment procedure are compared with the poses of the synthetically generated cameras that where used to generate the synthetic images. Since the reconstructions are obtained without any information about their location in the world frame and up to a common scale factor, a transformation must be applied to the reconstruction to make them comparable to the original data. The frame of the original data is considered as base frame. The reconstructions of the scene are transformed such that the pose of the first camera of the system coincides with the pose of the camera in the original system. Furthermore a scale factor is computed as the average distance between the camera centre of the first cameras to the camera centres of the other cameras involved to the scene. Thus, the reconstructions are scaled such that their scale factor is equal to the scale factor of the original system. This alignment is illustrated in figure 4.8.

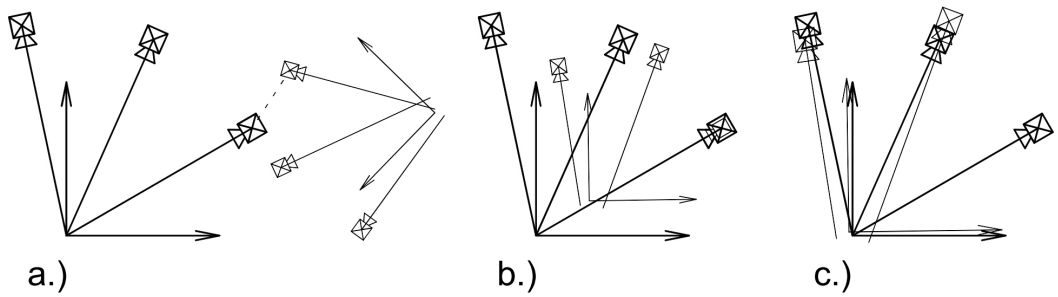



Figure 4.8: ***Alignment of a reconstructed scene to the original scene:*** *Schematic illustration of the alignment of two scenes. The cameras are illustrated with their principal axes. The synthetically generated reference scene is presented with thick lines, the reconstructed frame, containing errors, is presented with thin lines. The principal axes of the reference frames intersect in the origin of the world frame. The principal axes of the reconstructed frames however do not necessarily intersect in the same point because of a certain translational and rotational error of the cameras. In (a) the two corresponding systems are incoherently lying in the space. The dashed line between the first cameras suggests the link between the two systems. In (b) the reconstructed system is aligned to the reference system via the pose of the first camera. The reconstructed system will be scaled to coincide with the reference system. This is illustrated in (c). Thus, the translational and the rotational error can be determined.*

The pose of the first camera in the reconstructed scene corresponds perfectly with the pose of the first camera in the original scene. The other corresponding cameras may differ in their pose. This error can be determined as a translational and a rotational error. The translational error is simply the euclidean distance between the camera centres. The rotational error can be computed as a $3 \times 3$ rotation matrix that describes the rotational difference in 3-space. This rotation matrix can be expressed as a rotation axis and a corresponding angle. This angle is the rotational error between the two cameras.
Computing these errors for each camera in each run of each of the three configurations leads to table 4.4, where the translational error is displayed and table 4.5, where the rotational error is displayed. In both tables the error for each camera is averaged over the $500$ runs.

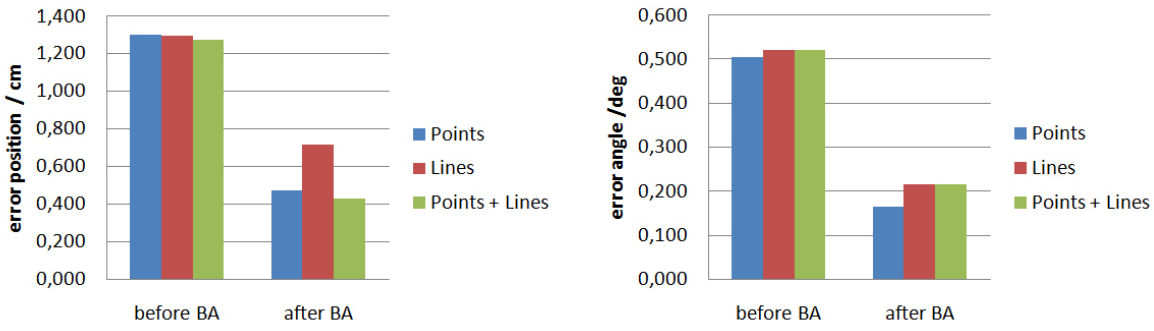
The entire scene consists of 60 randomly generated points, 60 randomly generated lines and the six cameras. Compared to the other evaluations on synthetic data the number of points and lines was increased, because the minimisation algorithm is applied on configurations consisting only of points and lines. The points and lines are reprojected into the images of the cameras and Gaussian noise with a $\sigma = 3$ pixel is added. From those two dimensional correspondences the three dimensional points and lines as well as the cameras are reconstructed. After the reconstruction process the bundle adjustment procedure is performed based on the results of the reconstruction. In the first step, only the $60$ points are considered as features. The sum of squares of the reprojection error of the points is the error function that has to be minimised. The rotational and translational error between the cameras is computed before and after the bundle adjustment procedure. In the second step,

| | points | | lines | | points + lines | |
|---|---|---|---|---|---|---|
| | translation error before BA /m | translation error after BA /m | translation error before BA /m | translation error after BA /m | translation error before BA /m | translation error after BA /m |
| camera 2 | 0.00759 | 0.00438 | 0.00756 | 0.00585 | 0.00753 | 0.00273 |
| camera 3 | 0.01300 | 0.00424 | 0.01271 | 0.00684 | 0.01248 | 0.00282 |
| camera 4 | 0.01122 | 0.00443 | 0.01134 | 0.00645 | 0.01124 | 0.00265 |
| camera 5 | 0.01478 | 0.00501 | 0.01448 | 0.00728 | 0.01454 | 0.00507 |
| camera 6 | 0.01860 | 0.00574 | 0.01872 | 0.00963 | 0.01812 | 0.00835 |
| mean /m | 0.01304 | 0.00476 | 0.01296 | 0.00721 | 0.01278 | 0.00432 |
| mean /cm | 1.304 | 0.476 | 1.296 | 0.721 | 1.278 | 0.432 |

Table 4.4: ***Translational error:*** *Deviation between the synthetically generated cameras and the reconstructed cameras before and after the bundle adjustment, averaged over 500 runs. It is compared for the case considering only point features, line features and for both point and line features.*

| | points | | lines | | points + lines | |
|---|---|---|---|---|---|---|
| | rotation error before BA /rad | rotation error after BA /rad | rotation error before BA /rad | rotation error after BA /rad | rotation error before BA /rad | rotation error after BA /rad |
| camera 2 | 0.00581 | 0.00240 | 0.00581 | 0.00175 | 0.00592 | 0.00170 |
| camera 3 | 0.00918 | 0.00261 | 0.00955 | 0.00275 | 0.00954 | 0.00277 |
| camera 4 | 0.00946 | 0.00294 | 0.00991 | 0.00374 | 0.00984 | 0.00374 |
| camera 5 | 0.00965 | 0.00322 | 0.01007 | 0.00489 | 0.01001 | 0.00490 |
| camera 6 | 0.00991 | 0.00330 | 0.01008 | 0.00581 | 0.01021 | 0.00583 |
| mean /rad | 0,00880 | 0,00289 | 0,00909 | 0,00379 | 0,00910 | 0,00379 |
| mean /deg | 0,504 | 0,166 | 0,521 | 0,217 | 0,522 | 0,217 |

Table 4.5: ***Rotational error:*** *Deviation between the synthetically generated cameras and the reconstructed cameras before and after the bundle adjustment, averaged over 500 runs. It is compared for the case considering only point features, line features and for both point and line features.*




(a) *Translational error* (b) *Rotational error*

Figure 4.9: ***Comparison of Bundle adjustment on points, lines and both points and lines:*** *In (a) the translational error is compared, in (b) the rotational error is compared.*

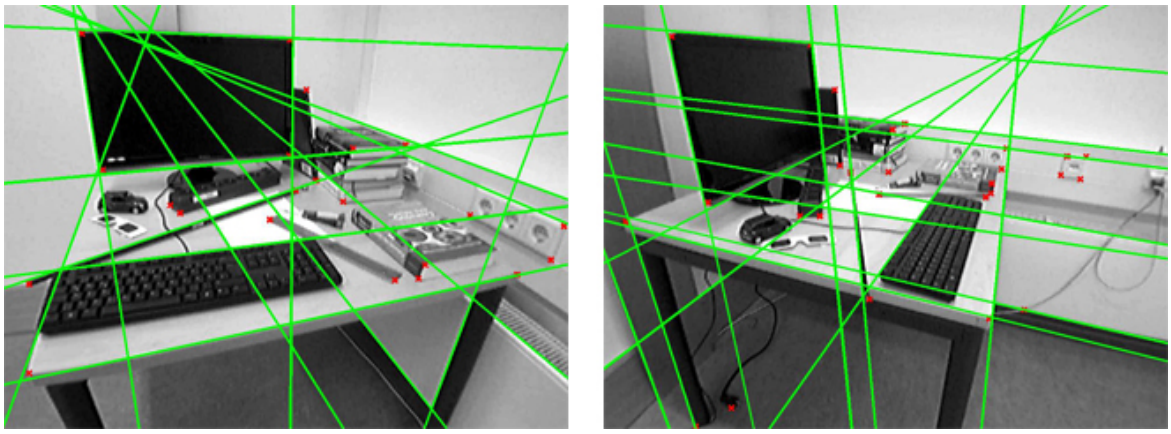
only the 60 lines are considered. Again, the error between the cameras is computed before and after the optimisation. In the last step both points and lines are considered and the error between the cameras before and after the bundle adjustment is computed. In table 4.4 and 4.5 as well as in figure 4.9 the error before and after the minimisation procedure for the different scenarios is compared.
Since in each run over each scenario a new error in the images was generated, the translational and rotational errors before the bundle adjustment procedure are similar, but not absolutely identical. Comparing the error for bundle adjustment on points and lines only, the algorithm performs better on points, as well for the translational as for the rotational error. The translational error for the bundle adjustment on points and lines leads to the best results and lies in the region of the error for the bundle adjustment only on points. The rotational error for the bundle adjustment on both points and lines however lies in the region of the error only for lines. For this behaviour no obvious reason could be identified. Thus, this behaviour may be subject to further research.

## 4.2 Evaluation with Real Image Data

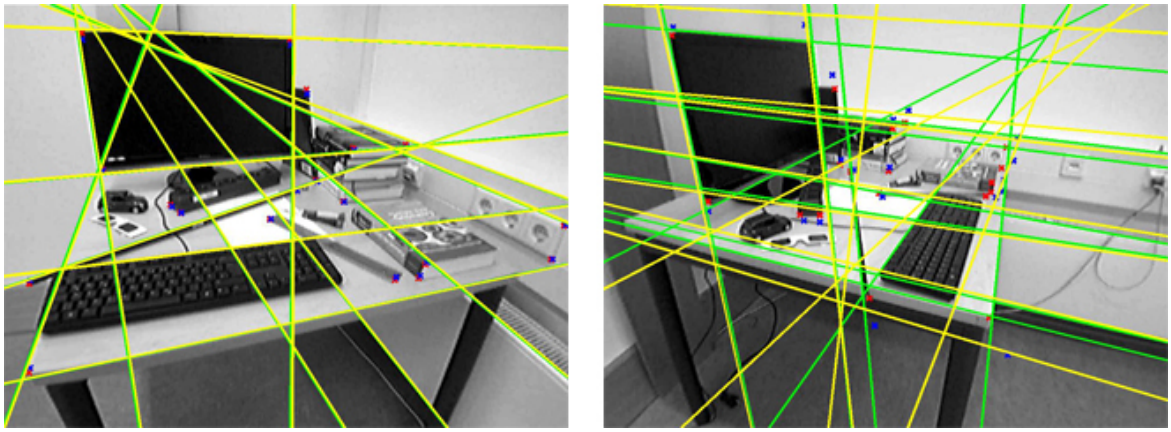
In addition to the evaluation with synthetic data the algorithm is also applied on real images. The scene is reconstructed from feature correspondences across the images. With the camera specified in chapter 1.3.2 a stream of a scene was recorded. By hand seven views that display the scene from different points of view where selected. In figure 4.10 the images involved in the bundle adjustment process are shown.

Figure 4.10: ***Image sequence*** *The sequence consists of 7 images, obtained from an image stream of a scene. The grayscale images are recorded with a wide angle lens and have a resolution of* $640 \times 480$ *pixels.*

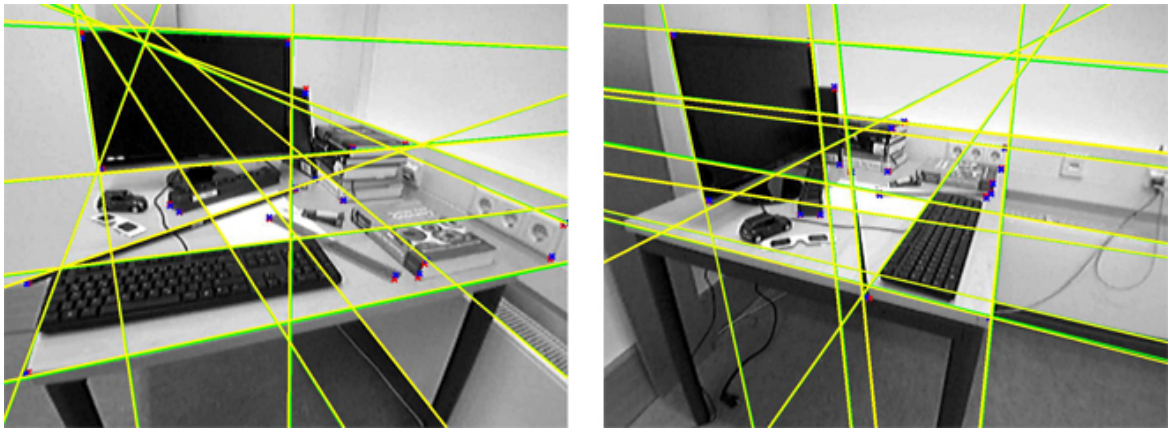
For this set of images, point and line correspondences are required. To find points in the images, an implementation of the Harris corner detector, borrowed from [33], was applied to each of the images to find significant sub-pixel precise corners in the image. Point correspondences across the images where selected by hand. Lines where obtained by defining two points on an edge in the image by hand. On the line connecting the specified points profiles perpendicular to the line where created. On these profiles the sub-pixel precise position of the edge was determined. Fitting a line trough the points defined by this means resulted in a sub-pixel precise line representing an image edge. In this manner line correspondences across the images where found. The selected point and line features are shown in figure 4.11(a). In total $34$ point correspondences and $17$ line correspondences are used in the reconstruction process. From these entities $19$ point correspondences and $13$ line correspondences are visible in all images. These features where refined in the bundle adjustment process.
As described in chapter 2 the reconstruction process is performed based on these features. Thus, a scene is obtained consisting of the reconstructed points, lines and seven cameras. Reprojecting these points and lines into the image planes of the cameras yields a reprojection error regarding the measured features. This is illustrated in figure 4.11(b).
The bundle adjustment process is applied to this scene in order to minimise the reprojection error over the complete scene. The result after the minimisation process is illustrated in figure 4.11(c).
The reprojection error after the reconstruction process has a standard deviation of $\sigma_a =$



(a) *Measured point and line features in the first and last image of the sequence. The measured point features are marked as red crosses, the measured line features are marked as green lines.*



(b) *Measured and reprojected features in the first and last image of the sequence after the reconstruction process, before performing bundle adjustment.*



(c) *Measured and reprojected features in the first and last image of the sequence after performing bundle adjustment. Comparing the images with (b) a decreasing deviation can be observed.*

Figure 4.11: ***Reprojection error images:*** *The green lines and red points represent the measured features in the image, the yellow lines and the blue points represent the reprojected features.*

18.818 pixel. The standard deviation of the reprojection error after the minimisation process is reduced to $\sigma_b = 2.983$ pixel. The reasons for such a big error may lie in the calibration of the camera. As already noted a web-cam with a fisheye lens was used to record the image stream. The calibration of the camera with wide angle lens is much more inaccurate than the calibration of a traditional camera. Especially in the image border the distortion could not be removed entirely. This can be seen in the images in figure 4.10 and in figure 4.11. A greater number of correspondences may also reduce the initialisation error. The convergence behaviour of the bundle adjustment is illustrated in the chart in figure 4.12.

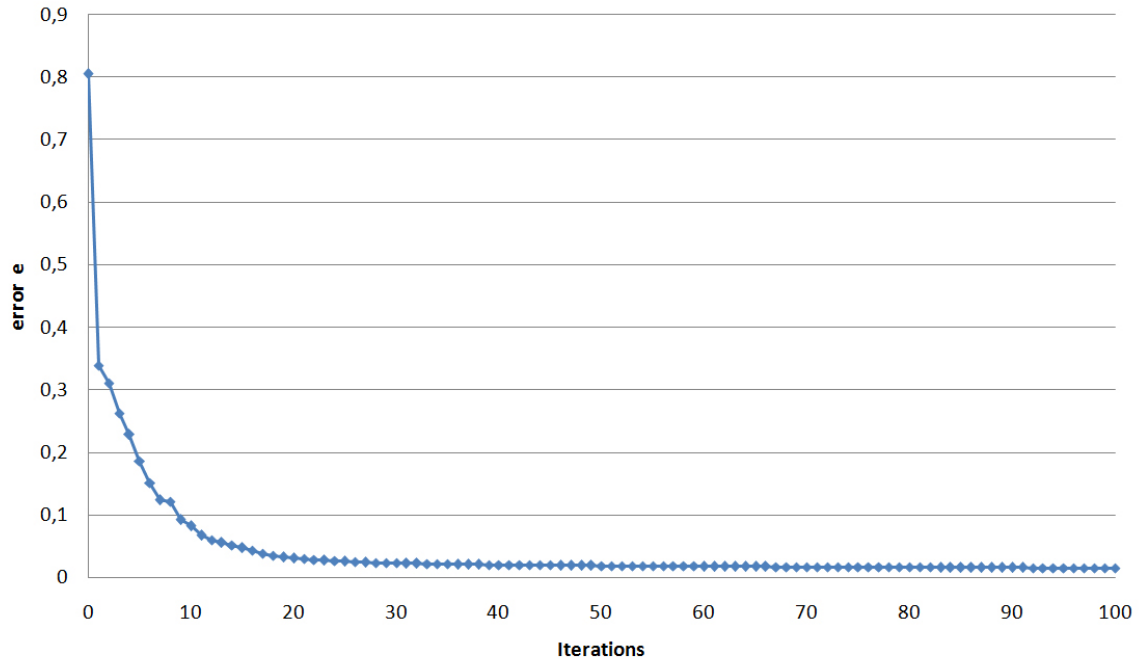



Figure 4.12: ***Image Sequence:*** *The error $e$ is plotted against the number of iterations. The curve shows the convergence behaviour of the bundle adjustment procedure applied on real image data.*

The error $e$ is determined in each step of the bundle adjustment process as the sum of squares of the reprojection errors of points and lines over all images. One can compare these results with the results in 4.1.1, where the behaviour of the minimisation procedure for different image noise is compared. The curve lies in between the curves corresponding to $\sigma = 6$ and $\sigma = 7$ in the chart in figure 4.2. Comparing the results of the standard deviations $\sigma_a = 18.818$ and $\sigma_b = 2.983$ with the results of the first test outlined in table 4.1, one can observe that $\sigma_b$, corresponding to the standard deviation before the bundle adjustment, is worse compared to the $\sigma_b$ corresponding to $\sigma = 6$ and $\sigma = 7$ in table 4.1. The standard deviation $\sigma_a$, corresponding to the error after the bundle adjustment, however lies again between the values of $\sigma_a$ in table 4.1 corresponding to $\sigma = 6$ and $\sigma = 7$.

# 5 Conclusion

In this last chapter a perspective is given how the application presented in this thesis can be developed further and which tasks may be improved. Furthermore this chapter includes a résumé that concludes the thesis.

## 5.1 Future Work

While working on the thesis and especially while evaluating the results of chapter 4 it has shown that the concept can be improved and further developed in several ways.
Thus, it might be an interesting task to augment the set consisting of features from points and lines considering other types of geometric primitives. Conics, that are described briefly in [25], chapter 2.8, have similar geometric properties than lines. It might be interesting to examine if conics can provide further informations in a tracking process and thus if they can be incorporated in the tracking and bundle adjustment procedure in a similar way as points and lines. Another interesting task might be to consider square angles as relevant features in the scene. In human made environments also square angles are abundant. Furthermore the consideration of CAD models that might be recognised in the scene may be a task for further research.
But even the representation of lines in 3-space might be improved. As already stated, lines have 4 degrees of freedom in the three-dimensional space. A line represented in Plücker coordinates (described in chapter 1.3.8) has 6 parameters. This representation was used for the reconstruction process. For the bundle adjustment procedure however the orthonormal representation (described in chapter 3.3.1) was selected. This representation allows the representation a line in 3D by the minimal set of 4 parameters. Although this representation is applicable in the bundle adjustment procedure, it cannot be used in the reconstruction of lines, because it has no geometrical meaning such that projective transformations or projections can be applied to it. On the other hand, the representation in Plücker coordinates is not appropriate in the bundle adjustment procedures because a minimal representation in the number of parameters is required. This is because correcting the parameters after an iteration step the internal constraints of the Plücker line will not be satisfied. This might lead to an invalid parameter vector that does not correspond to a valid line in the three-dimensional space.
For evaluating on real data points and lines where selected by hand. It is obvious that a solution is required that find points and lines automatically in the images and finds correspondences across images. While point matching algorithms, such as the FAST corner detector used in PTAM (proposed by Klein in [32]) are well known to perform fast and accurate, an appropriate line matching algorithm is required that finds line correspondences across the images.
The evaluation in chapter 4 showed that the behaviour of the algorithm can not be ex-

plained in each case. The second test case on synthetic data, where the influence of the baseline is examined, leads to an unexpected result (see chapter 4.1.2). To understand why this results occurred, the reconstruction and minimisation procedure might be observed more in detail, evaluating not only the end results, but also interim results of the different steps of the algorithm.
The fourth test scenario (see chapter 4.1.4) where the algorithm was tested and compared for points only, lines only and points and lines together, returned also results that might be subject to further research. It is obvious to see that the feature type has an influence on the performance of the bundle adjustment algorithm. For specifying exactly which configuration leads to the best results, an automatic line matching for real images must be provided. Then the results can be compared in an objective way.
This thesis provides a concept for the three-dimensional reconstruction of points and lines from image features. In a next step this approach might be implemented in a real-time tracking environment. One possibility for doing this might be the augmentation of PTAM [32]. Actually PTAM works on point features only. The concept developed throughout this thesis might be used considering point and line features in the images to incorporate these informations in the reconstruction and the bundle adjustment procedure of PTAM or a comparable tracking environment. Not until then a clear statement can be made if the use of line features in the image might lead to an improvement of the tracking and minimisation procedure and what kind of improvement can be achieved.

## 5.2 Résumé

In this thesis a method was developed to reconstruct a scene, consisting of points, lines and camera poses from images and perform an optimisation procedure over the complete scene minimising the reprojection error of points and lines in the images. The novelty to other approaches is to consider not only points, but also lines as features in the images. This new approach might be interesting for tracking applications in human environments such as in buildings, since various straight edges are present there. These straight edges can be treated as lines.
The concept described in this thesis might be incorporated in a tracking and mapping algorithm that uses points only as image features. Considering points and lines simultaneously might increase the information content in the images. Thus, informative results may be archived, especially in environments consisting mainly of straight lines.

# Bibliography


[1] K. B. Atkinson, editor. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, Caithness, Scotland, 1996.

[2] A Azarbayejani and A P Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995.

[3] Caroline Baillard, Cordelia Schmid, Andrew Zisserman, and Andrew Fitzgibbon. Automatic line matching and 3D reconstruction of buildings from multiple views.

[4] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, December 2005.

[5] James R Bergen, P Anandan, J Hanna, and Rajesh Hingorani. *Hierarchical Model-Based Motion Estimation*, volume 588, pages 237–252. Springer, 1992.

[6] Å Björck. *Numerical methods for least squares problems*, volume 31. SIAM, 1996.

[7] M Black. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[8] A Blake and A Zisserman. *Visual Reconstruction*, volume 6. MIT Press, 1987.

[9] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):1–21, 2005.

[10] P Burt and E Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[11] J Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[12] R. O. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proc 12th IEEE Int Symp on Wearable Computers, Pittsburgh PA, Sept 28 - Oct 1, 2008*, pages 15–22, 2008.

[13] N. Chiba and Takeo Kanade. A tracker for broken and closely spaced lines. In *Proceedings of the 1996 International Society for Photogrammetry and Remote Sensing Conference (ISPRS '98)*, volume XXXII, pages 676 – 683, 1998.

[14] Gabriella Csurka and Radu Horaud. Finding the Collineation Between two Projective Reconstructions apport. (3468), 1998.

[15] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[16] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 2006.

[17] E Eade and T Drummond. Scalable monocular slam. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Volume 1 CVPR06*, 1(c):469–476, 2006.

[18] O Faugeras and B Mourrain. *On the geometry and algebra of the point and line correspondences between N images*, pages 951–956. Number October. IEEE Comput. Soc. Press, 1995.

[19] Olivier Faugeras and Martial Hébert. *The Representation, Recognition, and Positioning of 3-D Shapes from Range Data*, pages 301–354. KLUWER Publisher, Co., 1987.

[20] Olivier D Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? *European Conference on Computer Vision*, 588:563–578, 1992.

[21] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[22] R I Hartley. Projective reconstruction and invariants from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1036–1041, 1994.

[23] R I Hartley. *A linear method for reconstruction from lines and points*, pages 882–887. IEEE Computer Society Press, 1995.

[24] R I Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[25] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[26] R.I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[27] Richard I Hartley. Estimation of relative camera positions for uncalibrated cameras. *Compute*, 92(1):579–587, 1992.

[28] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146 – 157, 1997.

[29] T.S. Huang and O.D. Faugeras. Some properties of the E matrix in two-view motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(12):1310–1312, 1989.

[30] Manuel Huber, Daniel Pustka, Peter Keitler, Echtler. Florian, and Gudrun Klinker. A System Architecture for Ubiquitous Tracking Environments. In *Proceedings of the 6th International Symposium on Mixed and Augmented Reality (ISMAR)*, November 2007.

[31] T Kanade. A theory of origami world. *Artificial Intelligence*, 13(3):279–311, 1980.

[32] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, November 2007.

[33] P. D. Kovesi. MATLAB and Octave functions for computer vision and image processing. Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[34] J J Leonard and H F Durrant-Whyte. *Simultaneous map building and localization for an autonomous mobile robot*, volume 3, pages 1442–1447. Ieee, 1991.

[35] H C Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[36] HC Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *in Computer Vision: Issues, Problems,*, page 61, 1987.

[37] B D Lucas and T Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, 130(x):674–679, 1981.

[38] D Marr and T Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976.

[39] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, volume 8. W.H. Freeman, 1982.

[40] D. Matinec and T. Pajdla. Line reconstruction from many perspective images by factorization. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, pages I–497–I–502, 2003.

[41] J E W Mayhew and J P Frisby. Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, 17(1):349–385, 1981.

[42] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *International Joint Conference on Artificial Intelligence*, 18:1151–1156, 2003.

[43] H H Nagel and W Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, 1986.

[44] M Okutomi and T Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[45] J. Philip. A Non-Iterative Algorithm for Determining All Essential Matrices Corresponding to Five Point Pairs. *The Photogrammetric Record*, 15(88):589–599, 1996.

[46] T Poggio, V Torre, and C Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 1985.

[47] Daniel Pustka, Manuel Huber, Christian Waechter, Florian Echtler, Peter Keitler, Joseph Newman, Dieter Schmalstieg, and Gudrun Klinker. Automatic configuration of pervasive sensor networks for augmented reality. *IEEE Pervasive Computing*, 10(3):68–79, July-September 2011.

[48] L G Roberts. *Machine perception of three-dimensional solids*, volume ed, pages 159–197. MIT Press, 1965.

[49] A Rosenfeld, R A Hummel, and S W Zucker. Scene labeling by relaxation operations. *Ieee Transactions On Systems Man And Cybernetics*, 6(6):420–433, 1976.

[50] C. Schmid and A. Zisserman. Automatic line matching across views. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 666–671. IEEE, 1997.

[51] S M Seitz, B Curless, J Diebel, D Scharstein, and R Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Volume 1 CVPR06*, 1(c):519–528, 2001.

[52] A Shashua. *Trilinearity in Visual Recognition by Alignment*, volume 800-801, pages 479–484. Springer–Verlag, 1994.

[53] A Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):779–789, 1995.

[54] Gabe Sibley. Relative Bundle Adjustment. *Electronic Notes in Theoretical Computer Science*, 220(3):5–21, December 2008.

[55] GG Slabaugh. Computing Euler angles from a rotation matrix. *denoted as TRTA implementation from: http://www. starfireresearch. com/services/java3d/samplecode/FlorinE ulers. html*, 1999.

[56] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1987.

[57] Peter Sturm, Saint Ismier, and France Firstlastinriafr. Multiple-view structure and motion from line correspondences. *Proceedings Ninth IEEE International Conference on Computer Vision*, (i):207–212 vol.1, 2003.

[58] R Szeliski and S B Kang. Recovering 3d shape and motion from image streams using nonlinear least squares. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 5(CRL 93/3):752–753, 1993.

[59] C J Taylor, David Kriegman, and P Anandan. *Structure and Motion in Two Dimensions from Multiple Images: A Least Squares Approach*, pages 242–248. IEEE Computer Society Press, 1991.

[60] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[61] Javier Tordable. Bundle Adjustment Refinement of Scenes with Moving Cameras. pages 1–21, 2009.

[62] B Triggs. Factorization methods for projective structure and motion. *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages845(June):845–851, 1996.

[63] Bill Triggs, P McLauchlan, and R Hartley. Bundle Adjustment - A Modern Synthesis. *Vision algorithms: theory*, 34099:298–372, 2000.

[64] J Weng, N Ahuja, and T S Huang. *Closed-form solution and maximum likelihood : A robust approach to motion and structure estimation*, pages 381–386. 1988.

[65] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998.

[66] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.