Signals & Systems

Segmentation and Motion Estimation of Multiple Independently Moving Objects in Stereo Video Streams

M.Sc. Thesis

René Willemink

University of Twente
Department of Electrical Engineering,
Mathematics & Computer Science (EEMCS)
Signals & Systems Group (SAS)
P.O. Box 217
7500 AE Enschede
The Netherlands

Report Number:	SAS027-05
Report Date:	September 21, 2005
Period of Work:	08/11/2004 – 22/09/2005
Thesis Committee:	Prof. Dr. ir. C.H. Slump
	Dr. ir. F. van der Heijden
	Prof. Dr. ir. P.P.L. Regtien

Abstract

A method for segmentation and estimation of object motion and structure from 3-d points in a dynamic scene, observed by a sequence of stereo images is proposed. The scope of the assignment is in a project named "Wearable Navigation Assistance - A Tool for the Blind", with the goal of developing a local navigation system for blind people. The proposed method is mainly an extension to a previously described method by Qian [1], with several changes to the method with respect to the measurements, implemented state estimation and clustering.

The proposed method is a recursive method where the estimation problem is solved by state estimation of a dynamic system. State estimation will be handled in our method by a Rao-Blackwellized particle filter, where the object motion part of the state is represented by a set of weighted samples and conditioned on the motion, the object structure is represented by Kalman filters.

The segmentation problem, defined as the task of segmenting the scene into independently moving objects, is solved by keeping track of which motion samples are compatible with which part of the object structure. By clustering motion samples according to similar object structure compatibility, a segmentation will result where the set of motion samples is divided into subsets, each representing a distinct object motion.

The feasibility and performance of the proposed method are investigated by running the method on synthetic scenes. Results from these experiments indicate that the method works well for the simulated scenes. The computational complexity of the method is quite high and as a consequence the method can not yet be implemented in a real-time scenario. Also further research needs to be performed to see how the method performs in real scenes.

Acknowledgments

When I started thinking about a subject for my master's assignment, which is the final part of my study Electrical Engineering at the University of Twente, I had a vague idea of what I wanted to do. I knew I wanted to do something in Computer Vision. I got interested in this while doing an Individual Research Assignment on Background Subtraction somewhere during my studies. So I contacted Dr. Ir. F. van der Heijden, who is working on this kind of research at our University. He had an interesting project which I decided to accept. The results of my investigation on this project are presented in this thesis.

The work described in this thesis would not have been realized without the help of many others. First I would like to thank my daily supervisor Ferdi van der Heijden, for his feedback and guidance during the project. Also, I would like to thank the other people involved in this project, among them are my two other supervisors Prof. Dr. Ir. C.H. Slump and Prof. Dr. Ir. P.P.L. Regtien. Furthermore, I would like to thank all other people for their support, like my roommates at Campuslaan 21. Last, but certainly not least, I wish to thank my parents and family for the support of my studies.

Enschede, September 19, 2005

René Willemink

Contents

1

Introduction

1.1 Scope

This assignment is part of a project named "Wearable Navigation Assistance - A Tool for the Blind" [2]. The goal of the project is to develop a local navigation system for blind people, that assists a blind person to walk and navigate independently through an unknown environment. In order to execute these tasks, the system should be able to inform the blind person about the environment and to alert him or her to avoid collisions. The assessment of the environment will be handled by two types of measurements, namely acoustical and optical. The information gathered by both measurement sources will be fused to generate a map of the environment (see Figure 1.1).

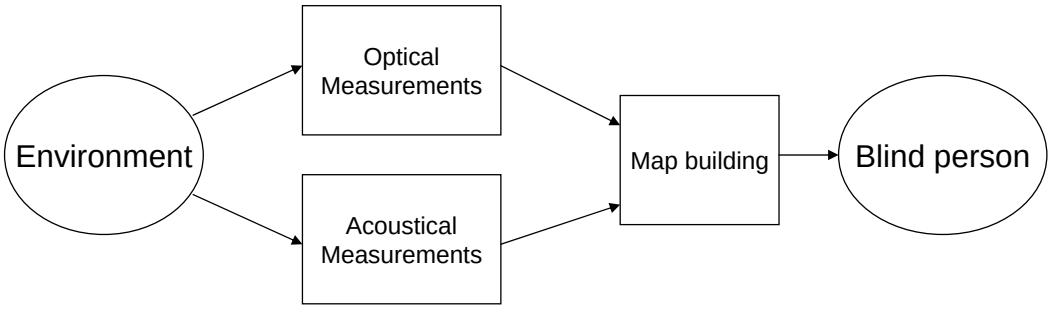


Figure 1.1: Electronic navigation system for blind people

The acoustical sensing system uses an active kind of measurement and will be based on time of flight (ToF) measurements. The sensing principle is based on measuring the time that elapses between the moment of emitting a sound wave and the moment of receiving the first echo of this sound wave. This measured ToF can be used to estimate the distance from the device to objects in the environment. The optical sensing system uses a passive kind of measurement. The environment will be observed by a video camera. By tracking the image positions of selected points in the scene through subsequent image frames, it is possible to reconstruct the motion of the camera and the 3-d positions of the selected points in the scene. The output of both the acoustical and optical system will be used to maintain and update a map of the environment by the map builder.

1.2 Problem definition

In this thesis we will investigate the problem of reconstructing object motion and object structure from a sequence of optical measurements, taken with a video camera. The assignment to be carried out is a continuation of the work of Jeroen Rozenberg in a previous M. Sc. project [3]. In the rest of this

thesis we will refer to this previous work as the "previous method". The previous method consisted of the sequential processing of measurements from a video stream, supported by inertial measurements of the rotational velocity of the video camera. These two measurement sources were combined in order to recursively estimate the relative motion of the camera through the scene and relative position of selected points in the observed scene. To visualize this, think for example of a person walking through a corridor. The camera is attached to the person, the relative motion is the motion difference between the camera and the corridor. When the relative motion is known, it can be used to transform the estimated corridor structure to coincide with the corridor as it is being observed by the person at the current time. The recursive estimation was implemented with an Extended Kalman Filter. The goal of the work described in this thesis is to make this process more robust by using a Particle Filter to handle the state estimation as opposed to the previously implemented Kalman Filter.

1.3 Outline of the report

In Chapter 2 previous work will be described and analyzed. First an introduction to the problem of recovering scene structure and motion through the scene will be given. This will be accompanied by literature references to relevant previous work. After that we will focus on recursive structure from motion estimation, since that will be the main goal of this work. The chapter will end with a more in depth view on two interesting methods. The first method is the one implemented in the previous M.Sc. assignment [3]. The second one can be seen as a basis of the proposed method in this thesis. In Chapter 3 the proposed method will be presented. We will start with a resume of important criteria that were used when designing this new method. After that a short overview of the method will be given. In the rest of the chapter the method will be further explained. In Chapter 4, the performance of the proposed method will be analyzed by executing and discussing several experiments. Finally in Chapter 5, some conclusions and recommendations for future work will be given.

2

Previous work

2.1 Introduction

The problem of using a sequence of images from an observed scene to reconstruct the structure of the scene and/or the motion of the observer through the scene is a common problem in Computer Vision and it is referred to in literature as the Structure from Motion (SfM) problem [4]. In this section, a general introduction to this problem will be given.

2.1.1 Camera model

The input to the SfM problem are images obtained by camera(s) observing the (unknown) scene. Each image consists of a 2-dimensional projection of the 3-dimensional scene on the image plane. The key of a SfM algorithm is to combine the information from all images in order to estimate a 3-dimensional reconstruction of the observed 3-dimensional scene and to estimate the motion of the camera through this scene. The relation between an observed image and the unknow scene can, in the most general case, be modeled by the pinhole camera model [5]. A schematic view of the model is displayed in Figure 2.1.

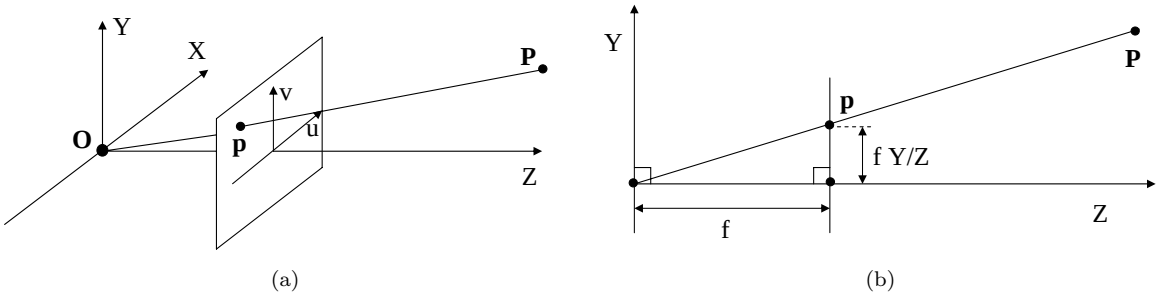


Figure 2.1: Pinhole camera geometry

Essentially, this pinhole camera model can be described by

$$\begin{aligned} u &= f\frac{X}{Z} \\ v &= f\frac{Y}{Z} \end{aligned} \tag{2.1}$$

where $\boldsymbol{p} = [u, v]^T$ is the image coordinate and $\boldsymbol{P} = [X, Y, Z]^T$ is the 3-dimensional coordinate of the imaged point. The focal length is given by f, which is the distance from the center of projection to the image plane, expressed in the unit of u and v (usually pixels). In computer vision, the perspective projection is often described as a linear mapping of homogeneous coordinates. This linear transformation

is given by

$$\boldsymbol{p} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P} = \boldsymbol{K} [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P} \tag{2.2}$$

The matrix $\boldsymbol{K}$ used here is called the calibration matrix, and describes the intrinsic parameters. So far we have only included the focal length of the camera. To model a more complete behavior of the pinhole camera, the calibration matrix can be extended to include the other intrinsic parameters of the camera like a non-zero principal point $[u_0 v_0]^T$, skew of the axes γ and non square pixels resulting in different values for the focal length along the u and v direction. The complete matrix $\boldsymbol{K}$ is then given by

$$\boldsymbol{K} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

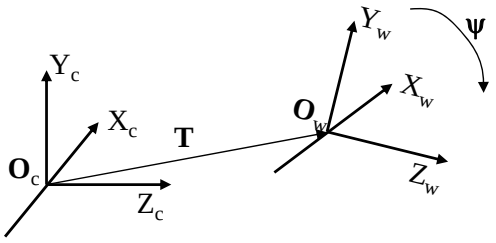


Figure 2.2: Scene and camera transformation

Because the scene can move relatively to the camera, a transformation is often used to describe the relation between world coordinates (the actual coordinates in the frame of reference in which we express the location points in the scene) and camera coordinates (the coordinates according to the frame of reference of the camera). This transformation consists of a rotation and a translation. The rotation is represented by three angles indicating the rotation about each axis and the translation by three distances indicating the distance traveled over each axis. Suppose that the observed scene, according to the observer, has rotated with an amount of $\boldsymbol{\Psi}$ and has translated with an amount of $\boldsymbol{T}$, then a point in the scene $\boldsymbol{P}_w$ will be visible to the observer at $\boldsymbol{P}_c$. The relation between the two is given by

$$\boldsymbol{P}_c = \boldsymbol{R}\boldsymbol{P}_w + \boldsymbol{T} \tag{2.4}$$

where $\boldsymbol{T} = [T_x, T_y, T_z]^T$ is the translation vector and $\boldsymbol{R}$ is a rotation matrix constructed from the individual angles $\boldsymbol{\Psi}$, defined as

$$\boldsymbol{R} = \begin{bmatrix} c\Psi_y c\Psi_z & c\Psi_y s\Psi_z & -s\Psi_y \\ s\Psi_x s\Psi_y c\Psi_z - c\Psi_x s\Psi_z & s\Psi_x s\Psi_y s\Psi_z + c\Psi_x c\Psi_z & s\Psi_x c\Psi_y \\ c\Psi_x s\Psi_y c\Psi_z + s\Psi_x s\Psi_z & c\Psi_x s\Psi_y s\Psi_z - s\Psi_x c\Psi_z & c\Psi_x c\Psi_y \end{bmatrix} \tag{2.5}$$

with $s\Psi = \sin(\Psi)$ and $c\Psi = \cos(\Psi)$. When homogeneous coordinates are used, this transformation can also be represented as a linear transformation. In homogeneous coordinates we write

$$\boldsymbol{P}_c = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix} \boldsymbol{P}_w \tag{2.6}$$

Eventually we are observing a point $\boldsymbol{P}_w$ in the scene and measure its projection $[u, v]^T$. The relation between the world coordinate point and its projection can be found when combining (2.1) and (2.4) leading to

$$\begin{aligned} u &= f \frac{\boldsymbol{R}^1 \boldsymbol{P}_w + T_x}{\boldsymbol{R}^3 \boldsymbol{P}_w + T_z} \\ v &= f \frac{\boldsymbol{R}^2 \boldsymbol{P}_w + T_y}{\boldsymbol{R}^3 \boldsymbol{P}_w + T_z} \end{aligned} \tag{2.7}$$

Here $\boldsymbol{R}^i$ means the i^{th} row of the matrix $\boldsymbol{R}$. To get a linear transformation, we use homogeneous coordinates again, and combine (2.2) and (2.6) resulting in

$$\boldsymbol{p} = [\boldsymbol{K} \,|\, \boldsymbol{0}] \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix} \boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R} \,|\, \boldsymbol{T}]\boldsymbol{P} \tag{2.8}$$

The expression can be combined into one matrix multiplication. The combined 3x4 matrix $\boldsymbol{M} = \boldsymbol{K}[\boldsymbol{R} \,|\, \boldsymbol{T}]$ is called in computer vision the camera projection matrix. It is important to note that two homogeneous coordinates actually represent the same point when they are equal up to a scale factor. The two homogeneous coordinates $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ represent the same point when $\boldsymbol{p}_1 = \lambda \boldsymbol{p}_2$ with $\lambda \neq 0$.

2.1.2 Correspondence problem

After imaging the scene with a camera, the next task is to extract useful information from the individual images. This information can only be extracted from the scene by proper illumination with a light source. Objects in the scene will reflect illuminated light and because of the different shapes and material properties of objects, these will become visible in the camera as pixels with a certain intensity value. In color cameras these intensity values are measured for each color channel independently. A particular point in the scene, will have a certain projection in the image plane. The projection of this point can be characterized by a small image patch (region in the image). When this particular point will change position, this image patch can be observed in some other location of the image. The changes of the image locations of projected points over time contains information about the projected point and the motion of this point with respect to the camera.

The correspondence problem is the problem of finding corresponding points in the image of the same 3-d point, between different images of the same scene (but with a possibly different point of view). When all locations in the image are considered, this results in a dense flow vector field of the whole image. However, not all parts of the image are uniquely distinguishable from their neighborhood. Different techniques were developed to overcome this problem, by trying to find very distinctive features in the first image, that can easily be tracked to their position in a second image. These distinctive features can be points, lines, curves or any other characteristic that have the property of being well distinguishable.

In most SfM algorithms, the correspondence part is separated from the algorithm and is assumed to be solved. The actual input to those algorithms is a list of correspondences and positions in different image frames, rather than the actual image measured with the camera. An important point to keep in mind however, is the fact that these correspondence measures are not ideal correspondences and are contaminated with noise. Other possible problems are occlusions (a correspondence can not be found because the point is not visible to the camera anymore) and false correspondences (the tracked feature is not a projection of the initial point anymore, but a look-a-like projection).

2.1.3 Reconstruction

When correspondences between images are found, the next task is to use these correspondences to reconstruct camera motion and a 3-d representation of the scene structure. Although a 3-dimensional reconstruction of the scene and camera translation is possible, this will be, in the absence of any absolute reference, a metric reconstruction (Euclidean up to a scaling factor) at most. Reconstruction methods from known correspondences can roughly be divided into three groups. These are *two-frame* methods, *multi-frame batch* methods and *multi-frame recursive* methods. In the following analysis of these three groups, we assume the type of correspondences are point correspondences. Furthermore we assume that the observed scene is a static scene.

two-frame With as little as two image frames of the same scene, both taken from a different standpoint, structure and motion can be reconstructed. When point correspondences are used, each point in the scene can be represented as a three dimensional point. For every observed point, a noisy two dimensional projection can be measured in both image frames. The motion that a camera has undergone between the two observations can be described by a rotation about three axes and a translation in three directions. When no absolute yardstick is used[1], the scene and camera

[1]The absolute yardstick can be for example a landmark in the scene, the distance between two points in the scene or a (partially) known camera translation

translation can at most be reconstructed up to an unknown global scale factor. When a calibrated camera is used, meaning the internal parameters of the camera are assumed to be known, we have five unknown camera parameters. Three for rotation and two for translation, because of the unknown scale factor. Suppose we observe n different points, then there are $3n$ unknown parameters describing each observed point in the scene. Every observed point introduces $4n$ non-linear equations, see (2.7). Each of this non-linear equations relates a measured image coordinate to a real world (possibly transformed) 3-d point. To solve this non-linear system of equations, we need to observe at least five points. However, due to the high non-linearity of the equations the problem is very complex and can lead to as much as ten different solutions [6]. An implementation of this non-linear approach can be found in [7].

To overcome the problem of the non-linearity, Longuet-Higgins [8] proposed a method that uses eight correspondences, leading to a linear system of equations. The method is based on calculating the essential matrix by using at least eight point correspondences. The essential matrix is a 3x3 matrix that encapsulates the geometry of two image frames of the same scene, taken by calibrated cameras. When more than eight points are used, the essential matrix can be calculated by linear least squares minimization. To obtain the translation and rotation, the calculated essential matrix can be decomposed into a rotation matrix and a translation vector.

When an uncalibrated camera is used there is another 3x3 matrix that can be used to encapsulate the geometry of two image frames. This matrix is called the fundamental matrix, and it can be calculated in the same linear way as the essential matrix, by using a set of at least eight corresponding points in both frames. A complete overview of methods to estimate the fundamental matrix can be found in [9].

multi-frame batch When more than two image frames are available, other techniques can be used to estimate structure and motion. In batch algorithms a solution is found by optimizing over the whole set of image frames, while constraining the structure to be rigid. Tomasi and Kanade [10] proposed a method to estimate object motion and structure from a set of orthogonal projection measurements. The method uses the factorization of a measurement matrix into an object structure and object motion matrix. Later Szeliski and Bing Kang [11] proposed a method that recovers 3-d object structure and object motion from perspective projection measurements. They make no use of a-priori information on object structure or object motion. The problem is formulated as a non-linear least squares optimization. Other examples of multi-frame batch methods can be found in [12], [13], [14], [15] and [16].

multi-frame recursive A batch algorithm finds a solution by optimizing over all available data. In the case that we want to work with estimates of the object structure and motion real time, i.e. have an estimate of structure and motion based on all the available data so far, a batch algorithm cannot be used anymore because it would require enormous storage and would be computationally unattractive. We have to use an online algorithm in that case. Such an online algorithm should have an estimate available based on all data so far and incrementally update this estimate by using the latest available data. This kind of estimation is called recursive estimation.

The problem can now be considered as a state estimation problem. The state being the current motion and structure parameters. To solve the state estimation problem we need to identify two models, the state space model and the measurement model. The state space model defines how a current state will evolve to the next state. In the SfM problem the structure would be modeled as being rigid, meaning no change in the structure parameters over time. The motion on the other hand can be modeled in different ways, for example as a random walk model by assuming no a priori information about the motion, only that it will move smoothly. The measurement model defines how the observations of the current state are related to the current state. In the SfM problem the observations can be points in the current image frame, one for each of the 3-d structure points. Each observed point is related to a 3-d structure point by a transformation and a camera projection (2.7).

The state estimation problem can be solved by using a state estimator. When the uncertainty in the models can be approximated as Gaussian uncertainty and the models are linear, the state can be calculated optimally with respect to the least squares estimation error by using a Kalman

Filter. When the models are non-linear, but the uncertainties introduces by the models are still Gaussian, an Extended Kalman Filter can (EKF) be used. An example of using an EKF in the SfM problem is the method of Azarbayejani [4]. This method assumes a camera with unknown focal length (uncalibrated), so the focal length is added to the state vector. Other examples of using EKF and SfM can be found in [17], [18], [19], [20], [21], [22], [23], [24], [25] and [26].

2.2 Recursive structure from motion estimation

There are a lot of different approaches to handle the SfM problem all with different constraints and implementations. For this project however, we are interested only in online methods. Most online methods can be set in the same framework. We will give here a basic framework for the recursive approach to the structure from motion problem. To explain the framework, we start with a basic introduction to state estimation. Then we identify the models that are needed to do SfM state estimation and finally we show two possible approaches of SfM state estimation. In upcoming section 2.3 we will give an example of two worked out implementations for both approaches found in literature.

2.2.1 State estimation

As explained before, the SfM problem can be formulated as a state estimation problem. This requires us to identify a model, with the following components

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{\eta}_{\boldsymbol{x}}) \tag{2.9}$$

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t, \boldsymbol{\eta}_{\boldsymbol{z}}) \tag{2.10}$$

Here $\boldsymbol{x}$ is a vector in a predefined state space and $\boldsymbol{x}_t$ represents the state of the system at time t. The true state to estimate however is hidden and only some properties of this state can be observed by an observer and are represented by the vector $\boldsymbol{z}$. An observation at time t is indicated by $\boldsymbol{z}_t$. To complete the model we have to specify the functions f and h. The function f is referred to as the state equation and it defines the transition from a given state at time t, $\boldsymbol{x}_t$, to the next state $\boldsymbol{x}_{t+1}$. Because in general, the next state cannot be determined exactly from the current state, the non-deterministic part of this transition is given by a random vector $\boldsymbol{\eta}_{\boldsymbol{x}}$. The function h is referred to as the measurement equation and it defines how the observations are related to the state. Also here there is a non-deterministic part $\boldsymbol{\eta}_{\boldsymbol{z}}$ which describes the amount of noise present in the observations.

With the state equation and measurement equation at hand, we can define two probability density functions. First we will define the probability of a next state while given all previous states, denoted by $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_{1:t})$. Since we constructed the state space so that the state equation can be used to predict a new state from only the current state (2.9), without considering any other previous states, this transition pdf can be written as a markov chain

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_{1:t}) = p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \tag{2.11}$$

The measurement equation can be used to construct the pdf of a measurement, given all available data $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t-1})$. Here too, we can conclude that this measurement is only dependent on the current state, because that is how we defined our sensor model in (2.10). This leads to

$$p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t-1}) = p(\boldsymbol{z}_t|\boldsymbol{x}_t) \tag{2.12}$$

Suppose that an initial guess of our state at time $t = 1$ is available. We can represent the uncertainty on this first state by the pdf $p(\boldsymbol{x}_1)$. With the available observation $\boldsymbol{z}_1$ at $t = 1$ we can update our knowledge on $\boldsymbol{x}_1$ by using Bayes rule as

$$p(\boldsymbol{x}_1|\boldsymbol{z}_1) = \frac{p(\boldsymbol{z}_1|\boldsymbol{x}_1)p(\boldsymbol{x}_1)}{p(\boldsymbol{z}_1)} = \frac{p(\boldsymbol{z}_1|\boldsymbol{x}_1)p(\boldsymbol{x}_1)}{\int p(\boldsymbol{z}_1|\boldsymbol{x}_1)p(\boldsymbol{x}_1)d\boldsymbol{x}_1} \tag{2.13}$$

After this initialization, we can recursively update our estimate on the state $\boldsymbol{x}_t$ by predicting a new state and update this prediction with an observation at each time step. This way, all available knowledge will

be optimally combined in the estimate of our state. Each recursion starts with a current estimate of the state, given by $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$. This posterior pdf encapsulates all available information on the state so far. The transition pdf will now be used to obtain a predicted next state as

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t}) = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{z}_{1:t})p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})d\boldsymbol{x}_t = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})d\boldsymbol{x}_t \tag{2.14}$$

An updated estimate on the next state can be calculated by taking the newest measurement $\boldsymbol{z}_{t+1}$ into account. Using Bayes rule again gives us

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t+1}) = \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{t+1}, \boldsymbol{z}_{1:t})p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t})}{p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_{1:t})} = \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{t+1})p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t})}{p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_{1:t})} \tag{2.15}$$

with

$$p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_{1:t}) = \int p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{t+1})p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t})d\boldsymbol{x}_{t+1} \tag{2.16}$$

This recursive algorithm can be executed online, by just continuously updating the pdf's. However, depending on the linearity of the state and measurement equations and the characterization of the state and measurement noise, an analytical solution to the problem might be difficult to obtain. For a complete overview of state estimation see for example the book of van der Heijden [27].

We will now see how the SfM problem can be adapted to a state estimation problem by formulating a state space and measurement model.

2.2.2 State space

We are interested in estimating the pose and structure of an object, so that we can project this object in our own local frame of reference, which corresponds to the camera frame of reference. Therefore the state space should consist of a part that indicates the pose of an object and a part that represents its structure. Since point based correspondences will be used, the structure of an object will be expressed as a set of 3-d points representing 3-d locations in an object frame of reference. The pose of the object is now defined as the transformation from the object frame of reference to the camera frame of reference. Because we assume that the object is rigid, this transformation is a Euclidean transformation, given by a rotation and a translation, see (2.4).

Altogether, we can parameterize the state as follows. The translation will be represented by the vector $\boldsymbol{T} = [T_x, T_y, T_z]^T$, containing the individual translation components over the three axes. The rotation will be represented by the vector $\boldsymbol{\Psi} = [\Psi_x, \Psi_y, \Psi_z]^T$, containing the individual rotations about each axis. These two vectors represent the pose of the object or actually the coordinate transformation from the local object frame to the camera frame. We will refer to these parameters as the motion parameters, since they describe the relative motion between the camera and the object. The object structure will be represented by a set of N points $\{\boldsymbol{P}_i\}_{i=1}^N$, where each individual point is a vector in 3-d space $\boldsymbol{P}_i = [X, Y, Z]^T$. Combining all these parameters gives the following state vector

$$\boldsymbol{x} = [\boldsymbol{T}^T, \boldsymbol{\Psi}^T, \boldsymbol{P}_1^T, \ldots, \boldsymbol{P}_N^T]^T \tag{2.17}$$

The next thing to identify is the state equation (2.9). If we look at the physical process, a rigid object moving around in a 3-d environment, we can easily propose the following model.

$$\begin{aligned} \boldsymbol{T}_{t+1} &= \boldsymbol{T}_t + \boldsymbol{\eta}_{\boldsymbol{T}} \\ \boldsymbol{\Psi}_{t+1} &= \boldsymbol{\Psi}_t + \boldsymbol{\eta}_{\boldsymbol{\Psi}} \\ \boldsymbol{P}_{i,t+1} &= \boldsymbol{P}_{i,t} \end{aligned} \tag{2.18}$$

The transformation is modeled here as a random walk process with $\boldsymbol{\eta}$ being zero mean Gaussian noise. This model assumes no a-priori knowledge on the object motion, except that it varies smoothly and with magnitude distributed according to $\boldsymbol{\eta}$. This model could be extended by adding motion velocity to the state and assuming that also this velocity varies smoothly. Actually this means we assume that the objects bear inertia, which is true in reality. The transition of $\boldsymbol{T}$ and $\boldsymbol{\Psi}$ from time t to $t+1$ is then fully

captured by their current state plus the velocity state. This would lead to the following model for the motion

$$\begin{aligned} \boldsymbol{T}_{t+1} &= \boldsymbol{T}_t + \dot{\boldsymbol{T}}_t \\ \dot{\boldsymbol{T}}_{t+1} &= \dot{\boldsymbol{T}}_t + \boldsymbol{\eta}_{\dot{\boldsymbol{T}}} \\ \boldsymbol{\Psi}_{t+1} &= \boldsymbol{\Psi}_t + \dot{\boldsymbol{\Psi}}_t \\ \dot{\boldsymbol{\Psi}}_{t+1} &= \dot{\boldsymbol{\Psi}}_t + \boldsymbol{\eta}_{\dot{\boldsymbol{\Psi}}} \end{aligned} \tag{2.19}$$

where $\dot{\boldsymbol{T}}$ and $\dot{\boldsymbol{\Psi}}$ indicate the velocities of the translation and rotation respectively.

The rigidness of the object is expressed by modeling the points as random constants (i.e. their true state does not change over time). The model could be made a little bit more loose with respect to the rigidness of the object by also modeling points as random walk processes with Gaussian noise having low variance.

$$\boldsymbol{P}_{i,t+1} = \boldsymbol{P}_{i,t} + \boldsymbol{\eta}_{\boldsymbol{P}} \tag{2.20}$$

This allows the object to undergo small deformations with respect to the rigid structure.

2.2.3 Measurements

A camera will be used to observe the object or scene. Each 3-d point $\boldsymbol{P}_i$ on the object that is visible in the image frame will be observed as a 2-dimensional projection $\boldsymbol{z}_i = [u_i, v_i]^T$ in the image frame. The measurement vector contains all these measurements stacked together as

$$\boldsymbol{z} = [u_1, v_1, \ldots, u_N, v_N]^T \tag{2.21}$$

All points have an individual measurement $\boldsymbol{z}_i$ which is independent on other points. For each of the points, the relation between the true state and the measurements is given by (2.7). This measurement equation takes into account the motion of the camera so far (indicated by the transformation) and the perspective projection of the camera. We see indeed that the measurements (according to the modeled measurement equation) are mutually independent.

The measurements are not ideal measurements of the true state but are corrupted with noise. The physical sources of this noise can be found in the camera and the tracking algorithm. The camera introduces quantization noise due to a limited pixel resolution and the camera lens can introduce non-linear relations in the mapping from world coordinates to image coordinates. The tracking algorithm is responsible for finding point feature correspondences between two image frames. Tracking can introduce noise because the appearance of a tracked image template can change over time due to illumination changes or perspective distortion due to a changed point of view. These noise sources can be approximated by additive Gaussian noise [28], so that each measurement is related to the true state by

$$\boldsymbol{z}_i = proj(\boldsymbol{P}_i, \boldsymbol{\Psi}, \boldsymbol{T}) + \boldsymbol{\eta}_z \tag{2.22}$$

Here *proj* is the perspective projection function defined by (2.7). The measurement noise is given by $\boldsymbol{\eta}_z$, which in this case is modeled as a Gaussian random vector with with zero mean and variance equal for both u and v.

2.2.4 Kalman Filtering approach

The Kalman Filter [29] is a state estimator which represents the posterior state pdf as a Gaussian distribution. In fact, all pdfs involved in the state estimation are represented in the Kalman Filter approach by Gaussian distributions. This allows a compact representation, since a Gaussian distribution is completely described by its mean and covariance. The Kalman Filter is the optimal state estimator for linear systems with Gaussian uncertainty with respect to the minimum mean square error. However, since the SfM problem is a non-linear problem we will use a modified version, called the Extended Kalman Filter (EKF) which can be used for non-linear problems.

In the SfM problem, the state equation (2.18) is linear, so that we can write

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{\eta}_{\boldsymbol{x}}) = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{\eta}_{\boldsymbol{x}} \tag{2.23}$$

with $\boldsymbol{A}$ the state transition matrix and $\boldsymbol{\eta_x}$ being a Gaussian random vector with zero mean and covariance $\boldsymbol{Q}$. We assume a posterior distribution, defined as a Gaussian pdf $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) \sim \mathcal{N}(\hat{\boldsymbol{x}}_t, \boldsymbol{P}_t)$ is available. Because of the linear state transition (2.23), the predicted next state can be obtained straightforward by using (2.14), leading to

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t}) \sim \mathcal{N}(\boldsymbol{A}\hat{\boldsymbol{x}}_t, \boldsymbol{A}\boldsymbol{P}_t\boldsymbol{A}^T + \boldsymbol{Q}) = \mathcal{N}(\hat{\boldsymbol{x}}_{t+1}^-, \boldsymbol{P}_{t+1}^-) \tag{2.24}$$

The next step would be to include our current measurement $\boldsymbol{z}_t$ to update the predicted state estimate. The measurement equation (2.22) however is non-linear, which makes the true Gaussian propagation of pdfs impossible. To overcome this problem, the measurement equation will be approximated by a first order Taylor series, evaluated around the predicted next state $\hat{\boldsymbol{x}}_t^-$

$$\boldsymbol{z}_{t+1} = h(\boldsymbol{x}_{t+1}, \boldsymbol{\eta_z}) \approx h(\hat{\boldsymbol{x}}_{t+1}^-) + \boldsymbol{H}(\boldsymbol{x}_{t+1} - \hat{\boldsymbol{x}}_{t+1}^-) + \boldsymbol{\eta_z} \tag{2.25}$$

here the matrix $\boldsymbol{H}$ is the Jacobian of the measurement function evaluated at $\hat{\boldsymbol{x}}_{t+1}^-$, defined as

$$\boldsymbol{H} = \frac{\partial h}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}_{t+1}^-) \tag{2.26}$$

and $\boldsymbol{\eta_z}$ is additive Gaussian noise with covariance $\boldsymbol{R}$. The linearized measurement equation will be used to update the predicted next state in order to get the posterior state pdf. The solution to equation (2.15) can be obtained as follows [27]

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t+1}) \sim \mathcal{N}(\hat{\boldsymbol{x}}_{t+1}^- + \boldsymbol{K}_{t+1}(\boldsymbol{z}_{t+1} - h(\hat{\boldsymbol{x}}_{t+1}^-)), (\boldsymbol{I} - \boldsymbol{K}_{t+1}\boldsymbol{H})\boldsymbol{P}_{t+1}^-) = \mathcal{N}(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{P}_{t+1}) \tag{2.27}$$

where $\boldsymbol{K}_{t+1}$ is called the Kalman gain and can be calculated as

$$\boldsymbol{K}_{t+1} = \boldsymbol{P}_{t+1}^-\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{t+1}^-\boldsymbol{H}^T + \boldsymbol{R})^{-1} \tag{2.28}$$

Recursively repeating these calculations lead to the recursive estimation of the SfM problem.

2.2.5 Particle Filtering approach

A Particle Filter [30] is a recursive filter where the posterior state distributions (2.15) are approximated by a set of weighted particles (samples). Each sample represents a possible realization of the current state with an associated weight. The associated weight of each sample indicates the probability of this realization. Suppose we have a set of N samples, $\{w_t^{(i)}, \boldsymbol{x}_t^{(i)}\}_{i=1}^N$, representing the current state. The posterior state distribution can then be approximated by

$$p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)}\delta(\boldsymbol{x} - \boldsymbol{x}_t^{(i)}) \tag{2.29}$$

where δ stands for the Dirac delta function. Total sum of the weight factors has to be equal to one, since $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$ is a pdf which integral over the complete input space should equal one.

The main advantage of a Particle Filter, in comparison with Kalman Filtering based methods, is the fact that the representation of the posterior state distribution is not limited to a Gaussian shape. In fact, with this sampling based method it is possible to approximate any kind of pdf. Because of this, there are also no limitations on the state space model and measurement model. The propagation of a Gaussians is limited to linear systems only, but since we are not using Gaussians anymore we are not limited to linear systems either.

In order to propagate our set of samples representing the posterior state at time t to a posterior state at time $t+1$, we have to draw samples from the posterior distribution given in (2.15). However, in the general case it is not possible to sample directly from this distribution. This problem can be solved by introducing a proposal distribution that we are able to sample. A requirement for this proposal density is that it overlaps the support of the new posterior distribution. The problem can then be formulated as follows. Suppose our target distribution is $p(\boldsymbol{x})$, from which it is not possible to sample, and our proposal

distribution is $q(\boldsymbol{x})$ which can be used to take samples from. By sampling the proposal distribution, it will be approximated by

$$q(\boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \tag{2.30}$$

which can be used to get a weighted sample set to represent the target distribution

$$\begin{aligned} p(\boldsymbol{x}) &= \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}q(\boldsymbol{x}) &\approx \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\frac{1}{N}\textstyle\sum_{i=1}^{N}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \\ &= \frac{1}{N}\textstyle\sum_{i=1}^{N}\frac{p(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \\ &= \frac{1}{N}\textstyle\sum_{i=1}^{N}w^{(i)}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \end{aligned} \tag{2.31}$$

In this formulation we assumed that $p(\boldsymbol{x})$ could be evaluated up to a normalizing constant so that its total integral over the input space equals one. This is however no requirement, since we can also use a function $\pi(\boldsymbol{x})$, provided that $\pi(\boldsymbol{x}) \propto p(\boldsymbol{x})$. By normalizing the sum of weights to equal one again, the true function $p(\boldsymbol{x})$ can still be approximated. The approximation (2.31) in that case will be

$$p(\boldsymbol{x}) \approx \sum_{i=1}^{N}\frac{\frac{\pi(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}}{\sum_{j=1}^{N}\frac{\pi(\boldsymbol{x}^{(j)})}{q(\boldsymbol{x}^{(j)})}}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) = \sum_{i=1}^{N}\tilde{w}^{(i)}\delta(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \tag{2.32}$$

This is the basic principle of a generic particle filter. There are quite some implementations of the particle filter, varying mostly in the choice of proposal distribution and sampling scheme used. Overviews of available particle filters can be found in [31], [30] and [32].

2.3 Example Methods

In this section two possible implementations of the recursive SfM problem will be discussed. A characterizing difference between the two methods is that in the first method state estimation will be handled by an Extended Kalman Filter while in the second method, the state estimation is handled by a particle filter. We will reference the first method as *Azarbayejani* and the second method as *Qian*, based on the names of the authors of the papers describing both methods.

The measurements used in both methods come from a single camera observing the scene. From the images captured by the cameras, point features are extracted and tracked over time. The locations of these point features are used as the measurements of the state space.

2.3.1 Azarbayejani

The method of Azarbayejani [4] estimates motion, structure and focal length simultaneously. Focal length is estimated, but can be removed from the state when a camera with fixed focal length is used. The state estimation is handled by an Extended Kalman Filter. Also a modified camera model, based on the pinhole model but with a different representation, was introduced here.

State space

The method estimates both motion and structure simultaneous. Motion is characterized here by a translation and a rotational velocity. The translation is represented straightforward by three parameters $\boldsymbol{T} = [T_x, T_y, T_z]^T$ describing the translation between the object frame of reference and the camera frame of reference. The rotational velocity $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ is used to update an external (not included in the state space) rotation matrix $\boldsymbol{R}$, that relates the rotation between the object frame and camera frame, see (2.4). So in order to represent the camera motion, the state space contains the following parameters

$$\boldsymbol{x}_m = [T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]^T \tag{2.33}$$

In this method, the camera frame is related to the pinhole frame by a translation along the z-axis in the positive direction with an amount of f (representing focal length). The pinhole camera model (2.1) has its origin in the center of projection (COP), but in this method the origin is chosen to be at the center of the image plane ($\boldsymbol{O}_C$). The situation is depicted in Figure 2.3. The resulting relation between

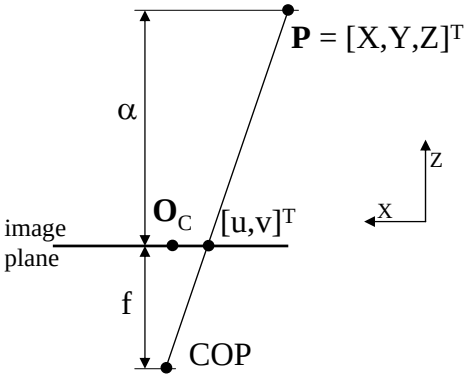


Figure 2.3: Geometry in Azarbayejani's method

image points and camera frame coordinates is thus given by

$$\begin{aligned} u &= f\frac{X_P}{Z_P} = f\frac{X}{Z+f} = \frac{X}{1+\beta Z} \\ v &= f\frac{Y_P}{Z_P} = f\frac{Y}{Z+f} = \frac{Y}{1+\beta Z} \end{aligned} \tag{2.34}$$

where $\beta = \frac{1}{f}$. When an uncalibrated camera is used, in the sense that the focal length is not known, the parameter β can be added to the state space

$$\boldsymbol{x}_c = [\beta] \tag{2.35}$$

The object structure will be represented in the state space only by the depth component of each point. The object frame and camera frame are initially the same, meaning complete 3-d information can be obtained by using the image locations of each point in the first frame. We get this complete 3-d point by inverting (2.34). The estimated depth of each point is captured by a parameter α so that $Z = \alpha$, leading to

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (1+\alpha\beta)u \\ (1+\alpha\beta)v \\ \alpha \end{bmatrix} \tag{2.36}$$

To represent object structure, for each object the depth parameter α is estimated and the image location $[u, v]^T$ from the first frame is not included in the state space but saved as an external parameter. However, since the measurements of the feature locations in the first frame are corrupted with noise, we will end up with biased positions of the reconstructed points. This noise is mainly due to limited pixel resolution and lens distortions. To overcome this problem, the bias in the feature location of each point in the first frame will be estimated as well. This will lead to the following reconstruction formula

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (1+\alpha\beta)(u+b_u) \\ (1+\alpha\beta)(v+b_v) \\ \alpha \end{bmatrix} \tag{2.37}$$

The parametrization of object structure in the state space now consists of three parameters per 3-d point. For each point we will have the following elements in the state vector

$$\boldsymbol{x}_s = [\alpha, b_u, b_v] \tag{2.38}$$

Models

The dynamics of the model are based on the assumption of observing a static, rigid scene. This means the propagation of the object parameters is straightforward and will be modeled as a random constant. The motion parameters are not constant, but do not depend on each other. There is no prior information about this motion, except that it is smooth from frame to frame. This kind of information will be modeled by a random walk model with small uncertainty. This results in a linear state transition model for the object motion

$$\boldsymbol{x}_{s,t+1} = \boldsymbol{A}\boldsymbol{x}_{s,t} + \boldsymbol{\eta}_{\boldsymbol{x}_s} \tag{2.39}$$

The matrix $\boldsymbol{A}$ is an identity matrix of appropriate dimensions. The vector $\boldsymbol{\eta}_{\boldsymbol{x}}$ is a zero mean random vector with a covariance matrix $\boldsymbol{Q}$, having only some of its diagonal elements non-zero. The non-zero elements correspond to the variance of the associated motion parameter.

The measurement model is non-linear and is composed of the camera model (2.34) and a transformation of all structure points with the estimated object motion (2.4). The global rotation matrix, used in the measurement model is updated with the estimated rotational velocity. Each measurement of a feature point can be handled separately. This results in the following form of the measurement model, where h is the non-linear model, $\boldsymbol{x}_{s,i,t}$ is the state of point i, $\boldsymbol{x}_{m,t}$ are the motion parameters and $\boldsymbol{R}_{t-1}$ is the externally updated rotation matrix,

$$\boldsymbol{z}_{i,t} = h(\boldsymbol{x}_{s,i,t}, \boldsymbol{x}_{m,t}, \boldsymbol{R}_{t-1}) + \boldsymbol{\eta}_{\boldsymbol{z}} \tag{2.40}$$

Limitations

The scene, observed by the camera, is assumed to be limited in a way that it can not extend the scene observed by the initial frame of the camera. Feature points are selected in the first image and these are the only points used in the estimate. The initial feature points are tracked through the sequence of images and SfM is based on these measurements only. Using this assumption simplifies the reconstruction of the full 3-d location of each feature point. The 3-d location of a point consists of three dimensions, but when considering the feature location in the first image, it is essentially reduced to a problem of recovering only one dimension.

Another limitation of this method is its robustness to outliers in the measurements. The Extended Kalman Filter assumes a Gaussian distribution on the measurement noise. When outliers are not detected, the estimated state will be biased, possibly leading to an unstable filter.

In an application where the criteria of a static scene are not met, the filter will also not work. It can not handle multiple independent motions in the scene.

2.3.2 Qian

In the method of Qian [33], structure and motion are estimated by using a sampling based method. Due to the high dimensionality of the problem, the estimation is split into separate estimation processes. First camera motion, consisting of rotation and translation direction, will be estimated. From this estimate the translation magnitude of the camera motion will be estimated. Finally, from those both estimates the depth of each structure point will be estimated. The reason for this division is the otherwise high dimensionality of the problem. The complete state vector consisting of camera motion and object structure can be very high. By dividing the estimation in several subprocesses, the dimensionality of each subprocess is reduced so that a sampling based method can be used.

Implemented particle filter

For all three estimation steps, the implemented particle filter is based on the same scheme. As proposal distribution the prior distribution is used, i.e. $q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t+1}) = p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t})$. This results, according to (2.31), in the following expression for calculation of the weights

$$w_{t+1}^{(i)} = w_t^{(i)} p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{t+1}^{(i)}) \tag{2.41}$$

An overview of the three estimation processes and how they are related to each other is displayed in Figure 2.4.

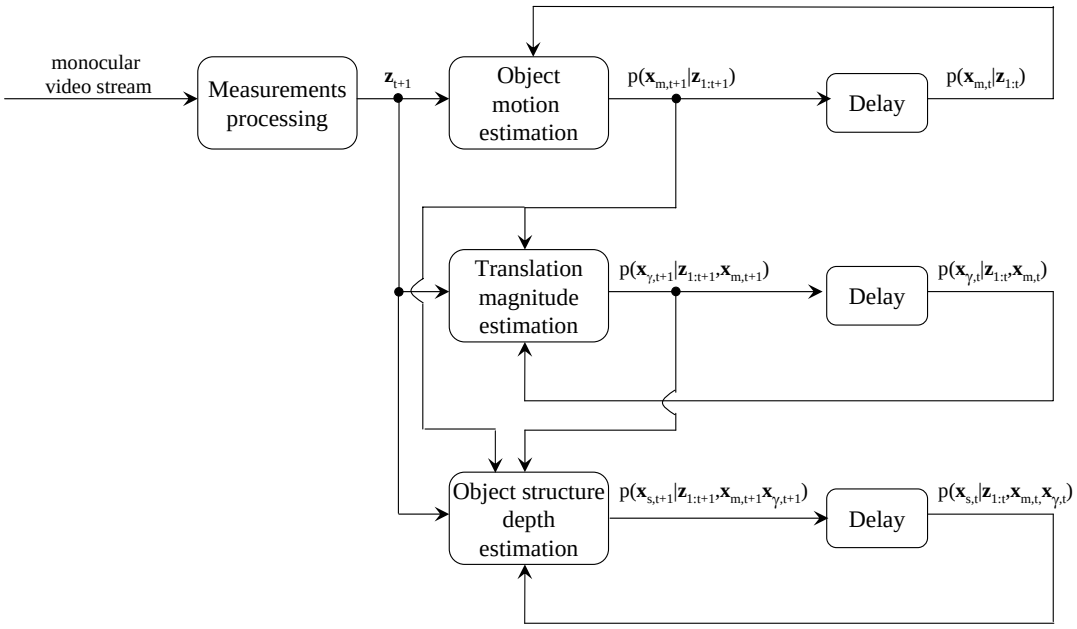


Figure 2.4: Overview of the state estimation in the proposed method

Motion estimation

In this step, rotation and translation direction will be estimated. Because a single camera is used, distances can only be recovered up to a global scale factor. Translation is therefore only represented in this first step by its direction, the magnitude of the translation is not estimated yet. A velocity motion model is used to represent both rotation and translation direction. This results in the following state space for the first estimation step

$$\begin{aligned} \boldsymbol{x}_m &= [\Psi_x, \Psi_y, \Psi_z, \alpha, \beta]^T \\ \dot{\boldsymbol{x}}_m &= [\dot{\Psi}_x, \dot{\Psi}_y, \dot{\Psi}_z, \dot{\alpha}, \dot{\beta}]^T \end{aligned} \tag{2.42}$$

Here $\boldsymbol{x}_m$ represents the current rotation and translation direction and $\dot{\boldsymbol{x}}_m$ the rotational and translational direction velocity. The rotations are represented by $\boldsymbol{\Psi} = [\Psi_x, \Psi_y, \Psi_z]^T$ and the normalized translation vector can be reconstructed from the directions α and β by $\boldsymbol{T} = [\sin(\alpha)\cos(\beta), \sin(\alpha)\sin(\beta), \cos(\alpha)]^T$. The transition model for this state space is defined to be

$$\begin{aligned} \boldsymbol{x}_{m,t+1} &= \boldsymbol{x}_{m,t} + \dot{\boldsymbol{x}}_{m,t} + \boldsymbol{\eta}_m \\ \dot{\boldsymbol{x}}_{m,t+1} &= \dot{\boldsymbol{x}}_{m,t} + \boldsymbol{\eta}_{\dot{m}} \end{aligned} \tag{2.43}$$

The uncertainties $\boldsymbol{\eta}_m$ and $\boldsymbol{\eta}_{\dot{m}}$ are chosen such that uncertainty in propagation of the positions is much smaller than the uncertainty in propagation of the velocities.

The measurement model relates each of the observed feature locations to the current state. Because the object structure (indicating the positions of the observed points in our local reference frame) is not represented in the state of this subprocess yet, the observed feature locations in the first image are used in the computation. By using these first image measurements together with the current image frame measurements and the predicted object motion, the measurement likelihood can be calculated by using epipolar geometry. The feature location in the first frame $\boldsymbol{z}_1$ and the predicted motion $\boldsymbol{x}_m$ can be used to project an epipolar line in the current image frame. The situation is shown in Figure 2.5. Suppose we are measuring the point $\boldsymbol{P}_i$. The observation of this point in the first frame is denoted by $\boldsymbol{z}_{i,1}$ and the observation in the current frame by $\boldsymbol{z}_{i,t}$. The measurement likelihood can now be evaluated by considering all points on the epipolar line

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t) = \int_{\boldsymbol{P}_l}^{\boldsymbol{P}_u} p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{P}) p(\boldsymbol{P}) d\boldsymbol{P} \tag{2.44}$$

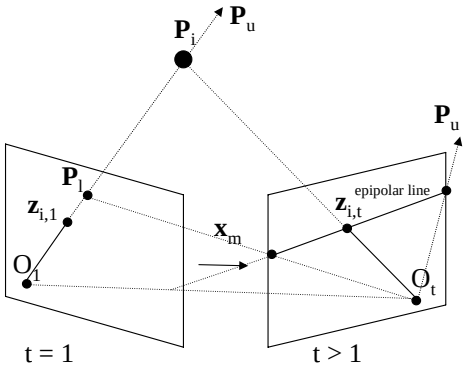


Figure 2.5: Epipolar geometry

The likelihood $p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{P})$ is evaluated by calculating the distance of the measurement to the corresponding point on the epipolar line. This distance can be transformed to a measure of likelihood given the variance of the measurement noise. Since no a priori information about $\boldsymbol{P}_i$ is available, except that it lies in front of both views, all points on the epipolar line will be considered with equal probability.

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t) \propto \int_{\boldsymbol{P}_l}^{\boldsymbol{P}_u} p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{P}) d\boldsymbol{P} \tag{2.45}$$

The boundaries of the integral can be calculated by considering that part of the epipolar line where $\boldsymbol{P}_i$ would still lie in front of both views. The approach used here is only valid in the case of non-zero translation. In the case of pure rotation, there is no epipolar geometry and the camera rotation can be calculated directly by using the feature locations from the first frame and the current measurements.

After estimating the motion, we end up with a posterior distribution of the motion parameters represented by a set of weighted samples

$$p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t}) \approx \sum_{i=1}^{N} w_{m,t}^{(i)} \delta(\boldsymbol{x}_{m,t} - \boldsymbol{x}_{m,t}^{(i)}) \tag{2.46}$$

The weights are calculated using (2.45) and the samples are drawn from the state transition model (2.43).

Translation magnitude estimation

With the given weighted set of samples representing $p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t})$, the next step is to estimate the translation magnitude. So far the translation was represented by a unit vector and now a magnitude will be assigned to it, thereby restricting the depth of the last feature point in the state space to be set to one. Thus translation magnitude will be estimated so that the depth value of the last feature point is normalized to one. The reason to set the depth of one of the feature points to one, is the fact that the object structure is supposed to be rigid, i.e. not constant in its own reference frame. The camera translation magnitude however, will change over time when the camera moves through the scene.

The translation magnitude will be denoted with γ in a state vector $\boldsymbol{x}_\gamma = [\gamma]$. The combined state with motion and translation magnitude is represented by

$$p(\boldsymbol{x}_{m,t}, \boldsymbol{x}_{\gamma,t}|\boldsymbol{z}_{1:t}) = p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t}) p(\boldsymbol{x}_{\gamma,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}) \tag{2.47}$$

In order to represent this combined motion and translation magnitude state as a set of weighted samples, the marginalized translation magnitude state will be represented by a set of weighted samples as well, $p(\boldsymbol{x}_{\gamma,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}^{(i)}) \approx \sum_{j=1}^{N} w_{\gamma,t}^{(i,j)} \delta(\boldsymbol{x} - \boldsymbol{x}_{\gamma,t}^{(i,j)})$. The combined posterior distribution can then be represented by the weighted sample set $\left\{ \{ w_{m,t}^{(i)} w_{\gamma,t}^{(i,j)}, [\boldsymbol{x}_{m,t}^{(i)}, \boldsymbol{x}_{\gamma,t}^{(i,j)}] \}_{j=1}^{M} \right\}_{i=1}^{N}$.

Calculation of the next marginalized translation magnitude posterior distribution is done by using Bayes rule resulting in

$$p(\boldsymbol{x}_{\gamma,t+1}|\boldsymbol{z}_{1:t+1}, \boldsymbol{x}^{(i)}_{m,t+1}) = \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{\gamma,t+1}, \boldsymbol{x}^{(i)}_{m,t+1})p(\boldsymbol{x}_{\gamma,t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}^{(i)}_{m,t})}{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}^{(i)}_{m,t+1})} \tag{2.48}$$

Samples will be drawn from

$$\boldsymbol{x}^{(i,j)}_{\gamma,t+1} \sim p(\boldsymbol{x}_{\gamma,t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}^{(i)}_{m,t}) \tag{2.49}$$

and weights will be calculated as

$$w^{(i,j)}_{m,t+1} = \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}^{(i,j)}_{\gamma,t+1}, \boldsymbol{x}^{(i)}_{m,t+1})}{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}^{(i)}_{m,t+1})} \tag{2.50}$$

The likelihood function $p(\boldsymbol{z}_{t+1}|\boldsymbol{x}^{(i,j)}_{\gamma,t+1}, \boldsymbol{x}^{(i)}_{m,t+1})$ uses the fact that the position of the last feature in the state space has a depth value of one.

This summarizes the global idea of how the translation magnitude is estimated from the sampled posterior distribution and formed into an extended sample set representing both motion and translation magnitude. For details on how the likelihoods are estimated and what prior distributions are used, see [33].

Feature depth estimation

In the final estimation step, the depth for each feature is estimated, based on the previously estimated motion and translation magnitude. After that a final sample set is created which contains the feature depth for each feature, the translation magnitude and the camera motion. The steps used here are roughly the same as in the translation magnitude estimation step.

Multiple objects

In a succeeding paper of Qian [1], the problem is extended to handle the case of multiple independently moving objects. To do so a validity vector $\boldsymbol{v}$, which indicates a measure of membership for each of the points in the state space to the corresponding motion sample, is added to the state. A high validity value for a point means that the motion of that motion sample is consistent with the respective point. The validity value for a given point and motion sample is updated according to the distance of the measurement of the feature point to the corresponding epipolar line. When this distance is larger than a certain threshold the validity value is decreased and when this distance is smaller than the threshold the validity value is increased. The vector $\boldsymbol{v}^{(i)}_{t+1}$, indicating the validity vector of the i'th motion sample at time $t+1$ is updated element wise, for the j'th element of that vector this is according to

$$v^{(i)}_{j,t+1} = \gamma v^{(i)}_{j,t} + \xi(e(\boldsymbol{x}^{(i)}_{m,t}, \boldsymbol{z}_{j,t})) + \eta_v \tag{2.51}$$

where γ is an exponential forgetting factor close to 1 and η_v represents noise in the validity value. $\xi(e)$ is a function of the distance to the epipolar line given the motion and the current measurement. It is implemented as

$$\xi(e) = \begin{cases} (\frac{e_{th}}{e+1})^2 & e < e_{th} \\ (\frac{e_{th}}{e+1})^2 - \frac{e+1}{e_{th}} & e \geq e_{th} \end{cases} \tag{2.52}$$

The likelihood for each motion sample will be calculated by taking the 7 strongest features for a drawn motion sample, based on their validity values. The set of 7 strongest features is denoted by $\mathcal{A}$. When there are no 7 positive validity values for this motion sample, the motion sample will be discarded. In the other case, the likelihood will be calculated based on the 7 strongest features and their distance to the epipolar line in the current image. The likelihood is then assumed to be proportional to

$$p(\boldsymbol{z}_t|\boldsymbol{x}_{m,t}) \propto e^{\frac{-\epsilon}{\sigma_u^2 + \sigma_v^2}} \sum_{i \in \mathcal{A}} \boldsymbol{v}^{(i)}_t \tag{2.53}$$

here ϵ can be seen as the weighted squared epipolar distance of the 7 strongest feature points, given by

$$\epsilon = \frac{\sum_{i \in \mathcal{A}} e_t^{(i)2} \boldsymbol{v}_t^{(i)}}{\sum_{i \in \mathcal{A}} \boldsymbol{v}_t^{(i)}} \tag{2.54}$$

When all weight values are calculated according to (2.53), the next step is to cluster features with similar validity values into the same cluster. In order to do so the set of weighted samples is sorted so that the highest weight come first. This sorted list is then iterated and based on the signs of validity values, cluster centers are generated. To generate a new cluster, a minimum cluster size of P_m is defined. A new cluster center is generated when the current motion sample in the iterated list has at least P_m positive unclustered validity values. The indices of the positive validity values are set as cluster center of the new cluster. When such a new cluster center is found, all other motion samples will be checked to be part of this cluster. This check involves looking the number of coinciding indices in the cluster center and the positive validity indices of the current motion sample. If more than P_m entries coincide, the check is positive. When a motion sample is positively checked it is added to that cluster. After this iteration a new possible cluster is looked for by iterating the list again, starting from the position just after the last cluster center and looking at validity vectors currently not present in any already generated cluster centers. The process is repeated until no new clusters can be formed. When some motion samples remain unclustered after all, they are added to a new, last cluster.

In pseudo code the clustering algorithm can be formulated as

1. Sort the motion samples, according to the calculated weights, such that the motion sample with the highest weight comes first
2. Generate sample cluster centers
   (a) Assume there are already K clusters generated. The indices of the existing cluster centers are ignored in the generation of a possible new cluster.
   (b) Browse through the sorted motion sample list downwards. When a sample in this list has a validity vector with more than P_m positive entries, not contained in any of the existing cluster centers, generate a new cluster. The center of this cluster will contain all the indices of positive entries in the validity vector of the current motion sample.
      i. For the generated cluster, search through the whole list of motion samples for compatible motions by looking at their validity vectors.
      ii. When at least P_m entries in the validity vector are also indices of the current cluster center, add this motion sample to the current cluster.
   (c) When there are less than P_m entries not contained in any cluster center, stop the loop, otherwise continue until all motion samples in the sorted list have been visited.
3. If there are any motion samples not contained in any of the above generated clusters, add those remaining motion samples to a new cluster.

The clusters are now used to balance the weighted sample set, so that all clusters have an equal share in the competition for samples. The balancing step consists of normalizing the weights in each cluster so that the sum of weights is equal to one. Then for each sample, the weights this sample has in all clusters is summed and divided by the total number of clusters. The result is a final balanced set of weights.

Limitations

Overall this method should be more robust than the Azarbayejani method. It allows for multiple hypotheses and does not get trapped in false local minima. The obtained accuracy however, could be less than the accuracy in Azarbayejanis method, since the estimation process is incrementally starting with camera motion, than including translation magnitude and finally including scene structure but without feeding back the incremented knowledge. Camera motion is estimated only on a very rough guess of the scene (using epipolar geometry). This guess is based only on feature measurements in the first frame,

leaving at least one dimension (Z-direction) for all features undetermined. When camera motion estimation is done, translation magnitude and scene structure are estimated in subsequent steps. The resulting translation magnitude and scene structure estimates are not used again in the next step of camera motion estimation. Better results might be obtained when using the translation magnitude and scene structure estimates again in the next motion estimation step.

Another drawback of this method, just like Azarbayejanis, is the assumption of a non expanding scene. The only scene that will be reconstructed here needs to be visible in the first frame already.

3

Proposed method

3.1 Introduction

Given the previous work and the scope of our project, it is now time to propose a method which solves our problem of recovering object structure and motion from image streams. We will start with an overview of the biggest limitations of the previous method. Solutions for these limitations will be found and finally we will propose a method which is designed to solve the current limitations.

3.1.1 Requirements

Recall that the goal of the project is to develop a local navigation system for blind people. In the ideal case, this local navigation system would work in any kind of environment and without limitations. Based on the previously implemented method, we can analyze the typical limitations of this method and try to find solutions for these limitations.

Dynamic scenes One of the biggest challenges is the fact that a normal real life scene is not static at all. The previous method assumed the scene was static. This assumption reduces the problem to the estimation of one rigid motion in the scene. In order to deal with a more complex and dynamic scene, we will express the scene as the composition of an unknown number of independently moving rigid objects. This adds a new problem to the task, namely the segmentation of the scene into rigidly moving objects. The segmentation can be based on the fact that all points belonging to the same object share the same motion. After the segmentation, structure and motion should be estimated for each of the moving objects.

Occlusions The previous method does not handle occlusions. Occlusion occurs when a feature that is being tracked can be seen by the camera anymore because the observed point is hidden/occluded by some other object. A closely related problem is the problem of features falling of the frame at one of the borders because the observed point is outside the camera's field of view. Both problems should be handled in order to make the method a robust method.

Expanding scenes Introducing new points in the scene, which were not observed in the first frame of the sequence, is a problem that is not dealt with in the previous method. Points are selected in the first frame and these are the only points that will be tracked and whose positions will be estimated. This simplifies the reconstruction method, however it is important that the method also works for expanding scenes.

Global scale factor The previous method reconstructs the scene up to a global scale factor. This means only relative sizes and distances in the scene are known. In our application, however, it is important to have the absolute sizes and distances. To recover this unknown scaling factor, some sort of reference of which we know the exact distance or size will be needed. This reference can be obtained by using stereo vision instead of images obtained by a single camera or by incorporating inertial acceleration measurements in the method.

Robustness We assume that the measurements are approximately distributed according to some predefined pdf. However, sometimes a measurement can be far away from the expected measurement and is not in accordance with the measurement model. Such a measurement is called an outlier. The previous method could not handle outliers and outliers would lead to instability problems in the filter. Outliers are likely to occur with this kind of measurement based on feature tracking.

Based on these limitations of the previous method, a new method will be proposed which ultimately should be resistant to the discussed problems.

3.1.2 Method overview

A schematic overview of the proposed method is displayed in Figure 3.1. We will briefly comment on each of the blocks in this method and in subsequent sections a more in depth description of all parts will be given.

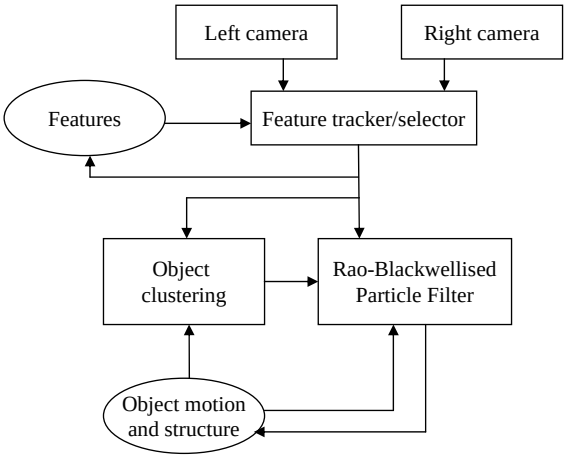


Figure 3.1: Overview of the method

The input to the method is given by two cameras, which we will name left camera and right camera. These two cameras together form the stereo vision system. Images from both cameras are used every time step to extract the positions of features. Features are the projection pairs of 3-d points onto the image planes of the camera. A list of these features will be maintained with information about the feature and when it has been observed. A mechanism will be used to remove old features from this list and select new features, based on the current camera images.

The stereo camera that is used is a calibrated stereo camera, meaning the intrinsic parameters of the setup are known. The stereo images are rectified first, so that (horizontal) epipolar lines in both images will be aligned. From these images a measurement in three dimensions, u, v and d will be obtained. The parameters u and v account for the x and y coordinate in the left frame. The parameter d is called the disparity and indicates the difference in the x coordinates between the left and right image. These measurements of each tracked 3-d point will be input to the rest of the algorithm.

The remaining part of the method is responsible for clustering the points into independently moving rigid objects and estimating the structure and motion for each of the clustered objects. The clustering is possible because we assume the objects are rigid, so that all points belonging to the same object share the same object motion with respect to the camera. The object motion defines the transformation of the object frame with respect to the camera frame, which will be represented by a translation and a rotation. A Rao-Blackwellized particle filter will be used to represent the state. The state in this particle filter consists of a structure part, a motion part and a membership part.

3.2 Stereo camera geometry

A stereo camera consists of two separate cameras, both observing the same scene but having a different center of projection or optical center. The advantage of a stereo camera over a single camera is the fact that by combining two views, taken from different locations, a 3-d image of the scene can be generated. A requirement for this is that both views overlap, i.e. that they both observe the same scene. The scene that is visible in both views can be reconstructed in 3 dimensions. However, there remains a big uncertainty in the estimated depth values.

3.2.1 Triangulation

We can think of the stereo setup as a left camera and a right camera. Both cameras observe the scene and the observations are interrelated by the transformation between the reference frames of both cameras. Both cameras have their own reference frame defined so that the the optical center coincides with the origin, the positive Z-axis points along the optical axis and the X- and Y-axes span the image frame, see Figure 2.1. The relation between the right (r) and the left (l) frame can be given by the a rigid transformation, consisting of a rotation matrix $\boldsymbol{R}$ and translation $\boldsymbol{T}$

$$\boldsymbol{P}_r = \boldsymbol{R}\boldsymbol{P}_l + \boldsymbol{T} \tag{3.1}$$

For each of the cameras, the 3-d points are related to image points by a perspective projection, according to the pinhole model (2.1). Combining the two projections and the rigid transformation gives us a linear system with four equations and three unknowns. We assume here that the focal length is equal for both cameras.

$$\begin{bmatrix} f & 0 & -u_l \\ 0 & f & -v_l \\ u_r R_{31} - f R_{11} & u_r R_{32} - f R_{12} & u_r R_{33} - f R_{13} \\ v_r R_{31} - f R_{21} & v_r R_{32} - f R_{22} & v_r R_{33} - f R_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f T_x - u_r T_z \\ f T_y - v_r T_z \end{bmatrix} \tag{3.2}$$

The translation is given here by the components of the vector $\boldsymbol{T} = [T_x, T_y, T_z]^T$. The rotation is given by the components R_{ij} of the rotation matrix $\boldsymbol{R}$, where i denotes the row and j the column. This system can be solved to get the solution to the general case with unconstrained geometry.

A common stereo setup is obtained with two parallel cameras, separated by a horizontal displacement. The stereo geometry is then constrained such that the transformation between the two cameras is reduced to a translation T_x along the X-axis and the rotation is set to zero. In this case, the relation between 3-d points and projected image coordinates is more simple. The rotation matrix corresponding to no rotation is the identity matrix. Two of the four equations will become equivalent, so that the linear system reduces to

$$\begin{bmatrix} f & 0 & -u_l \\ 0 & f & -v_l \\ -f & 0 & u_r \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f T_x \end{bmatrix} \tag{3.3}$$

with the solution

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{T_x}{u_r - u_l} \begin{bmatrix} u_l \\ v_l \\ f \end{bmatrix} \tag{3.4}$$

The stereo geometry with parallel cameras is displayed in Figure 3.2. The translation along the X-axis T_x is usually referred to as the baseline and indicated with b. The measurement of the right coordinate u_r is usually expressed as the distance in image coordinates between the left and right measurement and referred to as the disparity value $d = u_l - u_r$. In the case of parallel cameras, we finally end up with three measures, the two image coordinates in the left image frame u and v and the disparity value d. The process of recovering a 3-d location from these measurements is called triangulation.

3.2.2 Rectification

The stereo configuration with parallel cameras, as described above, is an ideal configuration in the sense that a) the simple geometry results in simple relations between image coordinates and b) the search

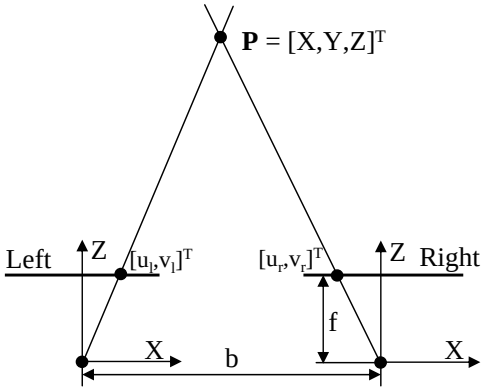


Figure 3.2: Parallel stereo geometry

between corresponding points in the image plane results in a search along a 1-d horizontal line in the second image. The fabrication of a perfect parallel stereo camera is quite difficult, small deviations in the alignments of the cameras are hard to prevent. However, images of a given (not perfectly parallel) stereo camera can be transformed so that they could have been originated from a perfect parallel camera. The transformation of original stereo images into parallel stereo images is called rectification.

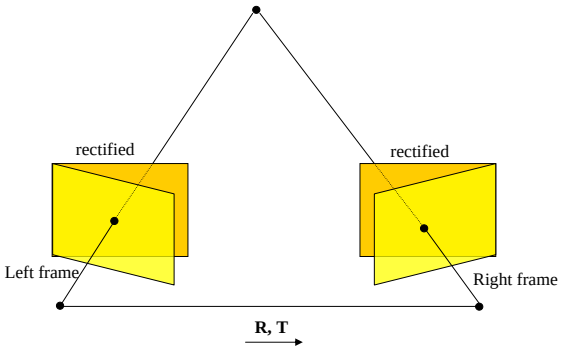


Figure 3.3: Stereo rectification

The rectification should thus lead to two transformations that transform each of the image planes to a rectified image plane. The geometric constraints on the rectified image planes is that epipolar lines in both image planes should be horizontal, so that corresponding points in both images have the same vertical coordinate.

The method used here is based on the method of Fusiello [34], however we assume that the intrinsic and extrinsic camera parameters are already separated, as opposed to the combined camera projection matrix that is used by Fusiello. In this method we will use the reference frame of the left camera as our (temporary) world coordinate frame. As a consequence, the origin of our world coordinate frame corresponds to the optical center of the left camera. Furthermore we assume that the external parameters, representing the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ relating the right camera frame of reference to the world frame of reference are known. The internal calibration matrices of both cameras are also assumed to be known. We can use the camera projection matrix to define a linear relation between the 3-d point and the projected point, using homogeneous coordinates (2.8). The original equations describing the relation between world coordinate points and image points in the left and right camera frame are then given by

$$\begin{aligned} \boldsymbol{p}_l &= \lambda_1 \boldsymbol{K}_l [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P}_w \\ \boldsymbol{p}_r &= \lambda_2 \boldsymbol{K}_r [\boldsymbol{R} \,|\, \boldsymbol{T}] \boldsymbol{P}_w \end{aligned} \tag{3.5}$$

where λ_1 and λ_2 are unimportant scale factors.

The key task of the rectification algorithm now, is first to find a rotation that rotates the world coordinate frame so that the proposed geometry of the stereo camera is met. We will denote the required rotation matrix with $\boldsymbol{R}_n$, where the subscripted n stands for new. This rotation matrix is build up from three vectors, each responsible for the rotation about each axis

$$\boldsymbol{R}_n = \begin{bmatrix} \boldsymbol{r}_x^T \\ \boldsymbol{r}_y^T \\ \boldsymbol{r}_z^T \end{bmatrix} \tag{3.6}$$

The new X-axis should be parallel to the baseline of the stereo system. The direction of the baseline is defined by the vector subtraction of the optical centers of both cameras, $\boldsymbol{C}_r - \boldsymbol{C}_l$. The X component of the rotation can now be found as the unit vector pointing along the baseline

$$\boldsymbol{r}_x = \frac{\boldsymbol{C}_r - \boldsymbol{C}_l}{||\boldsymbol{C}_r - \boldsymbol{C}_l||} \tag{3.7}$$

The optical center of the left camera was zero and the optical center of the right camera still needs to be found. This can be done easily by recalling that

$$\boldsymbol{P}_r = \boldsymbol{R}\boldsymbol{P}_l + \boldsymbol{T} \tag{3.8}$$

Setting $\boldsymbol{P}_r = \boldsymbol{0}$ and solving for $\boldsymbol{P}_l$ gives us the optical center of the right camera, expressed as a coordinate in the left camera frame. The solved optical center of the right camera then is $\boldsymbol{C}_r = -\boldsymbol{R}^T\boldsymbol{T}$. The X component of the new rotation is then defined as

$$\boldsymbol{r}_x = -\boldsymbol{R}^T \frac{\boldsymbol{T}}{||\boldsymbol{T}||} \tag{3.9}$$

The next rotation component, Y, should be orthogonal to X. Since there are no other geometrical constraints here, we can add a new constraint that minimizes the orientation difference between the original image planes and the rectified image plane. To minimize the orientation between an original image and the rectified image, we have to choose Y orthogonal to the original optical axis of the image. Because in general, the optical axes of the original images are not parallel, the least distortion is obtained when being orthogonal to the average direction $\boldsymbol{k}$ of both optical axes. This average direction is given by

$$\boldsymbol{k} = \boldsymbol{e}_3 + \boldsymbol{R}^T\boldsymbol{e}_3 = \boldsymbol{e}_3 + (\boldsymbol{R}^3)^T \tag{3.10}$$

Here $\boldsymbol{e}_3 = [0, 0, 1]^T$ and $\boldsymbol{R}^3$ is the third row of matrix $\boldsymbol{R}$. The Y component now has to be orthogonal to $\boldsymbol{k}$ and the X axis

$$\boldsymbol{r}_y = \frac{\boldsymbol{k}}{||\boldsymbol{k}||} \times \boldsymbol{r}_x \tag{3.11}$$

The final rotation component, Z, in turn has to be orthogonal to both the X and Y axes

$$\boldsymbol{r}_z = \boldsymbol{r}_x \times \boldsymbol{r}_y \tag{3.12}$$

The cross product is used here to obtain the orthogonal directions. Given all these vectors, the full rotation matrix (3.6) is now known.

The rectified image planes are parallel now, but in order to have corresponding vertical coordinates for corresponding points in both rectified planes, we also have to define a new calibration matrix $\boldsymbol{K}_n$. Both images should be rectified with the same calibration matrix in order to fullfill to the corresponding vertical coordinates constraint. Especially the chosen focal length of the rectifying calibration matrix is important. Varying the focal length will shrink or enlarge the rectified image. We will determine the largest focal length f_n from the original camera matrices and use this in our rectifying calibration matrix

$$\boldsymbol{K}_n = \begin{bmatrix} f_n & 0 & 0 \\ 0 & f_n & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

The principal point will be set to zero for simplicity.

With the calculated rotation matrix and projection matrix, we can define a transformation from the original images to rectified images. We wil show how this transformation can be calculated. For the left image we have

$$\begin{array}{rclcrcl} \boldsymbol{p}_{ol} &=& \lambda_1 \boldsymbol{K}_l [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P}_w & \Longleftrightarrow & [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P}_w &=& \frac{1}{\lambda_1} \boldsymbol{K}_l^{-1} \boldsymbol{p}_{ol} \\ \boldsymbol{p}_{nl} &=& \lambda_2 \boldsymbol{K}_n [\boldsymbol{R}_n \,|\, \boldsymbol{0}] \boldsymbol{P}_w & & & & \\ &=& \lambda_2 \boldsymbol{K}_n \boldsymbol{R}_n [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P}_w & \Longleftrightarrow & [\boldsymbol{I} \,|\, \boldsymbol{0}] \boldsymbol{P}_w &=& \frac{1}{\lambda_2} \boldsymbol{K}_n^{-1} \boldsymbol{R}_n^T \boldsymbol{p}_{nl} \end{array} \tag{3.14}$$

The subscript *ol* stands for original left and *nl* means new (rectified) left. This result can be used to find an expression for the transformation from $\boldsymbol{p}_{ol}$ to $\boldsymbol{p}_{nl}$

$$\boldsymbol{p}_{nl} = \lambda_3 \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_l^{-1} \boldsymbol{p}_{ol} \cong \boldsymbol{H}_l \boldsymbol{p}_{ol} \tag{3.15}$$

where $\boldsymbol{H}_l$ is the projective transformation defined by $\boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_l^{-1}$. The unimportant scale factor λ_3 is replaced here with the notion of equal up to scale factor. For the right image, we use $\boldsymbol{C}_r = -\boldsymbol{R}^T \boldsymbol{T}$ and note that the location of the optical center does not change to write

$$\begin{array}{rclcrcl} \boldsymbol{p}_{or} &=& \lambda_1 \boldsymbol{K}_r [\boldsymbol{R} \,|\, -\boldsymbol{R}\boldsymbol{C}_r] \boldsymbol{P}_w & & & & \\ &=& \lambda_1 \boldsymbol{K}_r \boldsymbol{R} [\boldsymbol{I} \,|\, -\boldsymbol{C}_r] \boldsymbol{P}_w & \Longleftrightarrow & [\boldsymbol{I} \,|\, -\boldsymbol{C}_r] \boldsymbol{P}_w &=& \frac{1}{\lambda_1} \boldsymbol{R}^T \boldsymbol{K}_r^{-1} \boldsymbol{p}_{or} \\ \boldsymbol{p}_{nr} &=& \lambda_2 \boldsymbol{K}_n [\boldsymbol{R}_n \,|\, -\boldsymbol{R}_n \boldsymbol{C}_r] \boldsymbol{P}_w & & & & \\ &=& \lambda_2 \boldsymbol{K}_n \boldsymbol{R}_n [\boldsymbol{I} \,|\, -\boldsymbol{C}_r] \boldsymbol{P}_w & \Longleftrightarrow & [\boldsymbol{I} \,|\, -\boldsymbol{C}_r] \boldsymbol{P}_w &=& \frac{1}{\lambda_2} \boldsymbol{R}_n^T \boldsymbol{K}_n^{-1} \boldsymbol{p}_{nr} \end{array} \tag{3.16}$$

resulting in the following transformation from $\boldsymbol{p}_{or}$ to $\boldsymbol{p}_{nr}$

$$\boldsymbol{p}_{nl} = \lambda_3 \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{R}^T \boldsymbol{K}_r^{-1} \boldsymbol{p}_{or} \cong \boldsymbol{H}_r \boldsymbol{p}_{or} \tag{3.17}$$

with the projective transformation $\boldsymbol{H}_r$ defined by $\boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{R}^T \boldsymbol{K}_r^{-1}$

After rectification, correspondences found in the rectified images can be used to calculate the 3-d position of the projected point. The baseline b is equal to $||\boldsymbol{T}||$ and the focal length f is defined in the rectifying calibration matrix $\boldsymbol{K}_r$. Correspondences in the rectified image can be found along epipolar lines, resulting in equal vertical coordinates $v = v_l = v_r$. The difference in horizontal coordinates is called the disparity d and is equal to $u_r - u_l$. By setting $u = u_l$ we now have three parameters that relate to the original 3-d position of the point (3.4)

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{b}{d} \begin{bmatrix} u \\ v \\ f \end{bmatrix} \tag{3.18}$$

The rectification algorithm was tested with two stereo images[1], obtained by a calibrated stereo setup. In Figure 3.4, the image planes are set out in 3-d space according to the stereo geometry of the setup. The rectification algorithm was then performed and the transformed images, together with several epipolar lines are displayed in Figure 3.5.

3.2.3 Summary

This section summarizes the rectification method in pseudo code. In order to let the rectification work properly the internal camera matrices $\boldsymbol{K}_l$ and $\boldsymbol{K}_r$ are needed. Furthermore the external parameters $\boldsymbol{R}$ and $\boldsymbol{T}$ that relate the right image frame to the left, according to (3.1), are needed as well. Given these parameters, the rectification can be summarized as

1. Calculate rotation matrix $\boldsymbol{R}_n$ to properly align the image frames via the following steps

   (a) Calculate first row of $\boldsymbol{R}_n$

$$\boldsymbol{r}_x = -\boldsymbol{R}^T \frac{\boldsymbol{T}}{||\boldsymbol{T}||}$$

[1]The images originate from http://www-rocq.inria.fr/ tarel/syntim/paires.html, copyright by "Institut National de Recherche en Informatique et Automatique", 1994-2004

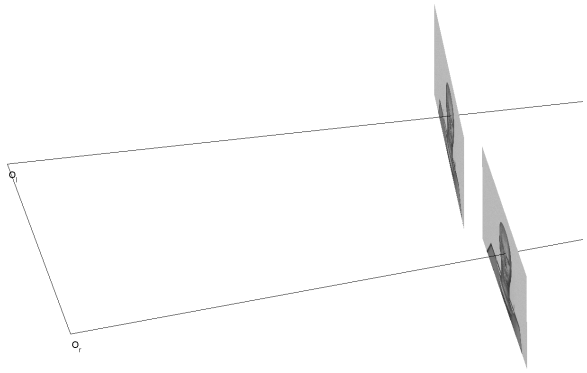
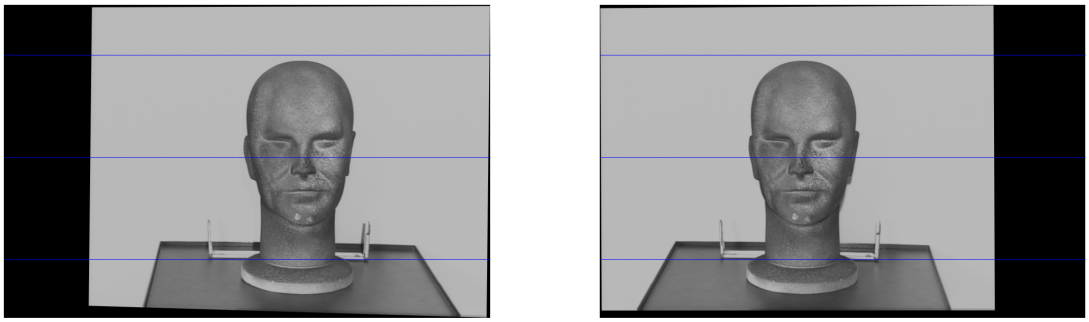


Figure 3.4: Geometry of stereo setup



(a) (b)

Figure 3.5: Rectified stereo images

(b) Calculate average direction $\boldsymbol{k}$ of both optical axes

$$\boldsymbol{k} = \boldsymbol{e}_3 + (\boldsymbol{R}^3)^T$$

(c) Calculate second row of $\boldsymbol{R}_n$

$$\boldsymbol{r}_y = \frac{\boldsymbol{k}}{||\boldsymbol{k}||} \times \boldsymbol{r}_x$$

(d) Calculate third row of $\boldsymbol{R}_n$

$$\boldsymbol{r}_z = \boldsymbol{r}_x \times \boldsymbol{r}_y$$

(e) Combine rows to form $\boldsymbol{R}_n$

$$\boldsymbol{R}_n = \begin{bmatrix} \boldsymbol{r}_x^T \\ \boldsymbol{r}_y^T \\ \boldsymbol{r}_z^T \end{bmatrix}$$

2. Find maximum focal length f_n from the original calibration matrices $\boldsymbol{K}_l$ and $\boldsymbol{K}_r$, and compose the new calibration matrix

$$\boldsymbol{K}_n = \begin{bmatrix} f_n & 0 & 0 \\ 0 & f_n & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Calculate left rectification transform

$$\boldsymbol{H}_l = \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_l^{-1}$$

4. Calculate right rectification transform

$$\boldsymbol{H}_r = \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{R}^T \boldsymbol{K}_r^{-1}$$

5. Calculate baseline

$$b = ||T||$$

6. Use rectification transforms to transform the input images to rectified images

7. When correspondences in the rectified images are found, the following relation exists between the measured u, v and d values and the 3-d position of the point

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{b}{d} \begin{bmatrix} u \\ v \\ f_n \end{bmatrix}$$

3.3 Measurements

When rectified stereo images are available, the next task is to extract point features from the images. Corresponding features need to be found between the left and right stereo image at the same time and also between stereo pairs over time. We will first discuss how correspondences can be found between two images. Based on this correspondence method a criterium will be formulated to find features that are well suited for tracking.

3.3.1 Feature correspondences

The key of finding feature correspondences is finding the corresponding position of a feature in the second image, based on a known position in the first image. For now we will assume the two images are related to eachother by a simple displacement

$$\mathcal{I}_1(\boldsymbol{p} - \boldsymbol{d}) = \mathcal{I}_2(\boldsymbol{p}) \tag{3.19}$$

thereby making two assumptions on the image model:

1. The illumination of the scene is assumed to be constant, although in reality this is not necessarily true. Illumination can change when the viewpoint of the camera has changed, due to a different reflection of the object when viewed from a different angle. Illumination can also change when the second image was taken at a different time than the first image and the illumination conditions are not constant over time. However for a small change in viewpoint and a small difference in time, the assumptions can be used to constrain the image model to constant illumination.

2. Deformations due to a different perspective projection of the object are assumed to be zero. In reality this is not the case and a different viewpoint of the camera with respect to the object can change its appearance in the image. However small subsequent changes in viewpoint will justify this assumption, the deformation can be neglected then.

Features will be represented by a set of pixels S around a center $\boldsymbol{p}_c$. By using a set of pixels rather than a single pixel, we are able to incorporate more of the structure of the point instead of only a single illuminance value of the center pixel. The correspondence of two features in two different images can now be defined as the difference between the intensity values of the pixels of the template in the first image and a region under consideration in the second image. The region under consideration in the second image, S_2 is related to the position of the template S plus a translation of $\boldsymbol{d}$. An appropriate error term that can be minimized is the squared difference in the intensity pixels between the template and the region under consideration

$$E(\boldsymbol{d}) = \sum_{i \in S_2} [\mathcal{I}_1(\boldsymbol{p}_i - \boldsymbol{d}) - \mathcal{I}_2(\boldsymbol{p}_i)]^2 \tag{3.20}$$

The Kanade Lucas Tomasi (KLT) tracker [35] solves this problem by using a Taylor series expansion of $\mathcal{I}_1(\boldsymbol{p}_i - \boldsymbol{d})$ truncated to the linear term, evaluated around the template region

$$\mathcal{I}_1(\boldsymbol{p} - \boldsymbol{d}) \approx \mathcal{I}_1(\boldsymbol{p}) - (\boldsymbol{\nabla}\mathcal{I}_1)^T \boldsymbol{d} \tag{3.21}$$

Suppose the template region consists of the set of pixels $\{\boldsymbol{p}_i\}_{i=1}^n$. Substituting (3.21) into (3.20) results in minimizing a linear least squares problem of the form $||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||^2$ where

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial \mathcal{I}_1}{\partial x}(\boldsymbol{p}_1) & \frac{\partial \mathcal{I}_1}{\partial y}(\boldsymbol{p}_1) \\ \vdots & \vdots \\ \frac{\partial \mathcal{I}_1}{\partial x}(\boldsymbol{p}_n) & \frac{\partial \mathcal{I}_1}{\partial y}(\boldsymbol{p}_n) \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \mathcal{I}_1(\boldsymbol{p}_1) - \mathcal{I}_2(\boldsymbol{p}_1) \\ \vdots \\ \mathcal{I}_1(\boldsymbol{p}_n) - \mathcal{I}_2(\boldsymbol{p}_n) \end{bmatrix} \tag{3.22}$$

The solution to this problem is found by differentiating the error term with respect to $\boldsymbol{d}$ and setting it to zero, resulting in

$$\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = 0 \iff \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{3.23}$$

In [35] a matrix $\boldsymbol{G} = \boldsymbol{A}^T\boldsymbol{A}$ and vector $\boldsymbol{e} = \boldsymbol{A}^T\boldsymbol{b}$ are constructed. An iterative solution is then found by calculating an estimated displacement based on the initial $\boldsymbol{G}$ and a constantly updated $\boldsymbol{e}_k$

$$\boldsymbol{d}_k = \boldsymbol{d}_{k-1} + \boldsymbol{G}^{-1}\boldsymbol{e}_k \tag{3.24}$$

The region of interest is adjusted in each iteration by the estimated displacement $\boldsymbol{d}_k$ until convergence is obtained.

Several improvements where made to the original KLT tracker. First there was the extension to model the template by allowing affine deformations, instead of translations only. This was introduced by Shi and Tomasi in [36] and can be used to monitor the quality of the tracked features. Later Jin [37] extended this to handle changes in illumination as well.

Next frame matching

The KLT approach can directly be used in the tracking problem of tracking a feature in an image at time t to a corresponding feature in a next image at time $t + 1$. The assumptions that have to be made here are that small the time step between subsequent images is small and the movement of the camera is also small. The tracker will start in the second frame with a region of interest which is positioned at location of the tracked feature in the previous frame. By iteratively finding the solution to (3.24) until convergence is obtained (i.e. no change in the obtained displacement d_k anymore), the new position of the feature can be obtained.

Stereo matching

For the stereo matching, the problem is slightly different. Since we are using rectified stereo images, the horizontal epipolar lines can be used to find a correspondence. The problem is thus a matter of finding the correct horizontal displacement between the left image and the right image. We will denote the required displacement with d, which is also referred to as the disparity. Another constraint can be added by noting that since the observed object is in front of the camera, we only have to search the left side of the epipolar line, starting from the initial point at $d = 0$. To find a correspondence, the error term (3.20) will be evaluated for all pixel locations along the left section of the epipolar line. This can be done by using a step size of one pixel while walking along the epipolar line. When the whole section of the line has been considered, the location that minimized the error term can be used to initiate a modified KLT tracker. This modified tracker will only solve for the horizontal displacement, so the accuracy stereo correspondence can be upgraded to sub pixel accuracy. The modified tracker is derived from the KLT tracker by setting $\boldsymbol{x} = [d_x, 0]^T$ in (3.23).

3.3.2 Selecting features

Based on the tracking algorithm, we can decide what kind of features should be selected for tracking. The selected features should be suitable to the method used in the tracking algorithm. Since we use the

KLT tracker, which involves solving a linear system with coefficient matrix $\boldsymbol{G}$ (3.24), we can try to select features based on properties of this coefficient matrix. A criterium based on this matrix was introduced by Tomasi and Kanade in [35]. In order to obtain a reliable unique solution, the coefficient matrix should be well conditioned and the entries should be above the noise level of the image. Well conditioned means the eigenvalues of $\boldsymbol{G}$ should not differ several orders of magnitude. Being above noise level means the smallest eigenvalue of $\boldsymbol{G}$ should be above a minimum value. Since the maximum eigenvalue is bounded by the fact that intensity variations in a window are bounded by the maximum allowable pixel value, a sufficient criteria is that the minimum eigenvalue is above a certain threshold λ_t. If λ_1 and λ_2 are the two eigenvalues of $\boldsymbol{G}$, than a feature can be selected for tracking when

$$\min(\lambda_1, \lambda_2) > \lambda_t \tag{3.25}$$

The eigenvalues of the coefficient matrix have an interesting physical interpretation. Moving the feature window in the original image in an arbitrary direction $\boldsymbol{h}$, corresponds [28] (by Taylor truncation to the linear term) to a change in squared intensity over the feature window and the displaced feature window of

$$\begin{aligned} E(\boldsymbol{h}) = \textstyle\sum_{i=1}^{n}[\mathcal{I}(\boldsymbol{p}_i) - \mathcal{I}(\boldsymbol{p}_i + \boldsymbol{h})]^2 &\approx \textstyle\sum_{i=1}^{n}[(\boldsymbol{\nabla}\mathcal{I})^T\boldsymbol{h}]^2 \\ &= \boldsymbol{h}^T\Big(\textstyle\sum_{i=1}^{n}(\boldsymbol{\nabla}\mathcal{I})(\boldsymbol{\nabla}\mathcal{I})^T\Big)\boldsymbol{h} \\ &= \boldsymbol{h}^T\boldsymbol{G}\boldsymbol{h} \end{aligned} \tag{3.26}$$

Using $||\boldsymbol{h}|| = 1$ and eigenvector theory, the intensity change in an arbitrary direction is bounded by

$$\lambda_1 < E(\boldsymbol{h}) < \lambda_2 \tag{3.27}$$

the physical interpretation can now be given as

- $\lambda_1 \approx \lambda_2 \approx 0$ means no intensity changes when moving in any direction, the feature can be considered as a flat region of the image.
- $\lambda_1 \approx 0$ and $\lambda_2 \gg 0$ means only intensity changes when moving in one particular direction, the feature is an edge or ridge.
- $\lambda_1 \gg 0$ and $\lambda_2 \gg 0$ means high intensity changes in any direction, the feature is a corner.

This interpretations points out that features that can be tracked best are corner like features.

The two most used corner detectors are the Harris corner detector [38] and the KLT corner detector [36]. These two corner detectors are almost identical, they both use the eigenvalues of $\boldsymbol{G}$ to calculate a corner strength value. Harris uses the measure

$$R_h = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(\boldsymbol{G}) + k\,\mathrm{tr}(\boldsymbol{G})^2 \tag{3.28}$$

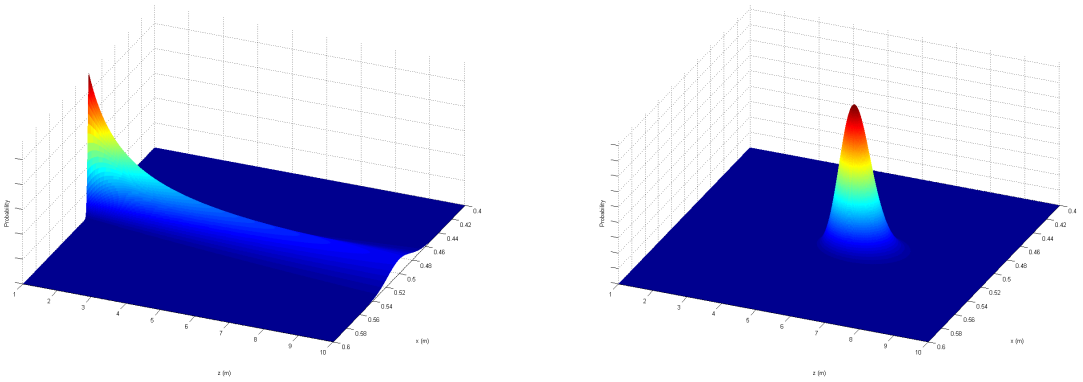
and the KLT corner detector uses the previously described measure

$$R_{KLT} = \min(\lambda_1, \lambda_2) \tag{3.29}$$

Trackable features are now selected as corners having the largest corner strength R.

3.3.3 Uncertainty in measurements

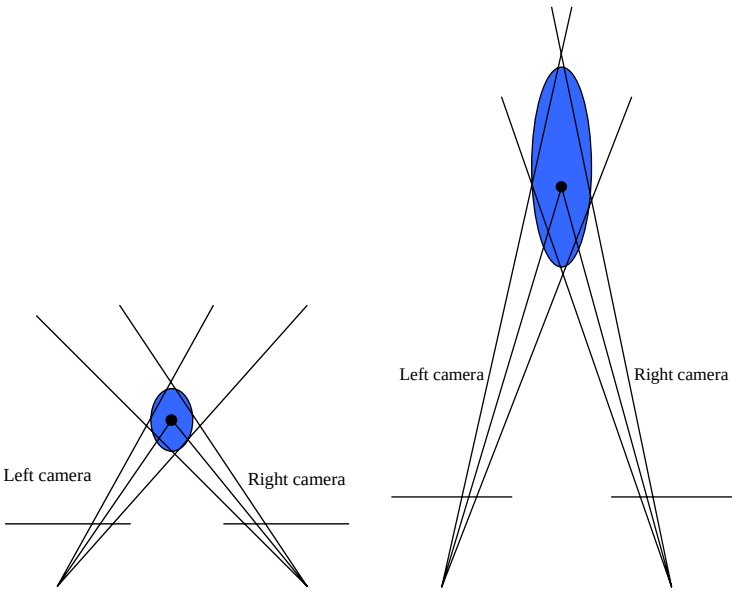
In this section, we will give some statements on the uncertainty in measurements and how these propagate to the uncertainty in a 3-d point. To start with the uncertainty analysis, we assume that the uncertainty in pixel position $\boldsymbol{p} = [u, v]^T$ of a tracked feature can be approximated by a Gaussian with a covariance of $\boldsymbol{\Sigma}_{\boldsymbol{p}} = \mathrm{diag}(\sigma_u^2, \sigma_v^2)$. This is the uncertainty on the position of $\boldsymbol{P} = [X, Y, Z]^T$ at a distance of $Z = f$ from the camera (i.e. the uncertainty in the focal plane). Along the ray, pointing from the optical center to the projection on the focal plane $\boldsymbol{p}$ of $\boldsymbol{P}$, this uncertainty will increase when the distance (Z) is increased. Since this relation is linear, a Gaussian propagation is possible. A simulation run of the uncertainty of measuring position with a single camera was done. The focal length was set to 6 mm, the horizontal size of the sensor set to 6.66 mm and the number of horizontal pixels to 640. An uncertainty in pixel coordinates of $\sigma_u = \sigma_v = 1$ pixels was used. The result of this simulation is displayed in Figure 3.6a.



(a) Single camera (b) Stereo camera

Figure 3.6: Probability density of position

Now we will extend our simulation to the case of a stereo setup. The two measurements can be fused together, by intersecting the uncertainty rays coming from both cameras. A Monte Carlo simulation was performed to simulate the resulting pdf of the position of the observed point. In this simulation the same camera as in the previous simulation was used twice, with a displacement of 10 cm along the X-axis and both cameras having the same orientation. The result of this simulation is displayed in 3.6b. The resulting pdf is not exactly Gaussian, but can be approximated by a Gaussian. The error that is made by this approximation is relatively small for points close to the camera and can be larger for points further away from the camera. The further away a point is, the heavier the tail of the distribution pointing away from both cameras will be. The situation is displayed in Figure 3.7.



(a) Short distance (b) Long distance

Figure 3.7: Gaussian assumptions on position uncertainty in measurements

For now we assumed that the uncertainty in u and v was equal for both the left and the right camera. In our description of the stereo geometry, we pointed out that the measurements will be given by u and v from the left camera and a disparity value d being the difference between the u values of the left and

right cameras. The uncertainty in this simplified representation of the measurements can be modeled by $\boldsymbol{p} = [u, v, d]^T$ with covariance $\boldsymbol{\Sigma_p} = \text{diag}(\sigma_u^2, \sigma_v^2, \sigma_d^2)$. The uncertainty of d, σ_d^2, is smaller than the uncertainty of u and v, because finding correspondences between two images of a parallel stereo setup is easier than finding correspondences across images with unknown translations. Recalling the relation between the observed point $\boldsymbol{P}$ and the measurement $\boldsymbol{p}$ defined in (3.18), we see the non-linearity. This means that the propagation of Gaussian uncertainty from a measurement to the uncertainty of the 3-d point is not straightforward. When the variances of $\boldsymbol{p}$ are not too big, the relation in (3.18), denoted here by the vector function f, can be linearized by using a Taylor series expansion truncated to the linear term around the measured value $\boldsymbol{p}_m$

$$\boldsymbol{P} = f(\boldsymbol{p}) \approx f(\boldsymbol{p}_m) + \boldsymbol{F}(\boldsymbol{p} - \boldsymbol{p}_m) \tag{3.30}$$

with $\boldsymbol{F}$ being the Jacobian

$$\boldsymbol{F} = \begin{bmatrix} (\boldsymbol{\nabla} f_X)^T \\ (\boldsymbol{\nabla} f_Y)^T \\ (\boldsymbol{\nabla} f_Z)^T \end{bmatrix} \tag{3.31}$$

This linearization can be used to approximate the 3-d position of a point based on Gaussian measurements as a Gaussian itself

$$p(\boldsymbol{P}|\boldsymbol{p}) \sim \mathcal{N}(f(\boldsymbol{p}_m), \boldsymbol{F}\boldsymbol{\Sigma_p}\boldsymbol{F}^T) \tag{3.32}$$

3.3.4 Summary

This section summarizes how measurements will be obtained from the rectified stereo images over time. The approach will be written in pseudo code.

Initialization

First we have to find suitable features in the image plane that can be used in the tracking process. This selection is based on the eigenvalues of the coefficient matrix $\boldsymbol{G}$, defined in (3.24). We will find suitable features from images obtained by the left stereo camera. The algorithm is defined as follows:

1. Calculate the image gradient over the whole image.
2. For each image point, calculate the cornerness measure, either the Harris measure

   $$R_h = \det(\boldsymbol{G}) + k\,\text{tr}(\boldsymbol{G})^2$$

   or the KLT measure

   $$R_{KLT} = \min(\lambda_1, \lambda_2)$$

   both based on the coefficient matrix $\boldsymbol{G}$, which can be calculated using the image gradients, see (3.22).
3. Find local maxima in the measures R over the whole image
4. Extract the best corners from the image, based on the calculated local maxima. Thresholding the best corners can be based on a maximum number of corners to find, or by setting a lower bound for the cornerness measure R.

Stereo matching

Once a feature has been selected for tracking, we have to match it in the right camera frame, so that it can be used as a measurement for the rest of the algorithm. The final output should be the position in the left frame $[u, v]^T$ and the disparity value d. For each corner, the disparity value d is calculated using the following algorithm:

1. Set $[u, v]^T$ to the position of the feature in the left image
2. Find the disparity value for this feature

(a) Search along the left section of the epipolar line in the right image, where $u_r \leq u_l$.

(b) For each displacement of 1 pixel in the left direction, calculate the error measure dependent on the disparity value $d = u_r - u_l$

$$E(d) = \sum_{i \in S} [\mathcal{I}_r(\boldsymbol{p}_i - [d, 0]^T) - \mathcal{I}_l(\boldsymbol{p}_i)]^2$$

(c) Find the global minimum of E, and use the disparity value d_{max} for which this minimum occurs to initialize a KLT tracker. When this miminum is above a certain threshold, a match could not be verified and the feature has to be discarded.

(d) Run the KLT tracker, initialized with $\boldsymbol{d} = [d_{max}, 0]^T$ and solve only for d_{max}. See the following section on how the KLT tracker will run.

Next frame matching

1. Set $\boldsymbol{p}_{t+1} = \boldsymbol{p}_t$, the position of the feature in the current frame to the position of the feature in the previous frame and set the unknown displacement to zero, $\boldsymbol{d}_0 = [0, 0]^T$. Set the iteration index to zero, $i = 0$

2. Calculate the coefficient matrix $\boldsymbol{G}$ from the image gradient in the region of the feature in the previous image, according to
$$\boldsymbol{G} = \boldsymbol{A}^T \boldsymbol{A}$$
where $\boldsymbol{A}$ contains the image gradient in the selected region of the previous image, according to (3.22).

3. Increment the iteration index, $i = i + 1$ and calculate the term $\boldsymbol{e}_i$ according to
$$\boldsymbol{e}_i = \boldsymbol{A}^T \boldsymbol{b}_i$$
where $\boldsymbol{b}_i$ is the difference in pixel intensity between the current feature region and the previous feature region, see (3.22). The intensity values in the current image are obtained by warping the image with the estimated displacement so far, $\boldsymbol{d}_{i-1}$.

4. Calculate the displacement according to
$$\boldsymbol{d}_i = \boldsymbol{d}_{i-1} + \boldsymbol{G}^{-1} \boldsymbol{e}_i$$

5. Do the iteration again when $\boldsymbol{d}_i - \boldsymbol{d}_{i-1}$ is not below a certain threshold (indicating convergence of the iteration) by jumping to 3.

6. The final value of $\boldsymbol{d}_i$ is used to calculate the new position of the tracked feature as
$$\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \boldsymbol{d}_i$$

3.4 Structure from motion estimation

In this section the proposed structure from motion algorithm will be explained. In order to do so, first the state space model will be discussed. We will show how we divided the state space into two parts, allowing for a mixed particle filter/Kalman filter representation and update mechanism of the state space. When the state space has been explained, we show in the next section how object structure and motion can be estimated for a static scene. When clustering information is available, the static scene situation is equivalent to handling one cluster at a time. Finally, by combining the object structure and motion estimation from all clusters, one particle set will represent all these different object motions. In the last section, we explain how the sample set can be clustered into one or more independently rigidly moving objects.

3.4.1 State space model

The state space model that will be used is more or less equal to much of the models seen in previous work. It basically consists of two parts, one representing object motion and one representing object structure, all stacked together in a state vector $\boldsymbol{x}$. The exact representation of this vector will be discussed further in this section. Measurements $\boldsymbol{z}$, as discussed in the previous section, are disparity space representations of 3-d structure points in the state. The dependencies between the measurements and the unobserved state, as given for the general case in (2.10) and of the future state given the current state (2.9), can be displayed in a Dynamic Bayesian Network (DBN) see Figure 3.8.

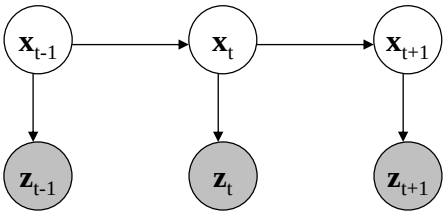


Figure 3.8: DBN for the state-space model

It is our intention to estimate the state with a particle filter, using a weighted set of samples. Each sample will be the realization of a possible state, represented as an n-dimensional vector. The dimensionality of this vector can get quite high, since it contains at least six dimensions for object motion and besides three dimensions for each structure point being estimated. For example, when 40 structure points are in the state, the state vector has a dimensionality of at least 126. To represent such a large state vector accurately, a lot of samples will be needed, which is unfavorable for the performance of the particle filter. Especially when the proposal distribution is a lot wider than the posterior distribution, an increased dimension will relate exponentially to the number of samples required to attain the same sample variance [39].

In order to reduce the state dimension, a technique called Rao-Blackwellization [40] will be used. Rao-Blackwellization of a particle filter is based on finding a certain structure in the model and use this structure to do a more efficient state estimation. In our problem of estimating object structure and motion a decomposition of the state space in two parts is possible. We can distinguish the object motion part and the object structure part, represented as $\boldsymbol{x} = [\boldsymbol{x}_m, \boldsymbol{x}_s]$. Conditioned on the object motion, the object structure can be represented by Gaussian uncertainties. The multi-modality of the SfM problem, introduced by ambiguous solutions or independent object motions in the scene can be captured with the representation of object motion as a weighted sample set. The structure in the model is represented with a DBN in Figure 3.9.

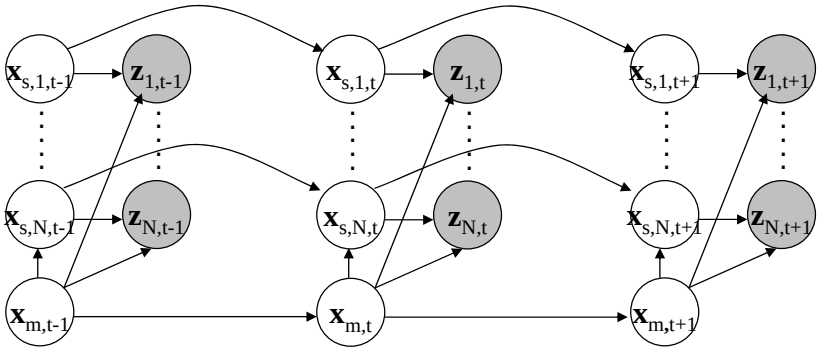


Figure 3.9: DBN for the object motion/structure state-space model

This decomposition into two parts, one conditioned on the other, leads to the following definition of the posterior distribution

$$p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t})p(\boldsymbol{x}_{s,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}) \tag{3.33}$$

with $\boldsymbol{x}_m$ representing object motion and $\boldsymbol{x}_s$ representing object structure. Object motion is represented with a Eucledian transformation (2.4) having the following elements

$$\boldsymbol{x}_m = [T_x, T_y, T_z, \Psi_x, \Psi_y, \Psi_z]^T \tag{3.34}$$

This representation can be extended with velocities as well, but this will increase the dimension of the state space. In the chapter on Experimentation we will find out what effect this has on the performance of the method. Object structure is represented with a set of 3-d points

$$\boldsymbol{x}_s = [\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n]^T \tag{3.35}$$

The state transition (2.9) is linear and can be described by

$$\begin{aligned} \boldsymbol{T}_{t+1} &= \boldsymbol{T}_t + \boldsymbol{\eta}_{\boldsymbol{T}} \\ \boldsymbol{\Psi}_{t+1} &= \boldsymbol{\Psi}_t + \boldsymbol{\eta}_{\boldsymbol{\Psi}} \\ \boldsymbol{P}_{i,t+1} &= \boldsymbol{P}_{i,t} + \boldsymbol{\eta}_{\boldsymbol{P}} \end{aligned} \tag{3.36}$$

where the Gaussian uncertainties $\boldsymbol{\eta}$ should be adjusted to match the correct physical behavior of the real scene. This will be investigated in the section on Filter tuning in the chapter Experimentation.

The measurement equation (2.10) is non-linear and obtained by combining (2.4) and (3.18). For each structure point a measurement equation is defined as

$$\begin{bmatrix} u_i \\ v_i \\ d_i \end{bmatrix} = \frac{f}{\boldsymbol{R}^3\boldsymbol{P}_i + T_z} \begin{bmatrix} \boldsymbol{R}^1\boldsymbol{P}_i + T_x \\ \boldsymbol{R}^2\boldsymbol{P}_i + T_y \\ b \end{bmatrix} \tag{3.37}$$

where $\boldsymbol{R}$ is the rotation matrix (2.5) based on $\boldsymbol{\Psi}$ and $\boldsymbol{R}^i$ denotes its ith row. f and b are the known focal length and baseline of the stereo camera. This is in the general case, suppose now that the rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ are known, which is the case when we condition the object structure on the object motion . The transformed position to camera coordinates can than be used in the measurement equation $\tilde{\boldsymbol{P}}_i = \boldsymbol{R}\boldsymbol{P}_i + \boldsymbol{T}$. The relation than reduces to

$$\begin{bmatrix} u_i \\ v_i \\ d_i \end{bmatrix} = \frac{f}{\tilde{Z}_i} \begin{bmatrix} \tilde{X}_i \\ \tilde{Y}_i \\ b \end{bmatrix} \tag{3.38}$$

which is the measurement equation used in the object structure state estimation.

3.4.2 Object structure and motion estimation

In order to estimate object motion and object structure, the structure points in the state space should be clustered into sets of features belonging to the same object. Since the clustering is based on the implementation of object motion and structure estimation, we will discuss this problem first and in the next section continue with the clustering problem. An overview of the state estimation in our proposed method is displayed in Figure 3.10.

We will start with the situation of a static scene, where all feature points are considered part of the same object. As mentioned before, the state is split up into two parts, an object motion and an object structure part. For both parts the estimation process is different and also the internal representation is different. Object motion is represented in the state vector $\boldsymbol{x}_m$ and the posterior distribution of the object motion represented by a weighted set of N_s samples

$$p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t}) = \sum_{i=1}^{N_s} w_t^{(i)} \delta(\boldsymbol{x}_{m,t} - \boldsymbol{x}_{m,t}^{(i)}) \tag{3.39}$$

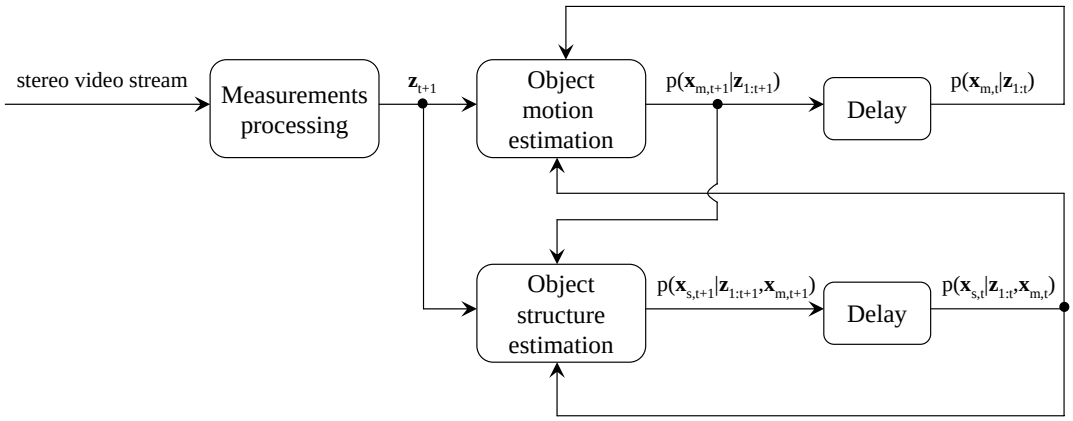


Figure 3.10: Overview of the state estimation in the proposed method

Conditioned on each motion sample, the object structure is represented by a state vector containing a set of N_p points $\boldsymbol{x}_s = [\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{N_p}]^T$. The posterior distribution of object structure, given a certain object motion, will be represented with a Gaussian distribution

$$p(\boldsymbol{x}_{s,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}) \sim \mathcal{N}(\boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t}) \tag{3.40}$$

Because the locations of the 3-d points inside the object (i.e. conditioned on the object motion) are independent, we can write

$$p(\boldsymbol{x}_{s,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}) = \prod_{i=1}^{N_p} p(\boldsymbol{P}_{i,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}) \tag{3.41}$$

where the posterior distribution of each individual point is represented by a Gaussian

$$p(\boldsymbol{P}_{i,t}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t}) \sim \mathcal{N}(\boldsymbol{\mu}_{P_{i,t}}, \boldsymbol{\Sigma}_{P_{i,t}}) \tag{3.42}$$

This is how the object motion and object structure will be represented in the algorithm.

The next problem is the propagation from the posterior state at time t to the next posterior state at time $t+1$. This next posterior state includes information of the dynamic change of the state according to the state transition model and the new observations of the state. In general this relation involves combining the state dynamics with the state observations using Bayes rule, according to (2.15). Since we defined our problem with two posterior distributions, both distributions have to be propagated. For the object motion this results in

$$p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t+1}) = \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1})p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t})}{\int p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1})p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t})d\boldsymbol{x}_{m,t+1}} \tag{3.43}$$

where the likelihood calculation involves solving the following integral

$$p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}) = \int p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}, \boldsymbol{x}_{s,t+1})p(\boldsymbol{x}_{s,t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t})d\boldsymbol{x}_{s,t+1} \tag{3.44}$$

Here the original likelihood function $p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}, \boldsymbol{x}_{s,t+1})$ depends on (3.37) where the measurements are assumed to be Gaussian distributed with covariance $\boldsymbol{\Sigma}_z = diag(\sigma_u^2, \sigma_v^2, \sigma_d^2)$. The prior $p(\boldsymbol{x}_{s,t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t})$ is also Gaussian distributed. However the relation $\boldsymbol{z} = h(\boldsymbol{x}_{m,t+1}, \boldsymbol{x}_{s,t+1})$, given in (3.37) is nonlinear, and because we integrate over $\boldsymbol{x}_{s,t+1}$, an analytical solution to the integral in terms of a resulting Gaussian pdf is not immediately possible. A numerical approach using Monte Carlo integration would be possible to evaluate the integral and another possibility is to linearize the relation (3.37) so that the likelihood function can be evaluated as a Gaussian. To save computational power, we will investigate the linearization of h. For this linearization there are two possibilities, linearization around the measured value and

linearization around the predicted state. Both types will shortly be discussed here. In this discussion we will use the transformed relation between points and measurements according to (3.38). A point will be transformed as

$$\tilde{\boldsymbol{P}}_i = \boldsymbol{R}\boldsymbol{P}_i + \boldsymbol{T} \tag{3.45}$$

and the corresponding covariance matrix of $\boldsymbol{P}_i$ is transformed as well

$$\tilde{\boldsymbol{\Sigma}}_{P_i} = \boldsymbol{R}\boldsymbol{\Sigma}_{P_i}\boldsymbol{R}^T \tag{3.46}$$

Linearization around predicted state This linearization means we approximate the relation (3.37) by a truncated taylor series expansion around the predicted state from $p(\boldsymbol{x}_{s,t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{m,t})$. For each point $\tilde{\boldsymbol{P}}_i$ transformed from the state vector $\boldsymbol{x}_{s,t+1}$ using $\boldsymbol{x}_{m,t}$, we can do the linearization around $\tilde{\boldsymbol{\mu}}_{P_i}$

$$\boldsymbol{z}_i = h(\tilde{\boldsymbol{P}}_i) \approx h(\tilde{\boldsymbol{\mu}}_{P_i}) + \boldsymbol{H}(\tilde{\boldsymbol{P}}_i - \tilde{\boldsymbol{\mu}}_{P_i}) \tag{3.47}$$

where $\boldsymbol{H}$ is the Jacobian matrix. This is the kind of linearization that is used in for example the Extended Kalman Filter for the measurement linearization. By using the approximation, the integral (3.44) can be solved analytically as

$$p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}) \sim \mathcal{N}(h(\tilde{\boldsymbol{\mu}}_{P_i}), \boldsymbol{\Sigma}_z + \boldsymbol{H}\tilde{\boldsymbol{\Sigma}}_{P_i}\boldsymbol{H}^T) \tag{3.48}$$

Linearization around measurements The second possibility is based on an initial transform of the measurements from disparity space to Euclidean 3-d space, by linearizing around the current measurement $\boldsymbol{z}_t$. For the measurement of the ith point from $\boldsymbol{x}_{s,t+1}$ this transform is defined by

$$\tilde{\boldsymbol{z}}_i = g(\boldsymbol{z}_i) \tag{3.49}$$

with g defined as

$$g(\boldsymbol{z}) = \frac{b}{d_i}\begin{bmatrix} u_i \\ v_i \\ f \end{bmatrix} \tag{3.50}$$

By linearizing g around a given measurement $\bar{\boldsymbol{z}}_i$ we get

$$\tilde{\boldsymbol{z}}_i \approx g(\bar{\boldsymbol{z}}_i) + \boldsymbol{G}(\tilde{\boldsymbol{z}}_i - \bar{\boldsymbol{z}}_i) \tag{3.51}$$

where $\boldsymbol{G}$ is the Jacobian matrix. With this measurement transformation, the integral (3.44) can be solved analytically as

$$p(\tilde{\boldsymbol{z}}_{t+1}|\boldsymbol{x}_{m,t+1}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{P_i}, \boldsymbol{G}\boldsymbol{\Sigma}_z\boldsymbol{G}^T + \tilde{\boldsymbol{\Sigma}}_{P_i}) \tag{3.52}$$

One of these linearization techniques can thus be used to evaluate the likelihood function. A straightforward particle filter implementation is now to use the prior distribution $p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t})$ as proposal distribution. Assuming that the previous posterior distribution was available in sampled form $p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t}) \approx \sum_{i=1}^{N_s} w_t^{(i)}\delta(\boldsymbol{x}_{m,t} - \boldsymbol{x}_{m,t}^{(i)})$, it is easy to sample from this proposal distribution

$$\begin{aligned} p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t}) &= \int p(\boldsymbol{x}_{m,t+1}|\boldsymbol{x}_{m,t})p(\boldsymbol{x}_{m,t}|\boldsymbol{z}_{1:t})d\boldsymbol{x}_{m,t} \\ &\approx \int p(\boldsymbol{x}_{m,t+1}|\boldsymbol{x}_{m,t}) \textstyle\sum_{i=1}^{N_s} w_t^{(i)}\delta(\boldsymbol{x}_{m,t} - \boldsymbol{x}_{m,t}^{(i)})d\boldsymbol{x}_{m,t} \\ &= \textstyle\sum_{i=1}^{N_s} w_t^{(i)} p(\boldsymbol{x}_{m,t+1}|\boldsymbol{x}_{m,t}^{(i)}) \end{aligned} \tag{3.53}$$

In our case this means sampling from a weighted mixture of Gaussians, since the transition density is assumed to be Gaussian (3.36). This leads to a representation of the prior density with a weighted sample set

$$p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t}) \approx \frac{1}{N_s}\sum_{i=1}^{N_s}\delta(\boldsymbol{x}_{m,t+1} - \boldsymbol{x}_{m,t+1}^{(i)}) \tag{3.54}$$

where $\boldsymbol{x}_{m,t+1}^{(i)}$ are drawn from the proposal distribution in (3.53). The posterior object motion distribution can be obtained now by substituting (3.54) into the posterior equation (3.43) resulting in

$$\begin{aligned} p(\boldsymbol{x}_{m,t+1}|\boldsymbol{z}_{1:t+1}) &\approx \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1})\frac{1}{N_s}\sum_{i=1}^{N_s}\delta(\boldsymbol{x}_{m,t+1}-\boldsymbol{x}_{m,t+1}^{(i)})}{\int p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1})\frac{1}{N_s}\sum_{i=1}^{N_s}\delta(\boldsymbol{x}_{m,t+1}-\boldsymbol{x}_{m,t+1}^{(i)})d\boldsymbol{x}_{m,t+1}} \\ &= \frac{\sum_{i=1}^{N_s} p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}^{(i)})\delta(\boldsymbol{x}_{m,t+1}-\boldsymbol{x}_{m,t+1}^{(i)})}{\sum_{i=1}^{N_s} p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}^{(i)})} \\ &= \sum_{i=1}^{N_s} w_{t+1}^{(i)}\delta(\boldsymbol{x}_{m,t+1}-\boldsymbol{x}_{m,t+1}^{(i)}) \end{aligned} \tag{3.55}$$

where each weight

$$w_{t+1}^{(i)} = \frac{p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}^{(i)})}{\sum_{j=1}^{N_s} p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{m,t+1}^{(j)})} \tag{3.56}$$

is calculated by using the likelihood linearization as described above.

The following step is the estimation of the posterior object structure distribution. As mentioned before, object structure will be represented with a set of points, each with its own associated Gaussian and conditioned on a given motion sample. Thus for each individual point and for each given sample, a posterior distribution has to be calculated. For a point j and a motion sample $\boldsymbol{x}_{m,t}^{(i)}$, the posterior distribution is given by

$$p(\boldsymbol{P}_{j,t}|\boldsymbol{z}_{j,1:t},\boldsymbol{x}_{m,t}^{(i)}) \sim \mathcal{N}(\boldsymbol{\mu}_{P,j,t}^{(i)},\boldsymbol{\Sigma}_{P,j,t}^{(i)}) \tag{3.57}$$

The propagation to the next posterior distribution $p(\boldsymbol{P}_{j,t+1}|\boldsymbol{z}_{j,1:t+1},\boldsymbol{x}_{m,t+1}^{(i)})$, by combining the state dynamics and state observations according to Bayes rule will be done with a Kalman Filter. The two alternatives for the representation of measurements, as shown before in the two linearization approaches for calculating the integral (3.44), lead to different implementations of this filter. We will show what these differences are in computation of the filter. To simplify things, the current state will be transformed according to the motion defined in $\boldsymbol{x}_{m,t+1}^{(i)}$ as given by (3.45) and (3.46), then the Kalman Filter will be executed and finally the estimated state and covariance will be transformed back using the inverse transforms of (3.45) and (3.46).

Linearization around predicted state In this approach, the real measurements in disparity space are used and an Extended Kalman Filter can handle the state estimation. The linearization of the measurement function is exactly implemented as given by (3.47). In order to calculate the posterior, first the prior is calculated using the state transition defined in (3.36) and the definition of the Kalman Filter in (2.24) as

$$p(\boldsymbol{P}_{j,t+1}|\boldsymbol{z}_{j,1:t},\boldsymbol{x}_{m,t}^{(i)}) \sim \mathcal{N}(\boldsymbol{\mu}_{P,j,t}^{(i)},\boldsymbol{\Sigma}_{P,j,t}^{(i)}+\boldsymbol{Q}) = \mathcal{N}(\boldsymbol{\mu}_{P,j,t+1}^{-(i)},\boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}) \tag{3.58}$$

where $\boldsymbol{Q}$ is the dynamic noise in the state transition, due to rigidity in the object structure. To upgrade it to a posterior estimate, the measurements are included. First the Kalman gain is computed

$$\boldsymbol{K} = \boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}\boldsymbol{H}^T + \boldsymbol{\Sigma}_{z,j,t+1}^{(i)})^{-1} \tag{3.59}$$

and then using (2.27) the posterior distribution is calculated by

$$\begin{aligned} p(\boldsymbol{P}_{j,t+1}|\boldsymbol{z}_{j,1:t+1},\boldsymbol{x}_{m,t+1}^{(i)}) &\sim \mathcal{N}(\boldsymbol{\mu}_{P,j,t+1}^{-(i)} + \boldsymbol{K}(\boldsymbol{z}_{j,t+1} - h(\boldsymbol{\mu}_{P,j,t+1}^{-(i)})), (\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}) \\ &= \mathcal{N}(\boldsymbol{\mu}_{P,j,t+1}^{(i)},\boldsymbol{\Sigma}_{P,j,t+1}^{(i)}) \end{aligned} \tag{3.60}$$

Linearization around measurements In this approach all measurements are transformed from disparity space to Euclidean space using (3.49). The resulting measurements $\tilde{\boldsymbol{z}}_{j,t+1}$ and measurement covariance $\tilde{\boldsymbol{\Sigma}}_{z,j,t+1}$ will be used as measurements in the Kalman Filter. The prior update is unchanged

$$p(\boldsymbol{P}_{j,t+1}|\boldsymbol{z}_{j,1:t},\boldsymbol{x}_{m,t}^{(i)}) \sim \mathcal{N}(\boldsymbol{\mu}_{P,j,t}^{(i)},\boldsymbol{\Sigma}_{P,j,t}^{(i)}+\boldsymbol{Q}) = \mathcal{N}(\boldsymbol{\mu}_{P,j,t+1}^{-(i)},\boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}) \tag{3.61}$$

But the measurement incorporation has changed, the Kalman gain is in this case defined by

$$\boldsymbol{K} = \boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}(\boldsymbol{\Sigma}_{P,j,t+1}^{-(i)} + \tilde{\boldsymbol{\Sigma}}_{z,j,t+1}^{(i)})^{-1} \tag{3.62}$$

so that the posterior is in this case reduces to

$$\begin{aligned} p(\boldsymbol{P}_{j,t+1}|\boldsymbol{z}_{j,1:t+1}, \boldsymbol{x}_{m,t+1}^{(i)}) &\sim \mathcal{N}(\boldsymbol{\mu}_{P,j,t+1}^{-(i)} + \boldsymbol{K}(\tilde{\boldsymbol{z}}_{j,t+1} - \boldsymbol{\mu}_{P,j,t+1}^{-(i)}), (\boldsymbol{I} - \boldsymbol{K})\boldsymbol{\Sigma}_{P,j,t+1}^{-(i)}) \\ &= \mathcal{N}(\boldsymbol{\mu}_{P,j,t+1}^{(i)}, \boldsymbol{\Sigma}_{P,j,t+1}^{(i)}) \end{aligned} \tag{3.63}$$

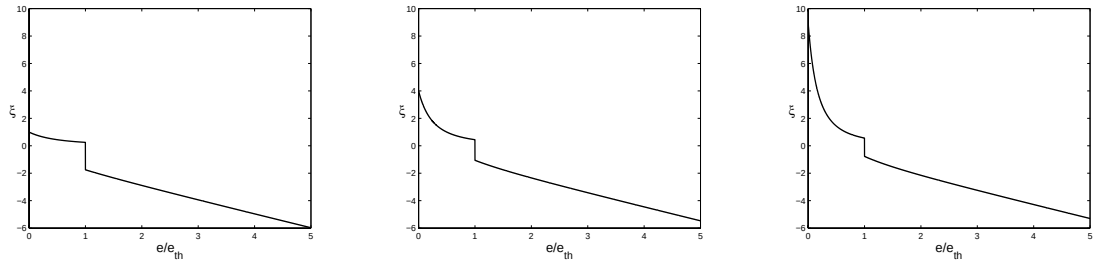
At the end of the next section on Object clustering, we will discuss how the above described object structure and motion estimation will be used in combination with the generated clusters.

3.4.3 Object clustering

Before we can estimate object motion and structure, the points in the state-space should be clustered according to their rigid motion. We will use the assumption of a scene containing only rigidly moving objects. However each object can have its own distinct rigid motion. The task to do now is the following: given a set of feature points, segment this set into new sets of feature points, so that each set has its own distinctive rigid object motion.

A single point has an unlimited scala of possible object motions, which all relate the 3-d location of the point in the object frame to the 3-d location in the camera frame. The trajectory of the point over time and the fact that objects bear inertia can be used to restrict the current object motion somehow. But still, given only one point, a lot of object motions are possible. A cluster needs to be generated by finding regions of high likelihood in the object motion space, valid for a multiple (at least equal to a minimum cluster size) number of points. Clustering on just the current motion likelihood might be difficult since the uncertainty in the measured object position is usually bigger than the expected motion difference between subsequent frames. Therefore, an approach has to be chosen where clustering is based on the joint likelihood over a certain time span.

In the method of Qian, this clustering was based on the validity vector of each motion sample, where the elements of the vector are validity values. A validity value for a certain feature can be seen as some sort of joint measurement likelihood for that feature over a recent time window using an exponential envelope. For a feature j in motion sample i, the validity value was updated according to (2.51). Which is mainly influenced by the function $\xi(e)$ (2.52) which depends on the distance to the epipolar line. The origin of the $\xi(e)$ function is not explained in the accompanying paper, but it looks a bit arbitrary since its shape depends on the chosen threshold e_{th}, see Figure 3.11.



(a) $e_{th} = 1$ (b) $e_{th} = 2$ (c) $e_{th} = 3$

Figure 3.11: Validity function for several values of e_{th}

We propose to use a different approach, but based on the same principle. Instead of a validity vector, we propose to use a membership vector $\boldsymbol{m} = [m_1, \ldots, m_{N_p}]^T$, where each entry describes the possible membership of that feature, indicated with a probability ranging from 0 to 1. Thus each entry describes the probability

$$m_j^{(i)} = P(M_j^{(i)}) \tag{3.64}$$

where $M_j^{(i)}$ is defined as: feature j is compatible with the motion $\boldsymbol{x}_m^{(i)}$. For each incoming measurement $\boldsymbol{z}_{j,t}$ the feature can be classified as being in group (i.e. consistent with the measurement) or out group. This classification can be based on thresholding the Mahalanobis distance between the measurement $\boldsymbol{z}_{j,t}$ and the projected measurement $h(\boldsymbol{x}_{j,t}^{(i)}, \boldsymbol{P}_{j,t})$. When this distance is smaller than a predefined threshold, the feature is supposed to be compatible with the object motion and when the distance is larger, the feature is not compatible with the object motion. Suppose the classification function ξ is defined as

$$\xi(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z}) = d_m(\boldsymbol{z}, h(\boldsymbol{x}, \boldsymbol{P})) < d_{th} \tag{3.65}$$

By accumulating the classification results over time, a membership probability can be obtained as

$$P(M_j^{(i)}) = \frac{1}{t} \sum_{\tau=1}^{t} \xi(\boldsymbol{x}_{m,\tau}^{(i)}, \boldsymbol{P}_{j,\tau}, \boldsymbol{z}_{j,\tau}) \tag{3.66}$$

A drawback of this function, is that it adapts very slowly to changes in the object structure. A feature point will only yield a probability of 0.5 or higher when it has been classified as in group for more than half the time. To account for the object splitting and merging in a more adaptive manner, an exponential forgetting window will be used to update the membership values. For each incoming measurement, the membership value will be updated by

$$m_{j,t+1}^{(i)} = (1 - \alpha) m_{j,t}^{(i)} + \alpha \xi(\boldsymbol{x}_{j,t+1}^{(i)}, \boldsymbol{P}_{j,t+1}, \boldsymbol{z}_{j,t+1}) \tag{3.67}$$

here α is our exponential forgetting factor. This way we can tune the period of adaptation to new object structure by varying the factor α.

After having defined the membership vector, motion samples have to be clustered into distinct objects. When enough samples with high weights are available, the different objects will become visible as different groups of motion samples, each group sharing more or less the same kind of membership vector. When the motions are distinctive enough, motion samples compatible with object motion A will have a completely different membership vector than motion samples being compatible with object motion B. Clustering can than be performed as proposed by Qian, based on sorting the motion sample weights, generate cluster centers and find compatible motion samples having a similar membership vector. A cluster is valid when it has at least P_m membership vector elements with a value of 0.5 or higher. This approach is based on the assumption that after a while the clusters will show up, because their internal structure can not be explained by a single rigid motion anymore.

A drawback of this kind of clustering is that not all information available in the motion samples is fully exploited here in the clustering task. The cluster centers are generated based on the motion samples having the highest weights, so that no information about the membership vectors of the rest of the samples is used. Therefore we propose a new clustering mechanism, where cluster centers are not generated from a single motion sample, but by using statistics of all motion samples together. The approach is based on finding the highest variation in the set of membership vectors, which can be used to hierarchical cluster the motion samples into groups with similar membership vectors. To amplify the differences between individual membership vectors, we will use a clipped version of the membership vector, where for each entry in the vector a value of 0 or 1 is possible. The clipping of a membership vector $\boldsymbol{m}^{(i)}$ from motion sample $\boldsymbol{x}_m^{(i)}$, results in the clipped membership vector, by thresholding the individual elements j according to

$$\tilde{m}_j^{(i)} = m_j^{(i)} > 0.5 \tag{3.68}$$

In the beginning some motion samples might contain a membership vector with only elements lower than 0.5. To get rid of these, we drop those clipped membership vectors that have less than P_m (being the smallest possible cluster size) elements set to one. We will thus end up with a set S containing the indices of the positively selected membership vectors, which resemble potentially moving objects. By finding the direction of highest variance between all membership vectors in the set, we will split the set into two, using that direction of highest variance. This idea is based on the Principal Direction Divisive Partitioning algorithm, introduced by Boley in [41]. Finding the direction of highest variance will be done by calculating the sample covariance matrix of all membership vectors in S.

$$\boldsymbol{A}_m = \frac{1}{N_s} \sum_{i \in S} (\tilde{\boldsymbol{m}}^{(i)} - \boldsymbol{\mu}_m)^T (\tilde{\boldsymbol{m}}^{(i)} - \boldsymbol{\mu}_m) \tag{3.69}$$

Here $\boldsymbol{\mu}_m$ is the sample mean of the membership vectors. The largest eigenvector of this matrix indicates the direction with highest variance. That eigenvector will be used to split the set into two subsets, where both subsets are related to different rigid motions. The indices of elements in the eigenvector having a positive value will be grouped together in a left cluster selection set C_{left} and the indices of elements containing a negative value will be grouped together in a right cluster selection set C_{right}. The original set of membership vectors will now be redistributed into a left membership vector S_{left} set and a right membership vector set S_{right}. A membership vector will be assigned to the left set when it has enough overlap with the left cluster selection set C_{left}, according to the minimum cluster size P_m. This can be evaluated with the following expression

$$(\tilde{\boldsymbol{m}}_j^{(i)})^T \boldsymbol{c}_{right} \geq P_m \tag{3.70}$$

where $\boldsymbol{c}_{right}$ is a vector containing ones for all elements having an index that is part of the cluster selection set C_{left} and zeros for all other elements. For the right set, a similar test will be used. These two subsets can be input to the same splitting algorithm again, when more rigid motions are present in a subset. In order to feed the subset to the splitting algorithm again, we will first remove those elements from the membership vector that are not in the cluster selection set, and continue with the reduced membership vectors. That way only the unclustered membership elements will be used in the next splitting phase. A question that now remains unanswered is when do we have to stop splitting? Or in other words, when does the remaining set contain only those elements that exactly belong to one rigid motion. A simple test is to see whether the number of elements is lower than the minimum allowed cluster size P_m. If this is the case, the set cannot be split up again. But to see if a set that contains much more than P_m elements, are all member of the same rigid motion is more difficult. A possibility would be to look at the size of the largest eigenvalue. If this would drop to a low number, the variance in the set has dropped so that it is possibly a set containing only one rigid motion. In the experimentation chapter, we will investigate this by watching the value of the largest eigenvalue and see if that can be used to judge whether a set contains only one rigid motion.

In the section on Object structure and motion estimation, we assumed a static scene were all points and measurements will be used in the state estimation. But in order to represent the object motion and structure of multiple independently moving objects, measurements have to be weighted differently for different object motions. The output of the clustering algorithm can help us with the state estimation, by weighting samples differently for different clusters, based on the points actually represented by each cluster. In order to finally use one set of weighted samples, the individual weights of all clusters are combined and weighted equivalently, so that the combined set represents multiple motions where each motion (independently moving object) is represented with an equal share of motion samples. This is based on the balancing idea that has been introduced in the method of Qian [1] and it is used to prevent the individual clusters from competing for the samples. We want to use an equal number of samples for all independently moving objects. To incorporate this idea in our Rao-Blackwellized particle filter, we have to calculate weights according to the generated clusters. Suppose the clustering algorithm found K clusters, each representing a set of points S_k. Then for each cluster and motion sample i, the cluster weights will be calculated according to

$$\tilde{w}^{(i),k} = \prod_{j \in S_k} p(\boldsymbol{z}_j | \boldsymbol{x}_m^{(i)}) \tag{3.71}$$

and normalized to produce a sum of one

$$w^{(i),k} = \frac{\tilde{w}^{(i),k}}{\sum_{i=1}^{N_s} \tilde{w}^{(i),k}} \tag{3.72}$$

The individual likelihood functions are evaluated according to (3.48) or (3.52) depending on the chosen linearization. The final combined weight set is then calculated by averaging the individual weights from each cluster

$$w^{(i)} = \frac{1}{K} \sum_{k=1}^{K} w^{(i),k} \tag{3.73}$$

for all motion samples i. The resulting posterior object motion distribution is now given by the weighted sample set $\{w^{(i)}, \boldsymbol{x}_m^{(i)}\}_{i=1}^{N_s}$.

The object structure update will not be influenced by the fact that points are clustered. For each motion sample, all points will be updated with the Kalman filter as described above. By treating all points equally in the object structure estimation, it is possible for points in different objects to merge together after a while of undergoing the same object motion.

3.5 Summary

Since our method is mainly based on the method of Qian, it is interesting to see what actually the differences and similarities between the two methods are. We will start this section with a comparison of the proposed method and Qians method.

At the end of this section we will give an overview of the whole method in pseudo code. The algorithm is split up into two blocks, an input block that pre-processes the raw camera images into useful measurements and a state estimation block in which the measurements are processed and used in combination with the state dynamics to update our posterior state estimate.

3.5.1 Comparison with Qians method

At the highest level, the two methods are based on the same principle. They both recursively segment and estimate object motion and object structure of possibly multiple independently moving objects. This recursive estimation is in both methods handled by some kind of particle filtering based technique. So the global idea of both methods is the same, measurements and state dynamics are combined to recursively estimate a posterior state.

A first difference between the two methods can be found when looking at what kind of measurements are used. Qians method uses measurements obtained from a monocular camera, while our method uses measurements obtained from a stereo camera. In both methods, these raw image measurements are first processed, resulting in a selection of new measurements for certain 3-d points in the scene which are then used as input for the rest of the method. In Qians method for each selected (and tracked) point, the measurement consists of the image coordinates of a projection of this 3-d point on the image plane. In our method each measurement is, in addition to this coordinate, extended with a disparity value. The disparity value is inversely related to the distance along the Z-axis of that 3-d point to the camera. The impact of using measurements with a disparity value is that instead of only relative distances of the object structure and motion, absolute distances can be estimated. A necessary condition for the recovery of absolute distances however, is that a calibrated stereo camera should be used.

In the represented state space of both methods there are differences as well. Object motion is more or less represented in a similar way, translations and rotations are used, but Qian extends these by incorporating motion velocities as well. For the object structure representation, the difference is that Qian only estimates the depth of a tracked point, while in our method the full 3-d position is estimated. Qian uses externally (outside of the state space) stored information, being the measurement of a feature in the first image, to upgrade this depth value to a complete 3-d position. This limits the reconstructed scene in a way that only the scene observed in the first measurement will be reconstructed. Besides these representational differences in the state space, the estimation of the state is also different. Because the dimension of the complete state space is quite high, both methods use some sort of marginalization technique to reduce this dimension. Qian divides the estimation process in three subprocesses. First object motion with only translation direction is estimated. Then conditioned on the object motion with translation direction, translation magnitude will be estimated. Finally, conditioned this complete object motion, the depth of all tracked points will be estimated. In our method, a similar division takes place, first object motion is estimated. Then, conditioned on the estimated object motion, the object structure will be estimated. However, the implementation of these estimation processes is quite different in both methods. Qian uses particle filters for all three subprocesses, while we use a particle filter only for the object motion, and represent the conditioned object structure efficiently with Kalman filters. The last difference between the state estimation in Qian and our method, is in the way the marginalized states are updated. Qian uses the externally stored first image frame measurements combined with epipolar geometry in the current image to update object motion in the next state, while in our method we integrate over the estimated object structure from the previous state to update object motion. The key difference between these two approaches is that Qian never uses the incremented knowledge on object structure to

update his object motion, while in our method we always use as much knowledge as available on object structure in the object motion update step.

The motion segmentation is also implemented differently in both methods. Although the basic idea is the same: for a given motion sample, quantify for each tracked point in the state space the probability that the motion defined by the given motion sample is compatible with the observations of the tracked point over a certain time window. For each given motion sample this will result in a vector with the length equal to the number of tracked points, where each element indicates a measure of that probability. In Qians method, this vector is called a validity vector and a high value (above zero) means very likely and a low value (below zero) means very unlikely. Our method uses a similar vector, which we call the membership vector. The elements indicate a probability measure defined from 0 to 1, a value higher than 0.5 means the tracked point has a higher chance of being compatible than incompatible with the given motion and vice versa. The way that both vectors are updated is also a bit different.

To segment the motion samples into different independent motions, the motion samples should be clustered so that motion samples having a similar membership/validity vector will be grouped together. The implementation of this clustering is different for Qians method and our method. Qian uses motion samples with high weights to define cluster centers, while we use statistics of all motion samples together to find discriminative points to generate cluster centers.

3.5.2 Algorithm in pseudo code

The algorithm will be presented with first a section on initialization, in the second section we describe how the algorithm is running.

Initialization

1. Calculate rectification transforms for the left and right camera, $\boldsymbol{H}_l$ and $\boldsymbol{H}_r$, using the algorithm described in section 3.2.3.
2. Rectify the left and right image by applying transformations $\boldsymbol{H}_l$ and $\boldsymbol{H}_r$ to both images.
3. Select initial corners from the left image, with the initialization procedure explained in section 3.3.4.
4. Verify the selected corners by finding correspondences in the right image. When for a selected corner no reliable correspondence can be found, try finding a new corner.
5. Use the initial list with selected corners to initialize the state space.
   (a) Create a list of N_s motion samples, with all translation and rotation parameters set to zero.
   (b) For each of the motion samples, set all elements in the membership vector to 0.5.
   (c) For each motion samples i, initialize the object structure representation. Each structure point j is represented with a separate Gaussian having mean $\boldsymbol{\mu}_{P,j,1}^{(i)}$ and covariance $\boldsymbol{\Sigma}_{j,1}^{(i)}$. The measurement (of the corner) $\boldsymbol{z}_j = [u_j, v_j, d_j]^T$ is used to initialize the mean and covariance, by using equation (3.32).

Running the algorithm

1. Obtain next measurements
   (a) Use the KLT tracker, described in section 3.3.4, to track each of the features (corners) to the next left image frame. When no correspondence can be found anymore, remove the feature from the list and remove it from the state.
   (b) Use the stereo matching algorithm, also described in section 3.3.4, to find a stereo match for each of the tracked features in the right frame. When no match can be found, remove the feature from the list and remove it from the state.
   (c) For all of the removed features, find new ones with the corner detector and add the to the state space. Initialize their membership values to 0.5 and initialize their mean and covariances in the state space, according to equation (3.32).

2. Draw new motion samples

   (a) For each of the motion samples, draw a new translation and rotation, according to the state transition model given in (3.36). Because the effective offspring (due to the low effective sample size) of each motion sample is quite low, we will boost the samples from the prior distribution. This means, for each original motion sample, we draw κ new samples and finally after the resampling step, only N_s samples will remain.

   (b) For each drawn motion sample i and each state space point j, calculate the Mahalanobis distance $d^{(i)}_{m,j}$ to the new observation of that point. To evaluate this distance, the measurement and the state space point should be aligned first in the same reference frame. This means the state space point has to be transformed according to the transformation defined in the drawn motion sample.

   (c) Update the membership vectors of all motion samples based on the calculated Mahalanobis distances. For a given sample i, the jth element of the membership vector will be updated according to

   $$m^{(i)}_{j,t+1} = (1-\alpha)m^{(i)}_{j,t} + \alpha(d^{(i)}_{m,j} < d_{th})$$

   where α is a learning constant whose value can be set, depending on the scene and camera frame rate.

3. Generate motion clusters

   (a) Remove all motion samples that have less than P_m (being the minimum cluster size) membership elements with a value of 0.5 or higher. The remaining motion samples will be clipped according to (3.68) and together form the set S.

   (b) Calculate the sample covariance matrix $\boldsymbol{A}_m$ of the clipped membership values according to

   $$\boldsymbol{A}_m = \frac{1}{N_s}\sum_{i\in S}(\tilde{\boldsymbol{m}}^{(i)} - \boldsymbol{\mu}_m)^T(\tilde{\boldsymbol{m}}^{(i)} - \boldsymbol{\mu}_m)$$

   where $\boldsymbol{\mu}_m$ is the sample mean.

   (c) Calculate the largest eigenvector λ_1 and the corresponding eigenvector $\boldsymbol{v}_1$ from the sample covariance matrix $\boldsymbol{A}_m$.

   (d) When the largest eigenvector is large enough, $\lambda_1 > \lambda_{th}$, the set contains more than one rigid motion, and can be split into two.

      i. Create a left cluster selection set, according to the positive entries in the largest eigenvector $\boldsymbol{v}_1$.
      ii. Assign all motion samples with at least P_m positive membership values in common with the left cluster selection set to the left cluster set S_l. Remove the elements from the membership vector which are not contained in the left cluster selection set.
      iii. Create a right cluster selection set, according to the negative entries in the largest eigenvector $\boldsymbol{v}_1$.
      iv. Assign all motion samples with at least P_m positive membership values in common with the right cluster selection set to the left cluster set S_r. Remove the elements from the membership vector which are not contained in the right cluster selection set.
      v. Try to split the two sets S_l and S_r again, by recursing again to 3b for both sets.

   (e) The final subsets (remaining end nodes) are the clusters for the next step of the algorithm, they will be numbered 1 to K, where S_k denotes the kth cluster.

4. Weight calculation and cluster balancing

   (a) For each individual cluster, weights will be calculated according to the strongest P_m elements in the cluster.

(b) For a given motion sample i and cluster k, calculate the weight $w_t^{(i),k}$ according to the product of the P_m highest likelihoods

$$w_t^{(i),k} = \prod_{j \in \boldsymbol{C}^{(i),k}} p(\boldsymbol{z}_j | \boldsymbol{x}_m^{(i)})$$

where $\boldsymbol{C}^{(i),k}$ are the indices of those P_m points having the highest likelihood for motion sample i using points in cluster k.

(c) Normalize the sum of all weights for each cluster k to equal $\frac{1}{K}$

(d) The final balanced set of weighed samples is obtained by summing the weights of the individual clusters

$$w_t^{(i)} = \sum_{k=1}^{K} w_t^{(i),k}$$

5. Resampling and object structure update

(a) Reduce the set of samples to a size of N_s again by resampling the set according to the calculated weights.

(b) For each of the resampled samples, update the object structure by running the Kalman filter. For a given motion sample i and given point j, the Kalman filter is used to update the mean $\boldsymbol{\mu}_{P,j,t}^{(i)}$ and covariance matrix $\boldsymbol{\Sigma}_{j,t}^{(i)}$ according to the Kalman filter equations defined in section 3.4.2.

4

Experimentation

4.1 Introduction

A set of experiments will be performed in order to evaluate the method under varying conditions. To accurately evaluate the performance of the method and how the performance depends on environmental conditions, synthetic scenes were used in the experiments. These synthetic scenes were "captured" with a synthetic camera, which has the same specifications as the real camera that will be used in some of the experiments. To simulate this camera as good as possible, a model of the camera with parameters obtained by calibration was used.

4.1.1 Camera hardware

The stereo camera that we will simulate in our experiments is the STH-MDCS2 from Videre Design. To process the information from this stereo camera, we have to fill in the fixed parameters in our camera model, or in other words, calibrate the camera. These include the intrinsic parameters like focal length and image plane origin and the extrinsic parameters that describe the relative orientation of the two cameras to each other. Besides that, there can be some lens distortion especially with wide angle lenses.

We have a rough idea of what the individual parameters should be according to the manufacturer, but due to small variations in the fabrication process the actual values of the parameters can deviate from the given parameters. To verify the given parameters we will use a calibration algorithm on the camera. In this case we will calibrate the camera with the equipped calibration program. This program calculates optimal parameters by solving a nonlinear set of equations, which is determined by at least 5 views of a calibration target observed by both cameras. The calibration target is a checkerboard with known dimensions. Most of the information about the unknown parameters is revealed when viewing the calibration target at different angles, varying the distance from the calibration target to the camera will give no extra information about these parameters.

For our application it is important that the camera's parameters do not change during operation. When the epipolar geometry is calculated with mismatched parameters, the resulting measurements with the camera will be biased. Therefore we would like to now how stable the camera's parameters are and also what the influence is of remounting lenses on the camera. To investigate this the camera was calibrated in two sets, where in between the two sets the lenses were unmounted and mounted again. In each set the calibration process was run ten times to see how consistent and accurate these parameters can be calculated. The dimensions of the camera according to the manufacturer are given in Table 4.1. The results of running this procedure for both sets are shown in Table 4.2. The tables show the average (μ) and standard deviation (σ) of the ten calibrations.

The left and right intrinsic parameters show the internal geometry of the camera. Here f_x and f_y are the focal length for the X and Y axes. A difference between these two means that the camera's image sensor has non-square pixels. The calibration results show that the focal length is estimated at approximately 5.8 mm while the manufacturer rated the lens as 4.8 mm. This difference can be due to the fact that the size of a pixel on the image sensor does not match its specification.

	value
f (mm)	4.8
u_0	160
v_0	120

(a) Intrinsic parameters

	value
T_x (mm)	-90
T_y (mm)	0
T_z (mm)	0
Ψ_x (rad)	0
Ψ_y (rad)	0
Ψ_z (rad)	0

(b) Extrinsic parameters

Table 4.1: Specifications according to manufacturer

	μ	σ
f_x (mm)	5.82	0.01
f_y (mm)	5.79	0.01
u_0	155.53	0.49
v_0	110.12	0.68
κ_1	-0.23	0.01
κ_2	0.13	0.01

(a) Left int. parameters, first

	μ	σ
f_x (mm)	5.82	0.02
f_y (mm)	5.80	0.02
u_0	159.09	0.94
v_0	116.21	0.78
κ_1	-0.21	0.00
κ_2	0.11	0.01

(b) Right int. parameters, first

	μ	σ
T_x (mm)	-89.04	0.31
T_y (mm)	-0.13	0.29
T_z (mm)	1.24	0.65
Ψ_x (rad)	0.000	0.002
Ψ_y (rad)	0.015	0.003
Ψ_z (rad)	0.000	0.000

(c) Ext. parameters, first

	μ	σ
f_x (mm)	5.82	0.02
f_y (mm)	5.78	0.02
u_0	156.21	0.72
v_0	109.07	1.79
κ_1	-0.22	0.01
κ_2	0.11	0.01

(d) Left int. parameters, second

	μ	σ
f_x (mm)	5.81	0.02
f_y (mm)	5.79	0.02
u_0	160.91	1.54
v_0	113.21	1.59
κ_1	-0.22	0.01
κ_2	0.11	0.01

(e) Right int. parameters, second

	μ	σ
T_x (mm)	-89.04	0.15
T_y (mm)	-0.16	0.31
T_z (mm)	0.33	0.82
Ψ_x (rad)	-0.004	0.004
Ψ_y (rad)	0.010	0.006
Ψ_z (rad)	-0.002	0.001

(f) Ext. parameters, second

Table 4.2: Calibration results

The center of the image plane is given by u_0 and v_0. In the ideal case the image sensor is positioned so that the ray perpendicular to the image plane and through the center of the image plane will hit the sensor in its center. In our case both image sensors are slightly off centered. We also see that remounting the lens (compare first and second set), changes this parameter. This change is probably due to a slightly different placement of the lens.

The calibration procedure also tries to estimate radial lens distortion, according to the following distortion model

$$u_d = u_u(1 + \kappa_1 r^2 + \kappa_2 r^4) \tag{4.1}$$

$$v_d = v_u(1 + \kappa_1 r^2 + \kappa_2 r^4) \tag{4.2}$$

where (u_d, v_d) denotes the distorted projection of the undistorted coordinate (u_u, v_u). The κ values are calculated for normalized coordinates.

The extrinsic parameters, which describe the transformation from the left camera reference frame to the right camera reference frame, are close to the expected values.

4.1.2 Synthetic scenes

A synthetic scene should be a reasonable approximation to a real life scene. Real life scenes can very well be approximated by a static background and (possibly) multiple independently moving objects. An observer moving through this scene would make the static background behave as another moving object, but with the difference that the size of this object is not limited to a dense subspace. Our synthetic scene

simulator will generate a background with some objects having a predefined position and size. To each object a certain motion has to be assigned, which it will execute during the simulation run.

4.1.3 Measure of performance

In order to quantify the performance of the method, we have to identify measures which indicate how well the method performs. Our method can be roughly divided into two parts. A clustering part that clusters feature points into separate clusters based on the motion of these feature points and a part that estimates the motion of each of the clusters/objects.

The performance of the object motion estimation can be quantified by comparing the estimated object motion with the ground truth of the object motion. Under controlled conditions, for example in a synthetic scene, this ground truth data is available. The mean square error of the estimated states can be calculated in this case and is a good measure for the performance of the estimator.

Suppose that the estimated state at time t is represented by $\hat{\boldsymbol{x}}_t$ and the true value of the state at that time by $\tilde{\boldsymbol{x}}_t$. The estimation error is then given by

$$\boldsymbol{e}_t = \hat{\boldsymbol{x}}_t - \tilde{\boldsymbol{x}}_t \tag{4.3}$$

The mean square error can be calculated as a matrix, identifying the errors for each of the individual parameters of the state. We are however not interested in the error for each separate item in the state vector, but more in the overall mean square estimation error. In order to quantify the mean square error as a single value we will calculate the expectation of the weighted sum of squared errors

$$m_t = \mathbb{E}[\boldsymbol{e}_t^T \boldsymbol{Q} \boldsymbol{e}_t] \tag{4.4}$$

where $\boldsymbol{Q}$ is a diagonal matrix which weights the influence of the individual parameters. In our experimentation we set the weighting matrix to the identity matrix, since the unities of the state variables were comparable. The estimated state $\hat{\boldsymbol{x}}_t$ can be represented by a probability density function $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$, so that the mean square error can be calculated as

$$m_t = \int (\boldsymbol{x}_t - \tilde{\boldsymbol{x}}_t)^T (\boldsymbol{x}_t - \tilde{\boldsymbol{x}}_t) p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) d\boldsymbol{x}_t \tag{4.5}$$

In most of our experiments, the state estimation is based on a Particle Filter. Evaluation of the mean square error in the case of the particle filter is straightforward

$$m_t = \sum_i (\boldsymbol{x}_t^{(i)} - \tilde{\boldsymbol{x}}_t)^T (\boldsymbol{x}_t^{(i)} - \tilde{\boldsymbol{x}}_t) w_t^{(i)} \tag{4.6}$$

There are also some experiments in which we use a Kalman Filter as state estimator. In case of the Kalman Filter we have $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}_t}, \boldsymbol{\Sigma}_{\boldsymbol{x}_t})$. The mean square error can then be calculated as

$$m_t = (\boldsymbol{\mu}_{\boldsymbol{x}_t} - \tilde{\boldsymbol{x}}_{\boldsymbol{t}})^T (\boldsymbol{\mu}_{\boldsymbol{x}_t} - \tilde{\boldsymbol{x}}_{\boldsymbol{t}}) + tr(\boldsymbol{\Sigma}_{\boldsymbol{x}_t}) \tag{4.7}$$

where tr stands for the trace of the matrix.

4.2 Filter Tuning

Before our Particle Filter based method can be used, we have to make decisions about the setting of certain parameters. These parameters can be divided into model parameters and design parameters. The model parameters are used to tune the state space model so that it is in accordance with the underlying physical model. The design parameters are used to tweak the design of the filter and include the number of samples to use and the kind of measurement linearization to use.

4.2.1 Model parameters

Model dynamics

The state space contains the set of states that we use to describe the physical process that will be estimated. In our method the physical process that will be estimated are the positions of one or more objects in an object frame and the 3-d transformation that relates each object frame to the observer frame. An object is represented in the state space $\boldsymbol{x}$ by a set of n 3-d points $\{\boldsymbol{P}_i\}_{i=1}^{n}$ and the transformation is represented by a translation $\boldsymbol{T}$ and rotation $\boldsymbol{\Psi}$ as displayed in (4.8).

$$\boldsymbol{x} = [T_x, T_y, T_z, \Psi_x, \Psi_y, \Psi_z, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n]^T \tag{4.8}$$

The estimated objects are rigid objects, meaning that points on the same object have to obey a rigidity constraint. This constraint captures the fact that points on the same object cannot move relatively to each other. Because in the real world objects might not be perfectly rigid, we can model this imperfection by letting the points in our model move a little bit in the object frame. This relation holds for all directions equally and its associated uncertainty $\boldsymbol{\eta}_P$ can thus be described by a diagonal covariance matrix with elements $\sigma_X^2 = \sigma_Y^2 = \sigma_Z^2$ having a low value.

For object motion uncertainty we have the translation noise $\boldsymbol{\eta}_T$ and the rotation noise $\boldsymbol{\eta}_\Psi$. Translation uncertainty is represented with a diagonal covariance matrix $\boldsymbol{\Sigma_T} = diag(\sigma_{T_x}^2, \sigma_{T_y}^2, \sigma_{T_z}^2)$ where we propose a small uncertainty for each direction with a standard deviation of around 0.03 m. Rotation uncertainty is also represented with a diagonal covariance matrix. It is defined as $\boldsymbol{\Sigma_\Psi} = diag(\sigma_{\Psi_x}^2, \sigma_{\Psi_y}^2, \sigma_{\Psi_z}^2)$. The standard deviations of these uncertainties can be set to around 0.02 rad. These uncertainties represent the transition from frame to frame. Assuming a frame rate of 30 fps, the maximum allowed object velocity will be 0.45 m/s and the maximum allowed angular speed 0.6 rad/s. These values can be tuned, but setting them to unnecessarily high values will make the sampling very inefficient as will be seen in the section on the number of samples to choose.

Measurements uncertainty

The uncertainty in measurements, as discussed before in Section 3.3.3, is captured by the three variances σ_u^2, σ_v^2 and σ_d^2. The uncertainty in pixel coordinates u and v can be set to an equal value. No real experiments were done to estimate this uncertainty, but a rough estimate is a standard deviation of about 1 pixel for u and v. The disparity can be estimated with a bit more accuracy, since this involves a one dimensional search and as a consequence we set its standard deviation to 0.5 pixel.

4.2.2 Design parameters

In the following section, will evaluate the choice of certain design parameters. In all tests a synthetic sequence of an object translating with 0.02 m/frame along the X-axis was used.

Choice of proposal density

The choice of the proposal density used to sample from is of crucial importance. The best proposal density is the one that approaches the true posterior density (target density) the best. There are no real requirements on the proposal density, except that it should overlap the support of the target density. However, the wider the proposal density is with respect to the target density, the more samples will be needed to accurately estimate this proposal density. In [32] and [39] the relation between the proposal density and the target density was analyzed by comparing the relation between the volume of the search space (variance of proposal density) and the volume of the target space (variance of target density). If the search and target spaces are hyper ellipses with radii r_s and r_t respectively then the number of effective samples is related to the total number of samples by

$$N_{eff} \approx N_s \left(\frac{r_t}{r_s}\right)^d \tag{4.9}$$

where d is the dimension of the target (and search) space. Thus when the ratio $\frac{r_t}{r_s}$ is low, the number of samples increases exponentially with the dimension.

Since we decided to use the prior distribution as proposal distribution (Section 3.4.2), the proposal distribution is at least as wide as the target distribution, so that the only requirement that we have is met. This prior distribution however is wider than the target distribution since the state dynamics are mostly modeled as random walk models, thereby being very generous and uncertain in an accurate prediction of the true target distribution.

Number of samples

The posterior distribution of the state is approximated by a set of weighted samples. As we have seen before, the effective number of samples that represent the posterior distribution could only be a fraction of the number of samples that were drawn from the proposal distribution. The number of samples that will be needed to sample the proposal distribution is thus related to the number of effective samples we desire. The higher the number of effective samples, the lower the variance of the estimated state compared to the true state is [39]. A good measure of performance is thus the effective number of samples. This effective number of samples can be approximated, as indicated in [31] and [30], by

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N_s} w^{(i)}} \tag{4.10}$$

The relation between the effective sample size N_{eff} and the sample size of the proposal distribution N_s is given by (4.9), where the effective sample size increases linearly with the proposal sample size. The factor that relates the two is given by the ratio of the hyper volume in the target space and the hyper volume in the search space. To test this relation, a simulation was performed. For each chosen sample size N_s, the effective sample size N_{eff} (4.10) was calculated for 10 independent runs. Then for each sample size the mean and standard deviation were calculated. For a number of sample sizes this was done and the resulting relation with error bars according to the standard deviation is displayed in Figure 4.1. The results were obtained by using measurement noise of $\sigma_u = \sigma_v = 1$ px and $\sigma_d = 0.5$ px. The uncertainty in state dynamics were as follows, for the translation $\sigma_{T_x} = \sigma_{T_y} = \sigma_{T_z} = 0.03$ m and for the rotation $\sigma_{\Psi_x} = \sigma_{\Psi_y} = \sigma_{\Psi_z} = 0.02$ rad. The effect of the number of features in the state space on the effective sample size can be observed by comparing the individual figures. An increasing number of features has a decreasing effect on number of effective samples. This relation is obvious since an increasing number of features means a more peaked/smaller likelihood. Furthermore, a smaller likelihood means the posterior distribution will also be smaller so that the estimate of the state becomes more accurate. The drawback of all this, is that in the case of a very peaked likelihood only a fraction of the samples are good samples.

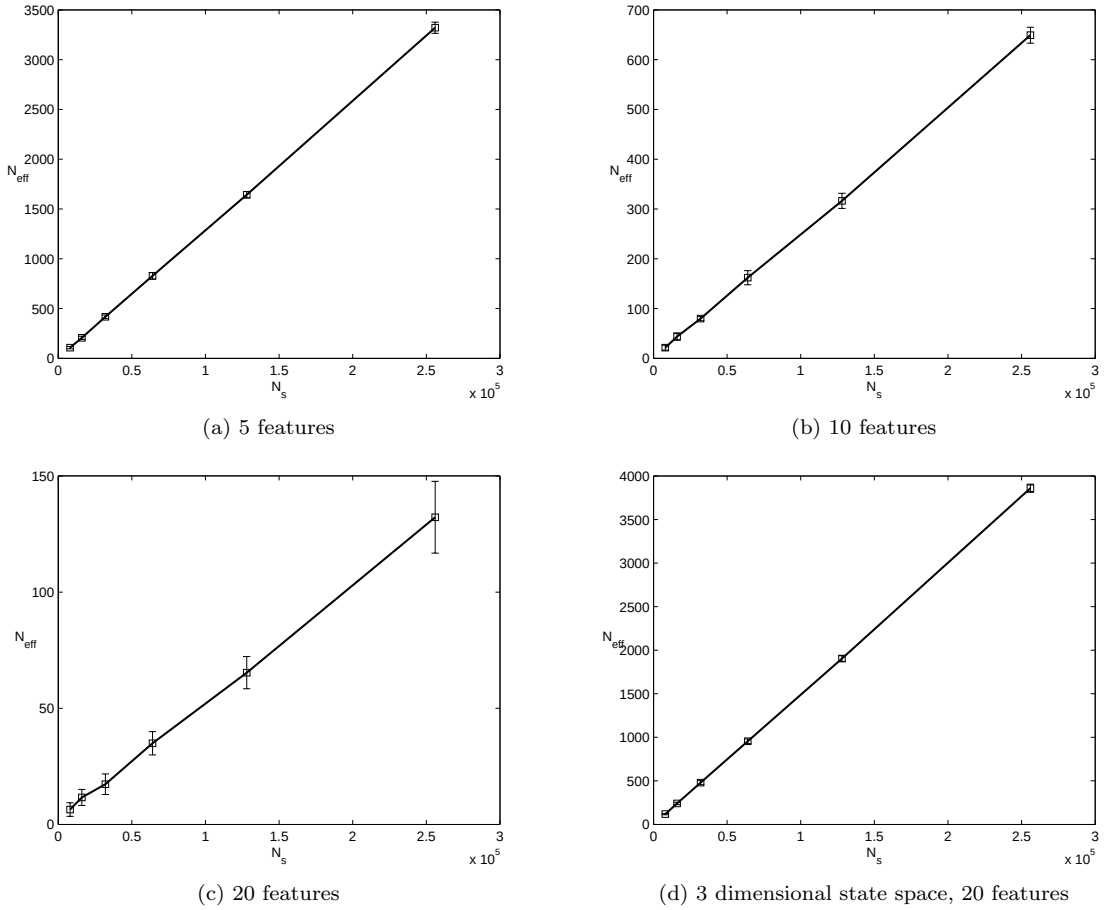
In a new simulation with 20 features, the state space was reduced to a dimension of 3, to see what effect this has on the sample survival ratio. In this simulation the rotations were set to their ground truth values and only the translation parameters were assumed to be unknown. The resulting relation between N_{eff} and N_s is given in Figure 4.1d and it is obviously works better, which is in accordance with (4.9).

Convergence

As seen in Figure 4.1, the effective number of samples increases when the likelihood is less peaked. The following question is does the particle filter converge, i.e. keep track of the true state, for a particular number of samples and number of features in the state space. This will be tested for scenarios where 5 and 10 features will be tracked. In order to quantify the convergence, we will calculate the mean square error (mse) of the estimated object structure with respect to the ground truth object structure. This error can be calculated for each separate feature in the state space. Because we will simulate a state space with 5 and a state space with 10 features, we will take the average mse on all points in the state. The performance measure on object structure convergence is thus based on the following number

$$e_{structure} = \frac{1}{N_p} \sum_{j=1}^{N_p} \mathbb{E}[(\boldsymbol{P}_j - \tilde{\boldsymbol{P}}_j)^T (\boldsymbol{P}_j - \tilde{\boldsymbol{P}}_j)] \tag{4.11}$$

where the expectation is taken over the posterior state distribution and $\tilde{\boldsymbol{P}}_j$ denotes the ground truth. For different sample sizes the simulation was performed. For each simulation, the results show the evolving



(a) 5 features

(b) 10 features

(c) 20 features

(d) 3 dimensional state space, 20 features

Figure 4.1: Comparison of N_s vs N_{eff}

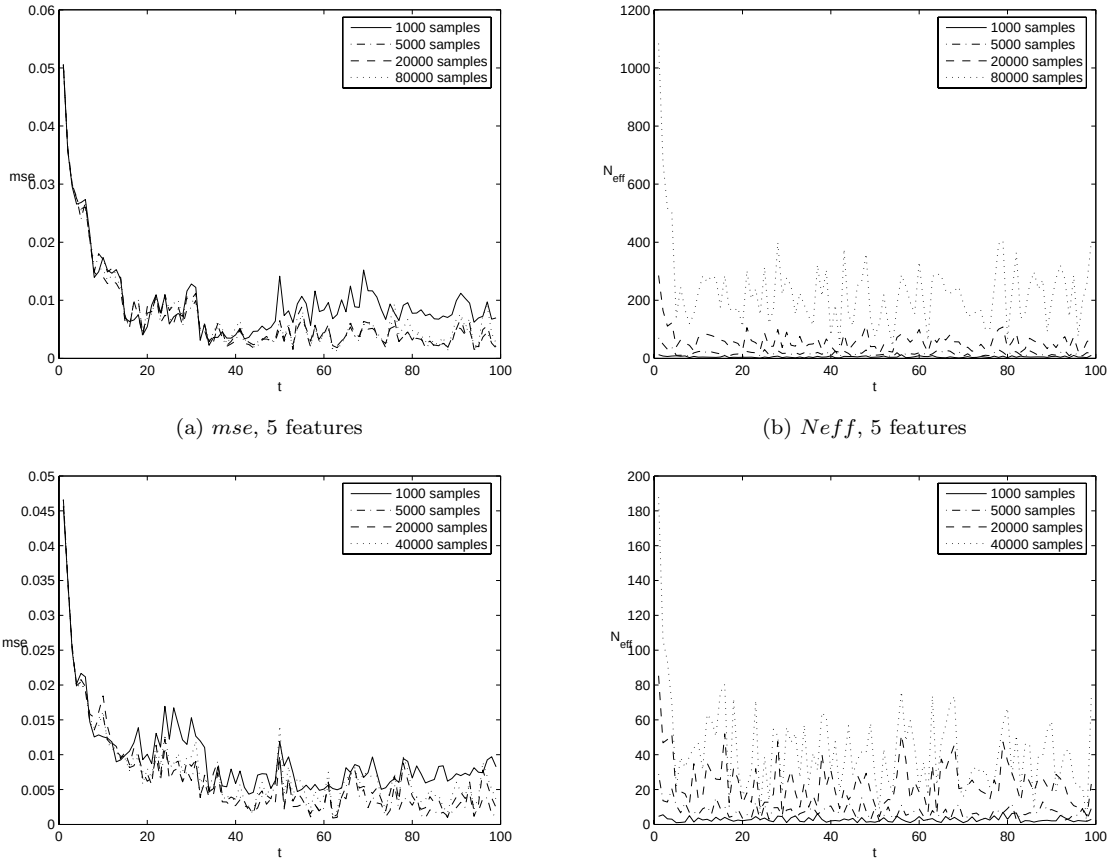
mse and corresponding effective sample sizes in the time for several sample sizes. The results of the simulations are displayed in Figure 4.2.

The remarkable result is that the mse is hardly influenced by the chosen number of samples. The mse is defined here as the difference between the points in the state space and the ground truth position of these points. In order to compare these points, the two reference frames are first aligned by using the estimated motion parameters for the state space object frame and the ground truth motion parameters for the ground truth object frame. This means the points could have changed position inside the state space frame, so that a different transformation results, leading to the same object frame. Since the mse is hardly influenced by the chosen number of samples, from 5000 samples and on there is not obvious difference, we can conclude that the particle filter converges when at least 5000 samples are used.

In a following experiment, we chose to not update the object structure, but instead use the initial object structure obtained from the first measurement. This is a similar approach as was used by Qian, where the updated object structure is not used to update the object motion estimate in subsequent estimation steps. The goal of this experiment is to find out how well the object motion can be estimated, without using a possibly internally shifted state object frame. The measure of performance used here is the mse of the estimated object motion.

$$e_{motion} = \frac{1}{N_p} \sum_{j=1}^{N_p} \mathbb{E}[(\boldsymbol{x}_{m,j} - \tilde{\boldsymbol{x}}_{m,j})^T (\boldsymbol{x}_{m,j} - \tilde{\boldsymbol{x}}_{m,j})] \tag{4.12}$$

For the case with no object structure update, two sets of simulations were performed. The results of these simulations are displayed in Figure 4.3a and b. The difference between the two is that in figure b we



(a) mse, 5 features

(b) $Neff$, 5 features

(c) mse, 10 features

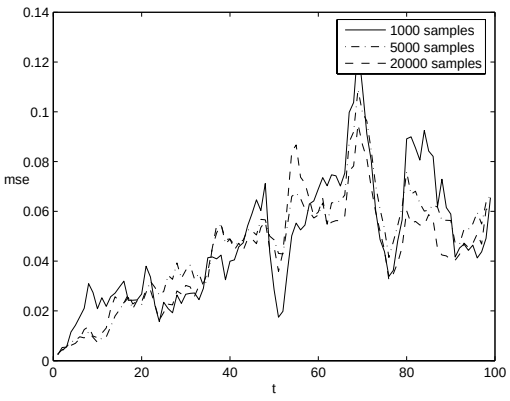
(d) $Neff$, 10 features

Figure 4.2: mse of object structure and Neff for a state space consisting of 5 and 10 features

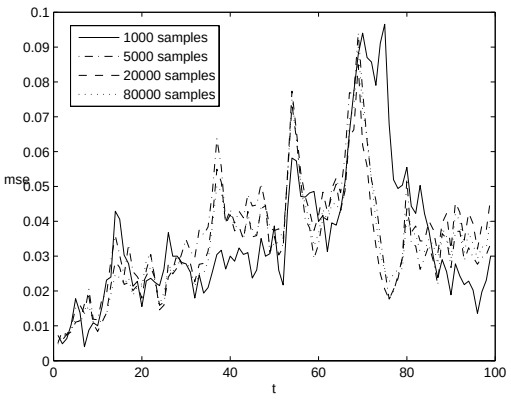
increased the object motion noise to $\sigma_T = 0.06$. Apparently this increased uncertainty on object motion gives better results, as the mse of the object motion is lower. Although the object is only translating with about 0.02 m/frame, setting the uncertainty in the translation to a higher value gives better results. The reason for this is probably due to the fact that the likelihood density is wider than we expected. To investigate if this could be the case, a graphical representation of the shape of the likelihood distribution was generated. Because essentially this likelihood distribution is six-dimensional, which is unsuitable for a plot, a 2-dimensional likelihood distribution was generated by varying only T_x and Ψ_y. To get enough coverage the boundaries of T_x and Ψ_y were set from -0.1 m to 0.1 m and from -0.03 rad to 0.03 rad respectively. This distribution was simulated for the case of 5 features in the state space and 10 features in the state space. The results are displayed in Figure 4.4. From this we can conclude that indeed the likelihood was much wider than we expected. It is actually a very peaked function, but not in all directions. Also we can conclude from the figure that the more features are in the state space, the narrower the likelihood is.

Another interesting thing to investigate, is to see what actually the effect is of using updated object structure to the estimated object motion parameters. In the experiment described above, the object structure was not updated and the mse object motion was calculated. In the following experiment we re-enabled the object structure update to see what effect this has on the accuracy of the estimated object motion. The possible drawback of updating the object structure is that the estimated transformation can not in all cases be related to the real transformation directly, since the points in the object structure can move around in their own reference frame. The results of this simulation are given in Figure 4.3c and d. Even though the internal object structure might have changed a bit, the estimate of the motion

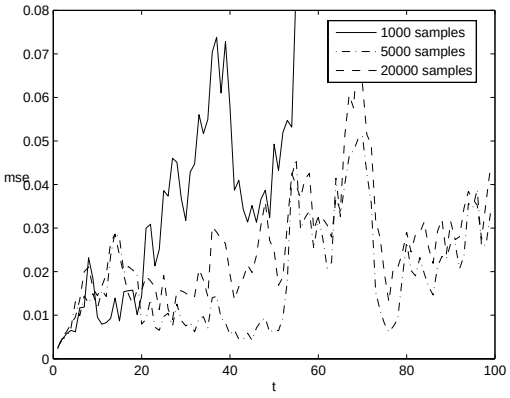
parameters is obviously better than in the case of no object structure update.



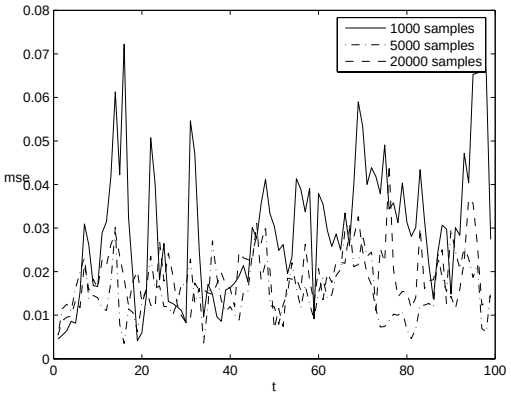
(a) $\sigma_T = 0.03$, $\sigma_\Psi = 0.02$, without structure update



(b) $\sigma_T = 0.06$, $\sigma_\Psi = 0.02$, without structure update

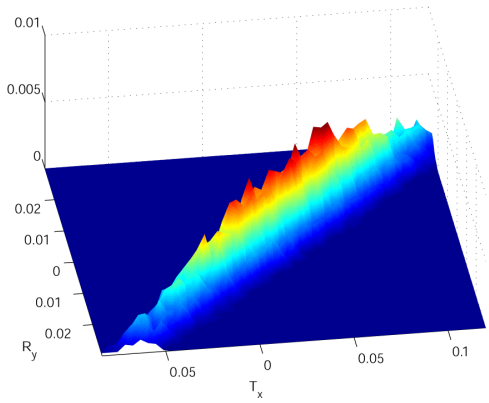


(c) $\sigma_T = 0.03$, $\sigma_\Psi = 0.02$, with structure update

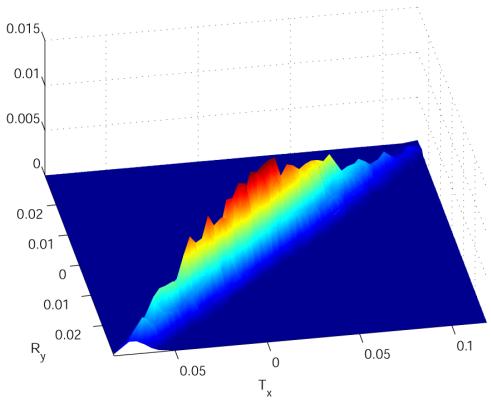


(d) $\sigma_T = 0.06$, $\sigma_\Psi = 0.02$, with structure update

Figure 4.3: mse of motion parameters



(a) 5 features in state space

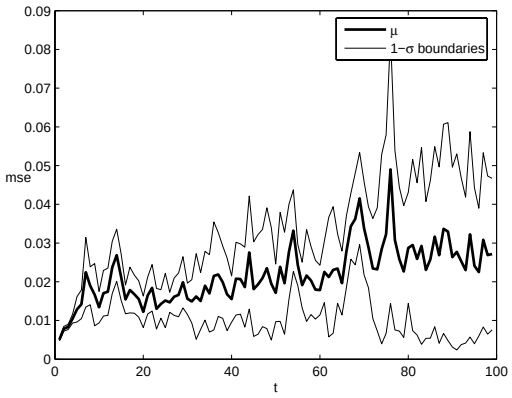


(b) 10 features in state space

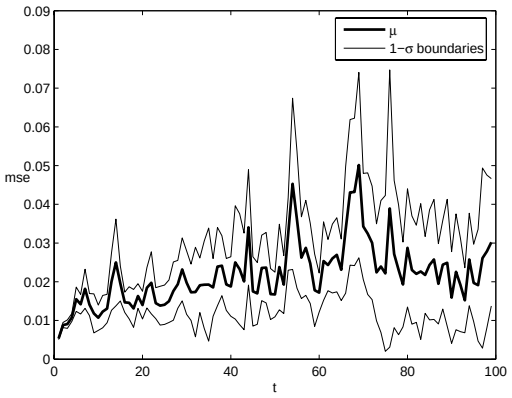
Figure 4.4: Marginal likelihood distributions, $p(\boldsymbol{z}|T_x, \Psi_y)$

Type of linearization

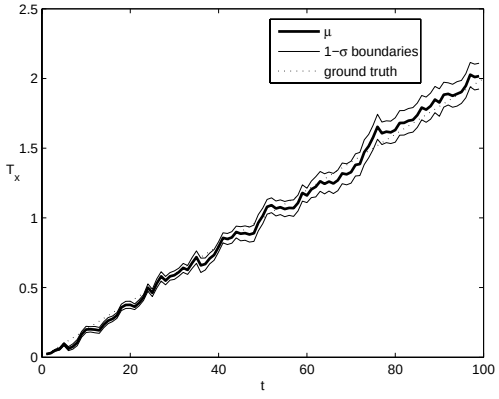
In the section on object structure and motion estimation, we talked about the kind of linearization to use in the measurement function. The two possibilities are linearization around the predicted state and linearization around the current measurement. To investigate which type of linearization performs better, a simulation was setup. In this simulation the sample size was to $N_s = 20.000$, and the scene consists again of a single moving object, translating around the X-axis with a speed of 0.02 m/frame. To quantify the performance, the mean square error of the motion parameters was calculated each time step in the run. Then for both linearization types, the simulation was run 10 times. The results of the simulations are displayed in Figure 4.5. Figures a and b show the mean square errors of the motion parameters, where its mean and standard deviation are calculated over the 10 individual runs. Figures c and d show the estimated motion component T_x over the 10 runs. Both methods keep track of the true state in the same manner. From these results we can conclude that the influence of the chosen linearization is quite small. The lower bounds of the mse error are almost similar and in the upper bounds the linearization around the measurement is a little bit higher, but not very significant. We can thus conclude that the type of linearization has no impact on the performance of the method.



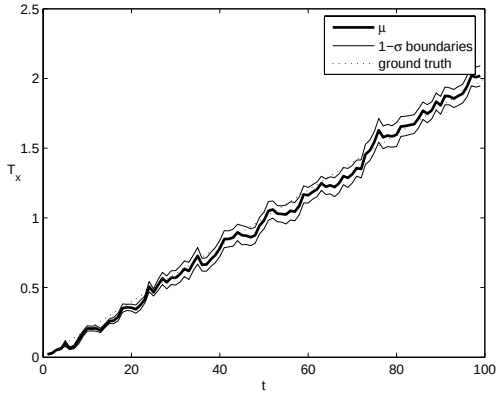
(a) Linearize around measurement, mse of motion



(b) Linearize around predicted state, mse of motion



(c) Linearize around measurement, mean of T_x



(d) Linearize around predicted state, mean of T_x

Figure 4.5: Comparison of different linearization types

4.3 Single moving object

4.3.1 Experimental setup

In this experiment, the synthetic scene consists of a single moving object. The movement of this object is defined as a translation over the X-axis and a rotation about the Y-axis. The translation over the X-axis is sinusoidal with amplitude of 0.5 m and a period of 100 frames. The rotation over the Y-axis is also sinusoidal and has an amplitude of 0.2 rad and a period of 100 frames. The rest of the transformation parameters are set to zero. The object is positioned at a distance of three meters away from the camera and has dimensions of one by one by one meters. A schematic overview of the setup is displayed in Figure 4.6. The noise of the measurement system was set to $\sigma_u = \sigma_v = 1$ px and $\sigma_d = 0.5$ px. The dynamic noise in the state space model was set to $T_x = T_y = T_z = 0.06$ m/frame and $\Psi_x = \Psi_y = \Psi_z = 0.02$ rad/frame. To make sure the particle filter will keep track of the true state, we set the sample size to a large enough value of $N_s = 40.000$.

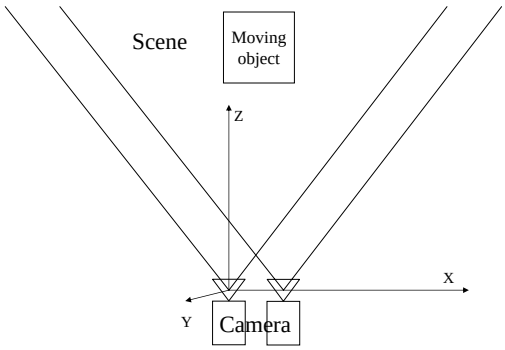


Figure 4.6: Experimental setup of single moving object

4.3.2 Results

The trajectory of the moving object and the estimated trajectory by our Rao-Blackwellized particle filter are displayed in Figure 4.7. Results are given for all motion parameters and ground truth is displayed as well. From these results we can conclude that the particle filter keeps on track. An interesting thing to investigate now, is how a Kalman filtering based method would perform on this estimation problem. Since the results show a uni-modal estimated state, a Kalman filter should be able to track the true state as well. To see the difference, an Extended Kalman filter was implemented. The Extended Kalman filter was implemented with the same state space model and measurement model as used in our Rao-Blackwellized particle filter, except that the state is represented as one large state vector instead of object structure conditioned on object motion as in our particle filter. The results of the Kalman filter tracking the state are displayed in Figure 4.8. These results show that also the Kalman filter is able to keep track of the true state.

By looking at these plots it is not directly clear which method performs better, the results seem more or less equivalent. To see the exact difference in performance with respect to the mse of the estimated motion parameters, the mse of the particle filter was plot together with the mse of the Kalman filter. These mean square error measures are displayed in Figure 4.9. The particle filter is looking just a little bit better here. But given the computational advantage of a Kalman filter, for this simple scene a Kalman filter can do the job as well.

4.3.3 Discussion

When using simple scenes, containing only one moving object, the performance of our Rao-Blackwellized particle filter based methods is comparable to the performance of a Kalman Filter based method. The performance of the particle filter is slightly better, but seen the computational advantage of using a

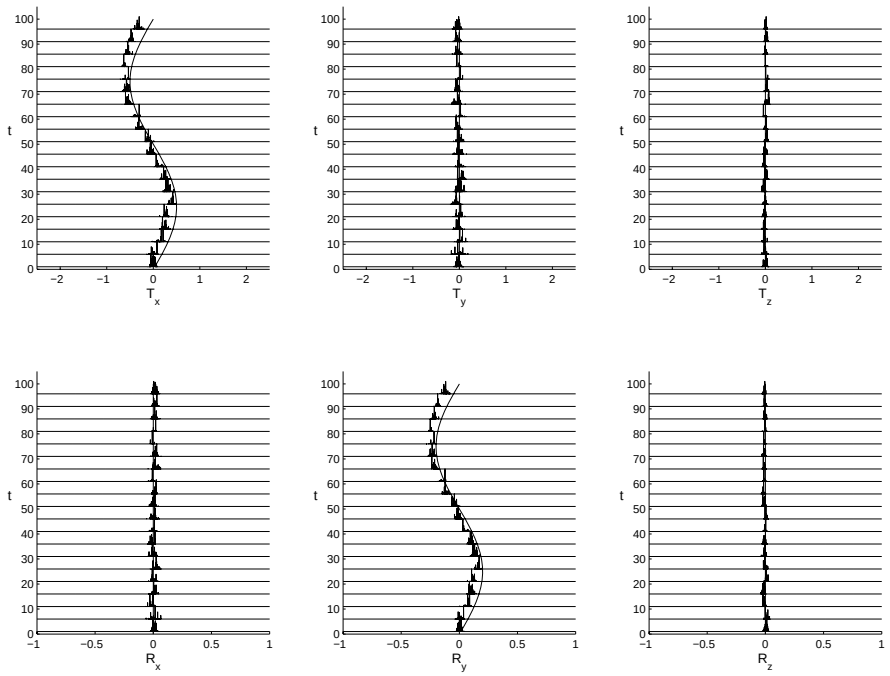


Figure 4.7: Particle filter tracking the object motion of a single moving object

Kalman Filter it has no real advantages to use the particle filter for such scenes. However, this is valid for the ideal case with Gaussian distributed measurement noise. In real situations, the measurement noise not likely to be distributed as Gaussian, it will contain several outliers. In order for the Kalman Filter to keep track under these circumstances, an outlier detection method should be added or else it will loose track and become unstable.

4.4 Multiple moving objects

The goal of this experiment is to investigate the capabilities of this method to segment and estimate the motions of multiple independently moving objects.

4.4.1 Experimental setup

A synthetic scene will be used, containing three independently moving objects. On each object 10 features will be tracked during the sequence of 100 frames. The virtual stereo camera, used to take measurements of the synthetic scene, is modeled with the parameters obtained by calibration of a real stereo camera. The noise of the measurement system was set to $\sigma_u = \sigma_v = 1$ px and $\sigma_d = 0.5$ px. The dynamic noise in the state space model was set to $T_x = T_y = T_z = 0.06$ m/frame and $\Psi_x = \Psi_y = \Psi_z = 0.02$ rad/frame. Three moving objects were used, the parameters of each of the objects are displayed in Table 4.3.

To investigate the motion estimation and motion segmentation possibilities, the simulations are in increasing order of difficulty. In order to see if the mixed representation of multiple objects in one particle filter is feasible, a simulation will be performed with manually segmented clusters. (i.e. segmentation of feature points according to their true object membership). When this works, the cluster size will be set to a lower value, which is necessary to accommodate the automatic clustering of object motions. This lower cluster size is also evaluated first with manually segmented clusters. The final test will consist of this

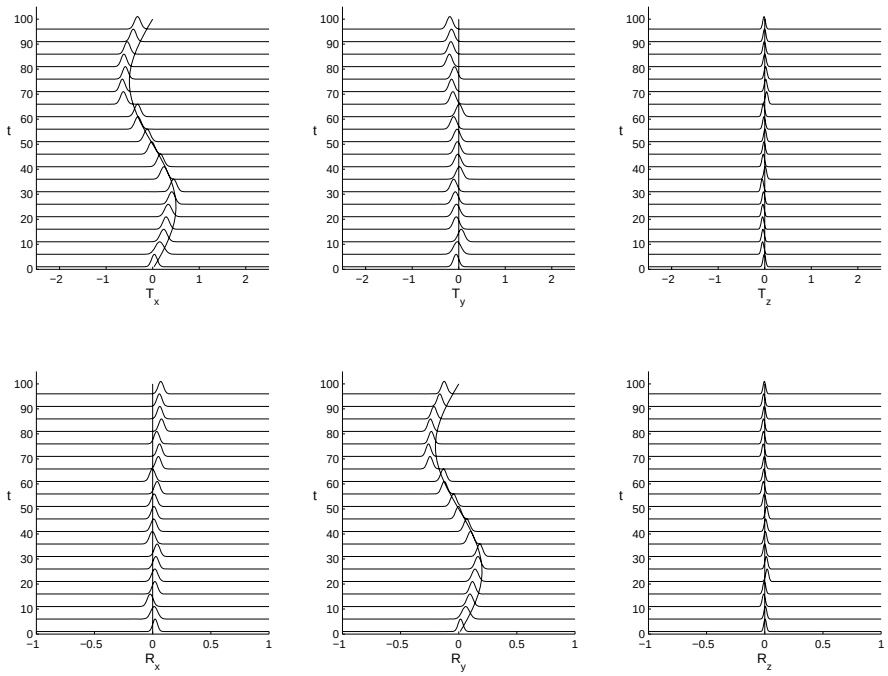


Figure 4.8: Kalman filter tracking the object motion of a single moving object

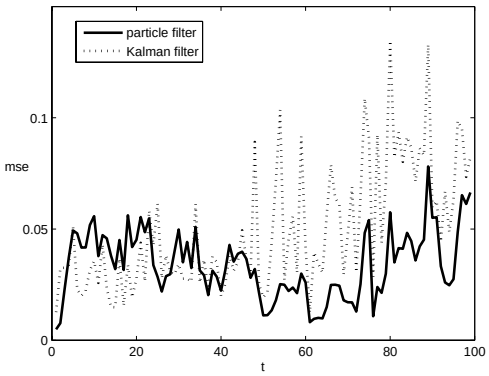


Figure 4.9: Comparison of particle filter versus Kalman filter, mean square error of estimated motion parameters with respect to ground truth

lower cluster size combined with automatically segmented motions, according to our proposed clustering algorithm.

4.4.2 Results

Ideal segmentation

We started with the case were all features were manually segmented into the correct cluster, with a cluster size of $P_m = 10$. Because the clustering might be easier when the structure is not updated, we simulated

	speed (./frame)
T_x	0.02
T_y	0
T_z	0
Ψ_x	0
Ψ_y	0
Ψ_z	0

	initial position (m)
X	-1
Y	0
Z	3

(a) Object 1

	speed (./frame)
T_x	-0.025
T_y	0
T_z	0
Ψ_x	0
Ψ_y	0
Ψ_z	0

	initial position (m)
X	1.5
Y	0
Z	4.5

(b) Object 2

	speed (./frame)
T_x	0
T_y	-0.02
T_z	0
Ψ_x	0
Ψ_y	0
Ψ_z	0

	initial position (m)
X	0
Y	1
Z	3

(c) Object 3

Table 4.3: Parameters of moving objects in object clustering experiments

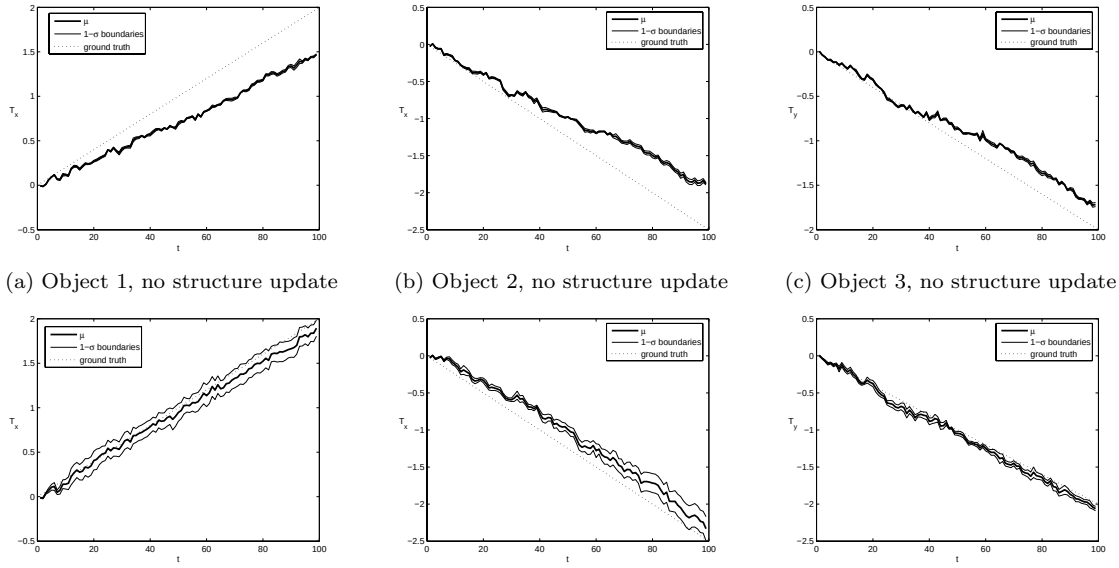
a scenario without structure updates and a scenario with structure updates. The sample size was set to $N_s = 40.000$ in these simulations. For each of the simulations, we used the manually segmented features to calculate the weights for each of the objects on the motion sample set. These three sets of weights are used to estimate the mean of the object motion parameters. The results of these simulations are displayed in Figure 4.10. The mean and variance are calculated over 4 different runs. From these results it becomes clear that by incorporating the object structure update, the estimates are much closer to the ground truth values, but have a relative big variance. Without object structure update the results are biased, but with low variance. That bias is due to the fact that the initial object structure is only based on the first measurement, which means we are using a quite rough approximation of the object structure. The best fit of the object motion, using this initial object structure, is apparently not very close to the real object motion. However, by incorporating the object structure updates, the estimates become much close to the ground truth values. In further experiments we will therefore always use the updated object structure.

Automatic segmentation

In this experiment, the minimum cluster size was set to $P_m = 5$ and the sample size was set to $N_s = 80.000$. A first simulation was performed with manually segmented clusters. The results of the estimated mean of one run and the ground truth are displayed in the top three figures of Figure 4.11. In these figures we see that eventually the ground truth is quite well approximated by the estimated mean of each of the clustered sample sets. Since this seems to work (for this lower minimum cluster size), we can extend the problem by adding the motion segmentation task to it.

In section 3.4.3 on Object clustering, we explained the object clustering which is based on the membership vectors of the individual motion samples. For this clustering problem, two methods were discussed. The first method was based on the implementation of Qian, which uses the motion samples with the highest weight to generate cluster centers. The second method was derived from the Principal Direction Divisive Partitioning (PDDP) algorithm, and looks at the highest direction of variation between all membership vectors in order to hierarchical cluster the set. Both clustering methods will be tested with the synthetic scene described above.

To quantify the results of the clustering, we used the following approach. In both methods, at each time instant, each estimated cluster is represented with a set of membership numbers/indices, indicating points in the state space, that belong to the respective cluster. These estimated clusters can change over time, the order and contents of the resulting clusters is not the same between different time instants of the sequence. To evaluate the clustering results, we have to match ground truth clusters with the estimated clusters each time step, so that we know which true cluster is accounted for by which estimated cluster. To match a true cluster with an estimated cluster, they have to share at least P_m positive membership elements in common. Different true clusters can point to the same estimated cluster, for example when



(a) Object 1, no structure update (b) Object 2, no structure update (c) Object 3, no structure update

(d) Object 1, with structure update (e) Object 2, with structure update (f) Object 3, with structure update

Figure 4.10: Manually segmented motion samples, $P_m = 10$. Shown are the mean values of the estimated object motion over different runs

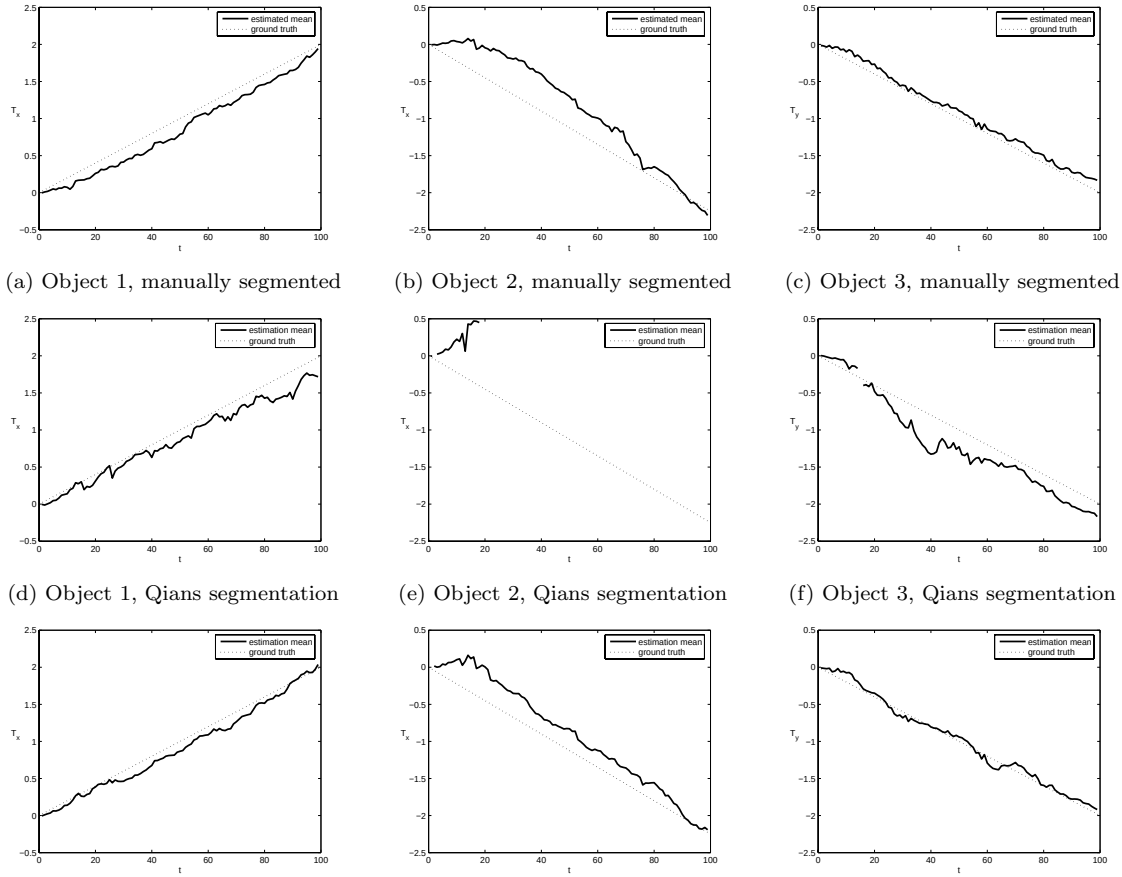
there is only one estimated cluster, containing all membership elements. After this matching has been performed, we can use properties of the estimated clusters to quantify the performance of the clustering methods.

Two properties of the estimated clusters will be used to evaluate the performance of the clustering methods. First we calculate the mean of the estimated motion for each of the estimated clusters and compare this with the ground truth of the true motion belonging to that object and secondly we will look at the membership vectors and compare these to the true memberships.

The mean will only be shown for one component of the motion, namely the one which is changing over time. For object 1 this is T_x in the positive direction, for object 2 this is T_x in the negative direction and for object 3 this is T_y in the negative direction, see also Table (4.3). The results of the calculated means are displayed in Figure 4.11. Figures d-f show the means with Qians clustering method and g-i show the means with the PDDP based clustering method. Qians clustering method works well for object 1 and object 3, but not for object 2 whose results are off the scale. On the other hand, with the PDDP based clustering method it works well for all objects.

When we look at the segmentation results for the membership points for Qians clustering method, Figure 4.12, we see again that there are some problems in finding the true segmentation. For object 1 and object 3 it seems to work fine, but for object 2 there is some influence of object 3 visible as well, especially when we look at the segmentation results over time. For the PDDP based cluster method, the segmentation is a lot better, as can be seen in Figure 4.13. It converges to the correct solution quite fast.

For the PDDP based clustering algorithm, one parameter had to be setup, which controls the splitting of nodes in the hierarchical clustering algorithm. This parameter sets a threshold on the largest eigenvalue of the estimated covariance matrix of the remaining motion samples in the node under consideration. When the largest eigenvalue of the estimated covariance is below this threshold, the set will not be split up anymore under that node. The resulting set of clusters is the set of all final end nodes after executing the clustering algorithm. In our simulation setting the threshold was no big problem and clustering worked immediately quite well, but its value might depend on the situation.



(a) Object 1, manually segmented (b) Object 2, manually segmented (c) Object 3, manually segmented

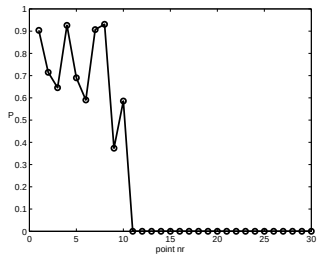
(d) Object 1, Qians segmentation (e) Object 2, Qians segmentation (f) Object 3, Qians segmentation

(g) Object 1, proposed segmentation (h) Object 2, proposed segmentation (i) Object 3, proposed segmentation

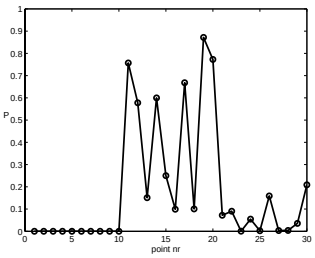
Figure 4.11: Manually and automatically segmented motion samples, $P_m = 5$. Shown are the mean values of the motion estimation for each of the estimated clusters

4.4.3 Discussion

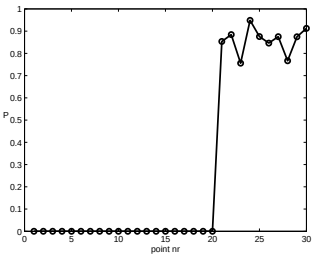
We have shown that it is possible to estimate multiple object motions together in one particle filter with the proposed method. The motion estimation works better when object structure is updated recursively as well. To evaluate the performance of the object clustering, the estimated motion for each object and the estimated segmentation (i.e. which points belong to which object) were investigated. From the results it can be concluded that by using the PDDP based clustering approach, both the motion estimation and segmentation work very well. Clustering based on Qians approach works less well, since it looses track of the second object. Both simulation were performed under similar conditions, using the same number of samples and exactly the same measurements of the scene.



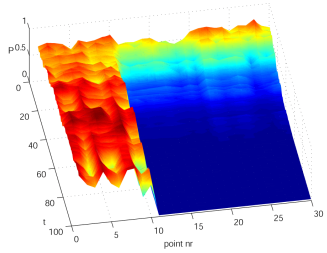
(a) Object 1, $t = 100$



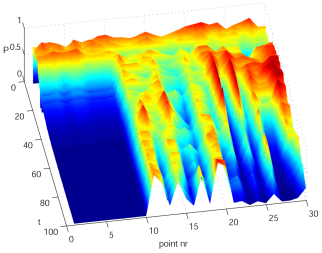
(b) Object 2, $t = 100$



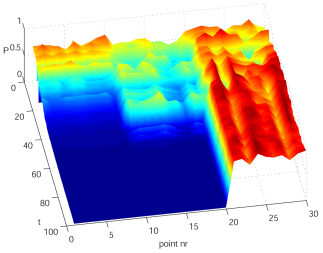
(c) Object 3, $t = 100$



(d) Object 1, whole sequence

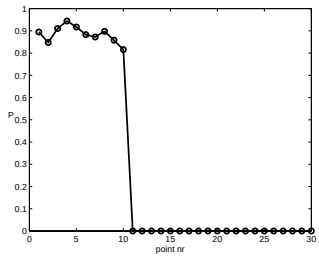


(e) Object 2, whole sequence

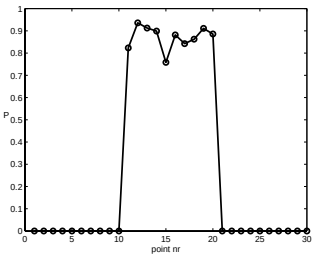


(f) Object 3, whole sequence

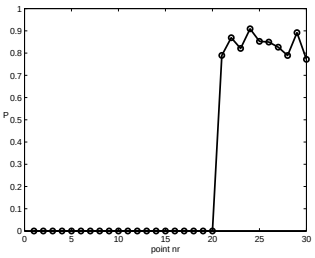
Figure 4.12: Automatically segmented clusters with Qians clustering method, the displayed plots are sample means of the membership probabilities. Figures a-c show the final segmentation in the last frame, figures d-f show the evolving segmentation for $t = 2..100$



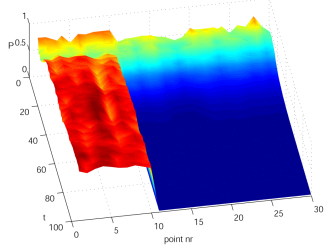
(a) Object 1, $t = 100$



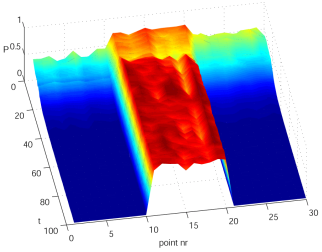
(b) Object 2, $t = 100$



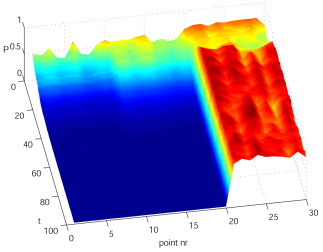
(c) Object 3, $t = 100$



(d) Object 1, whole sequence



(e) Object 2, whole sequence



(f) Object 3, whole sequence

Figure 4.13: Automatically segmented clusters with our clustering method, the displayed plots are sample means of the membership probabilities. Figures a-c show the final segmentation in the last frame, figures d-f show the evolving segmentation for $t = 2..100$

5

Conclusions and recommendations

In this section we will discuss the proposed method and the results from the experimentation. Conclusions will be drawn from these results and we will propose some additional improvements and give suggestions for further research on this topic.

5.1 Conclusions

The initial goal of this project was to investigate the weaknesses of the implemented method [3] in a previous MSc. assignment and try to improve this method by proposing a method based on particle filtering. In Chapter 2, we analyzed this previous method, together with other available methods in the field. Our analysis in that section was restricted to recursive multi frame methods, based on the scope of the project.

The analysis of the previous method lead to several limitations which could be improved. One of the biggest limitations was the fact the the previous method would only work in static scenes. Since the assumption of a scene being static is violated in many real life scenes, we took this limitation as our main objective to improve the previously implemented method. The issue of robustness of the previous method can be seen in the same context as the problem of handling a non-static scene. The previous method was not resistant to outliers, when the assumption of a static scene was violated (resulting in outliers in the measurements), the estimated scene structure and camera motion would become unstable. Another limitation which really needs to be handled, seen the scope of this project, is the unknown scaling factor of recovered distances. We took care of this limitation as well in our proposed method by using a calibrated stereo camera as measurement source. The last disadvantage of the previous method, which makes it not very practical, is the fact that only non-expanding scenes can be reconstructed. A full overview of the limitations is given at the beginning of Chapter 3. The mentioned topics here are included and solved in the proposed method.

The proposed method is mainly based on a method described in a paper by Qian [1], although there are some differences. These differences include the used measurements, the implemented state estimator and the clustering algorithm. Our method uses measurements obtained from a calibrated stereo camera, which makes it possible to estimate absolute distances, opposed to the monocular camera used in Qians method which leads to relative distances. We use a state which is split up into an object motion part represented by a set of weighted samples and, conditioned on the motion, a structure part represented by Kalman filters. In the updates on object motion, the incremented knowledge on estimated object structure is used, thereby optimally combining all available knowledge in the state estimation step. Qians method also divides the state space, but it lacks the feedback of incremented knowledge from the marginalized parts of the state space. The differences become clear when we look at the figures describing both state estimation processes. Figure 2.4 shows Qians method and Figure 3.10 shows the proposed method. The posterior knowledge at the previous time instants on marginalized states is not fed back in the respective update steps, instead only prior knowledge is used in these steps. Another limitation of Qians method is that it supports only non-expanding scenes while we allow scenes to expand by adding new features to the state space over time. The last difference between the two methods is in the clustering algorithm.

Qian uses motion samples with high weights to define cluster centers, while we use statistics of all motion samples together to find discriminative points to generate cluster centers.

The feasibility and performance of the proposed method were tested with the use of measurements obtained from synthetic scenes. The algorithm is feasible when it runs real-time. A general problem with this method, as with most sampling based methods, is its computational complexity. The time resources of the algorithm increase linearly with the amount of features being tracked (i.e. the number of available measurements) and they also increase linearly with the number of samples used in the estimation process. The memory resources are also linearly related to the number of tracked features and the amount of samples in use. However, a real time implementation of the algorithm is not yet possible.

The performance of the proposed method was measured in terms of the estimation convergence (does the filter keep track of the true state) and the segmentation capabilities (how well are points clustered into the correct motion cluster). When enough samples are used, the particle filter will keep track of the true state, as is shown in the experiments. For the case of only one object in the scene, undergoing time varying translations and rotations, the results of the particle filter on this scene are comparable to the results of using a Kalman filter, although the particle filter has a slightly lower mean square error. The segmentation capabilities of our method were investigated by using a synthetic scene containing three independently moving objects. We compared our proposed clustering algorithm with the Qians clustering algorithm. Our algorithm is capable of detecting all three objects and estimating their individual motions, Qians clustering method fails here and only detects two objects correctly. The advantage however of Qians clustering algorithm, is that no crucial parameters have to be set, while in our algorithm we have to set a parameter (threshold) that controls whether the clustering results are final, so that no more splitting should occur. However, in our experiment the value of the parameter seemed not to be very crucial for the correct operation of the algorithm, since setting it to our initial value obtained good results already.

5.2 Recommendations

More extensive simulations on both synthetic and also real life scenes should be executed. These are not included here due to the limited time available for this project. These more extensive simulations should investigate other interesting properties of our method, for example the capabilities of objects merging and splitting over time. These capabilities have to be measured for typical scenes and mainly the influence of the value of the rigidity uncertainty in the object structure update should be investigated. Another interesting issue is what kind of motion model is more appropriate (actually, more efficient with respect to the required number of samples) in representing these real life scenes, Qian uses for example a velocity motion model while we use only a Brownian motion model. A Brownian motion model leads to a lower dimensional state space but the question is whether this helps in keeping the required number of samples low.

Besides doing more experiments to investigate the performance of the proposed method better, it is also wise to look back and discuss whether the resulting method is actually feasible given the scope of the project. The proposed method is a very generic method, ideally it works for many scenes and with quite high accuracy. The drawback however, is that it is a very computational complex method, which cannot yet be implemented real time. Seen the scope of the project, a local navigation system for blind people, having a real time implementation is a requirement. The bottleneck in the performance of the proposed method is the calculation of the weights for each sample. This calculation involves matrix multiplications and matrix inversions which require a lot of processing time. To decrease the processing time, faster computer hardware is needed or better optimization should take place, the weight calculation for example can very well be parallelized.

The proposed method is generic and does not include much prior knowledge on scenes. Including more prior knowledge might make the problem easier and perhaps allow for a faster algorithm. Examples of prior knowledge which could be incorporated are the following

Background constraint A large part of the observed scene consists of a static background. The moving objects in the scene occupy less space. A consequence of this fact is that when features are selected in the image frame, in general, the majority of those features belongs to the static background. Thus a priori, the chance of a feature belonging to the background is higher than that of belonging to a moving object.

Moving objects constraint For an independently moving object, apart from the static background, we can in general assume that features being tracked on this object are closely located together in 3-d space. Thus the probability that two features, separated by a small distance in the scene, belong to the same moving object is much higher than the probability that two features, separated by a large distance in the scene, belong to the same moving object.

Clustering on the transformations, described by the motion samples, is something that could be researched as well. Having multiple independently moving objects results in multiple modes in the pdf of the motion samples. Clustering is then the task of identifying all these modes and finding the corresponding motion samples in each mode.

Bibliography

[1] G. Qian and R. Chellappa, "Bayesian algorithms for simultaneous structure from motion estimation of multiple independently moving objects," *IEEE Transactions on Image Processing*, pp. 94–108, 2005.

[2] F. van der Heijden and P. P. L. Regtien, "Wearable navigation assistance - a tool for the blind," *Measurement Science Review*, vol. 5, no. 2, pp. 53–56, 2005.

[3] J. Rozenberg, "Depth estimation using a moving camera equipped with an inertial measurement unit," Master's thesis, University of Twente, 2004.

[4] A. Azarbayejani and A. P. Pentland, "Recursive estimation of motion, structure, and focal length," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 6, pp. 562–575, 1995.

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

[6] O. Faugeras, *Three-Dimensional Computer Vision: a Geometric Viewpoint*. The MIT Press, 1993.

[7] B. K. P. Horn, "Relative orientation," *Int. J. Comput. Vision*, vol. 4, no. 1, pp. 59–78, 1990.

[8] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, Sept. 1981.

[9] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *Int. J. Comput. Vision*, vol. 27, no. 2, pp. 161–195, 1998.

[10] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Int. J. Comput. Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[11] R. Szeliski and S. B. Kang, "Recovering 3d shape and motion from image streams using non-linear least squares," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1993.

[12] J. I. Thomas, A. Hanson, and J. Oliensis, "Refining 3d reconstruction: a theoretical and experimental study of the effect of cross-correlations," *CVGIP: Image Underst.*, vol. 60, no. 3, pp. 359–370, 1994.

[13] S. Soatto and R. Brockett, "Optimal structure from motion: Local ambiguities and global estimates," in *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 282, IEEE Computer Society, 1998.

[14] M. Pollefeys, *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences*. PhD thesis, Katholieke Universiteit Leuven, 1999.

[15] L. Torresani and A. Hertzmann, "Automatic non-rigid 3d modeling from video," in *European Conference on Computer Vision*, 2001.

[16] L. Hajder and D. Chetverikov, "Robust structure from motion under weak perspective," in *International Symposium on 3D Data Processing, Visualization and Transmission*, 2004.

[17] S. Soatto, R. Frezza, and P. Perona, "Motion estimation via dynamic vision," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, 1996.

[18] S. Soatto and P. Perona, "Recursive 3-d visual motion estimation using subspace constraints," *Int. J. Comput. Vision*, vol. 22, no. 3, pp. 235–259, 1997.

[19] S. Soatto and P. Perona, "Reducing "structure from motion": A general framework for dynamic vision part 1: Modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 9, pp. 933–942, 1998.

[20] S. Soatto and P. Perona, "Reducing "structure from motion": A general framework for dynamic vision part 2: Implementation and experimental assessment," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 9, pp. 943–960, 1998.

[21] T. Jebara, A. Azarbayejani, and A. Pentland, "3d structure from 2d motion," *IEEE Signal Processing Magazine*, vol. 16, no. 3, pp. 66–84, 1999.

[22] S. You and U. Neumann, "Fusion of vision and gyro tracking for robust augmented reality registration," in *IEEE Conference on Virtual Reality*, 2001.

[23] G. Qian, R. Chellappa, and Q. Zheng, "Robust structure from motion estimation using inertial data," *Optical Society of America Journal A*, vol. 18, pp. 2982–2997, Dec. 2001.

[24] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 523–535, 2002.

[25] D. Strelow and S. Singh, "Online motion estimation from image and inertial measurements," in *IEEE Workshop on Applications of Computer Vision*, 2003.

[26] H. Jin, P. Favaro, and S. Soatto, "A semi-direct approach to structure from motion," in *International Conference on Image Analysis and Processing*, 2003.

[27] F. van der Heijden, R. P. W. Duin, D. de Ridder, and D. M. J. Tax, *Classification, Parameter Estimation and State Estimation*. Wiley, 2004.

[28] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto, "Improving feature tracking with robust statistics," *Pattern Analysis and Applications*, vol. 2, no. 4, pp. 312–320, 1999.

[29] R. E. Kalman, "A new approach to linear filtering and predictionn problems," *Transaction of the AMSE - Journal of Basic Engineering*, vol. 82, pp. 34–45, 1960.

[30] M. S. Arulampalam, S. Maskell, N. Gordon, , and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 194–188, 2002.

[31] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[32] Z. Chen, "Bayesian filering: From kalman filters to particle filters, and beyond," tech. rep., Adaptive Systems Laboratory, McMaster University, 2003.

[33] G. Qian and R. Chellappa, "Structure from motion using sequential monte carlo methods," *Int. J. Comput. Vision*, vol. 59, no. 1, pp. 5–31, 2004.

[34] A. Fusiello, E. Trucco, and A. Verri, "Rectification with unconstrained stereo geometry," in *British Machine Vision Conference*, 1997.

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

[36] J. Shi and C. Tomasi, "Good features to track," in *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[37] H. Jin, P. Favaro, and S. Soatto, "Real-time feature tracking and outlier rejection with changes in illumination," in *International Conference on Computer Vision*, vol. 1, p. 684, 2001.

[38] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of the 4th ALVEY Vision Conference*, pp. 147–151, 1988.

[39] K. Choo and D. J. Fleet, "People tracking using hybrid monte carlo filtering," in *Eighth International Conference on Computer Vision (ICCV'01)*, vol. 2, p. 321, 2001.

[40] A. Doucet, N. de Freitas, K. Murphy, and S. Russel, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 176–183, 2000.

[41] D. Boley, "Principal direction divisive partitioning," *Data Min. Knowl. Discov.*, vol. 2, no. 4, pp. 325–344, 1998.