



UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Facial Feature Reconstruction using Structure from Motion

by

Pieter Albertus Rautenbach

*Thesis presented at the University of Stellenbosch in partial
fulfilment of the requirements for the degree of*

Master of Science in Electronic Engineering
with Computer Science



Department of Electric and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leaders:

Prof. J.A. du Preez	Prof. B.M. Herbst
Electronic Engineering	Applied Mathematics
University of Stellenbosch	University of Stellenbosch

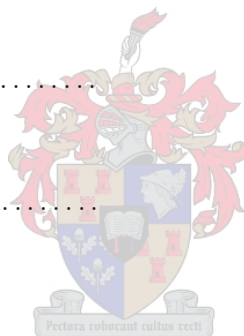
April 2005

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

Date:



Abstract

Facial Feature Reconstruction using Structure from Motion

P.A. Rautenbach

Department of Electric and Electronic Engineering

University of Stellenbosch

Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Electronic with Computer Science)

April 2005

Structure from Motion suggests that an object or scene's three-dimensional structure can be determined from its observed two-dimensional motion. Human efforts, manifested in computer algorithms, try to mimic the enormous power of the visual processing capabilities of the human brain. We present an algorithm to estimate structure, using the Unscented Kalman Filter, from the motion of point-wise features, produced by the Kanade-Lucas-Tomasi feature tracker. The algorithm is evaluated critically against an extensive set of motion sequences, with special attention paid to facial feature reconstruction.

Samevatting

Herkonstruksie van Gesigskenmerke d.m.v. Struktuur vanuit Beweging

P.A. Rautenbach

Departement Elektries en Elektroniese Ingenieurswese

Universiteit van Stellenbosch

Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MScIng (Elektronies met Rekenaarwetenskap)

April 2005

Struktuur vanuit Beweging stel voor dat 'n voorwerp of omgewing se driedimensionele struktuur vanuit die waargeneemde tweedimensionele beweging bepaal kan word. Menslike pogings, vergestalt in die vorm van rekenaar-algoritmes, probeer om die enorme krag van die visuele verwerkingsvermoëns van die menslike brein na te boots. Ons stel 'n algoritme voor om struktuur vanuit die beweging van puntsgewyse kenmerke, soos geproduseer deur die Kanade-Lucas-Tomasi kenmerksoeker, met die Unscented Kalman Filter af te skat. Die algoritme word krities teen 'n omvattende stel bewegingsekwensies geëvalueer en spesiale aandag word aan die herkonstruksie van gesigskenmerke verleen.

Acknowledgements

I would like to express my sincere gratitude toward the following people and organisations who have contributed to making this work possible:

- the Stellenbosch University and Faculty of Engineering, who supplied me with the necessary knowledge
- the National Research Foundation, who made a large contribution to the funding of my post-graduate studies
- my study leaders for their ideas, motivation and expertise
- Simon Julier and Rudolf van der Merwe, who made time to answer my questions about the Unscented Kalman Filter
- Gerhard Esterhuizen, the C++ guru
- Stephen, my companion in the DSP-lab; Michael, my flatmate, who is always keen for a braai; Francois, for intricate academic discussions; David, my best friend, who supported me during those hard times
- all my friends not explicitly mentioned here
- the University Choir, for great times
- special thanks goes to my parents who brought me up, supported me in everything I've ever done and for giving me those opportunities.

Dedications



Hierdie tesis word opgedra aan my ouers.

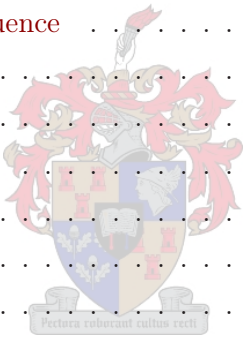
Contents

Declaration	i
Abstract	ii
Samevatting	iii
Acknowledgements	iv
Dedications	v
Contents	vi
List of Figures	ix
List of Tables	xi
Nomenclature	xii
Acronyms	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Background	2
1.3 Literature synopsis	3
1.4 Objectives	4
1.5 Contributions	4
1.6 Overview of this work	5
1.7 Previous work	6



2	In-depth exploration	7
2.1	Filtering methods	8
2.1.1	Kalman Filters	8
2.1.2	Particle Filters	8
2.2	Motion estimation	9
2.3	Structure estimation	11
2.3.1	Shape from Shading	11
2.3.2	Stereovision	11
2.3.3	Structure from Motion	11
2.4	Summary	13
3	The quaternion	14
3.1	Turning towards history	14
3.2	Why quaternions?	15
3.3	Quaternion fundamentals	16
3.4	Rotations using quaternions	18
3.5	Derivative of a quaternion	20
3.6	Summary	22
4	The Kalman Filter paradigm	23
4.1	Evolution of the Kalman Filter	23
4.1.1	Linear Kalman Filter	24
4.1.2	Extended Kalman filter	26
4.1.3	Unscented Kalman Filter	28
4.2	Summary	37
5	Structure from Motion	38
5.1	Observation modelling	39
5.2	Structure and motion modelling	43
5.3	Initialisation and setup	45
5.4	Summary	47
6	Implementation issues	48
6.1	Unscented Kalman Filter	48
6.1.1	Initialisation and setup	48

6.1.2	Optimisation	49
6.1.3	Sigma structure	49
6.2	Feature tracking	50
6.2.1	Feature window size	51
6.2.2	False features	52
6.3	Feature occlusion	52
6.4	Radial lens distortion	54
6.5	Real-time performance	56
6.6	Summary	57
7	Experimental investigation	58
7.1	Error calculation	59
7.2	Cube sequences	60
7.2.1	Pure synthetic sequence	60
7.2.2	Noisy synthetic sequence	65
7.2.3	Quasi-real sequence	69
7.2.4	Results	75
7.3	Real sequences	76
7.3.1	Hotel sequence	76
7.3.2	Face sequence	77
7.3.3	Results	82
7.4	Summary	83
8	Conclusions	84
A	Supplementary information	88
A.1	Resources	88
A.2	MATLAB scripts	89
A.3	Libraries	90
A.4	Filter software	91
A.5	Facial feature reconstruction	94
B	Supplementary CD	95
	Bibliography	96



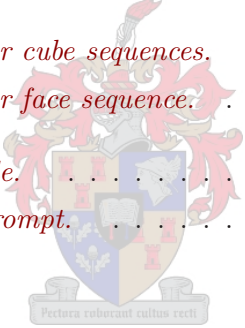
List of Figures

1.1	<i>Fool the eye and the mind with optical illusions.</i>	2
3.1	<i>An old-style guidance system using three gimbals.</i>	15
3.2	<i>A quaternion is unique, except for its sign.</i>	19
4.1	<i>The UKF correctly predicts the state.</i>	29
4.2	<i>Output states for input state with $x = 5$ and $\sigma_x^2 = 0.1$.</i>	34
4.3	<i>Output states for input state with $x = 1$ and $\sigma_x^2 = 0.1$.</i>	35
4.4	<i>Output states for input state with $x = 0.05$ and $\sigma_x^2 = 0.1$.</i>	36
5.1	<i>Linear perspective projection of a point.</i>	39
5.2	<i>Scaled points yield the same observation.</i>	40
5.3	<i>Linear perspective projection of a point in terms of the OCS.</i>	41
6.1	<i>A feature tends to slide along an edge of an object.</i>	52
6.2	<i>Motion of one feature of the hotel.</i>	53
6.3	<i>Estimated structure parameters of the hotel.</i>	54
6.4	<i>Estimated rotation of the hotel.</i>	54
6.5	<i>Estimated translation of the hotel.</i>	55
6.6	<i>The effect of radial lens distortion on straight lines.</i>	55
6.7	<i>Radially distorted and corrected frame from the face sequence.</i>	56
7.1	<i>Motion of the eight vertices of the synthetic cube.</i>	61
7.2	<i>Estimated structure parameters of the synthetic cube.</i>	61
7.3	<i>Estimated rotation of the synthetic cube.</i>	61
7.4	<i>Estimated translation of the synthetic cube.</i>	62
7.5	<i>Convergence for the synthetic cube sequence.</i>	62

7.6	<i>Extract from the reconstruction sequence of the synthetic cube.</i>	64
7.7	<i>Motion of the eight vertices of the noisy synthetic cube.</i>	65
7.8	<i>Estimated structure parameters of the noisy synthetic cube.</i>	65
7.9	<i>Estimated rotation of the noisy synthetic cube.</i>	66
7.10	<i>Estimated translation of the noisy synthetic cube.</i>	66
7.11	<i>Convergence for the noisy synthetic cube sequence.</i>	66
7.12	<i>Extract from the reconstruction sequence of the noisy synthetic cube.</i>	68
7.13	<i>One frame from the POV-Ray sequence.</i>	69
7.14	<i>Motion of the six features of the POV-Ray cube.</i>	69
7.15	<i>Estimated structure parameters of the POV-Ray cube.</i>	70
7.16	<i>Estimated rotation of the POV-Ray cube.</i>	70
7.17	<i>Estimated translation of the POV-Ray cube.</i>	70
7.18	<i>Convergence for the POV-Ray cube sequence.</i>	71
7.19	<i>The reconstructed POV-Ray cube and its lengths.</i>	71
7.20	<i>Extract from the reconstruction sequence of the POV-Ray cube.</i>	74
7.21	<i>One frame from the hotel sequence.</i>	76
7.22	<i>Two different views of reconstructed hotel.</i>	77
7.23	<i>One frame from the face sequence.</i>	77
7.24	<i>Estimated structure parameters of the face.</i>	78
7.25	<i>Estimated rotation of the face.</i>	78
7.26	<i>Estimated translation of the face.</i>	78
7.27	<i>Head-on view of the reconstructed face.</i>	80
7.28	<i>Two different views of reconstructed face.</i>	80
7.29	<i>Extract from the reconstruction sequence of the face.</i>	81
A.1	<i>Directory structure of the project.</i>	88
A.2	<i>DtFilter_N class hierarchy.</i>	91
A.3	<i>UKFFunction class hierarchy.</i>	92

List of Tables

4.1	<i>Summary of the LKF's equations.</i>	25
4.2	<i>Summary of the EKF's equations.</i>	27
4.3	<i>Weights needed by the UKF.</i>	29
4.4	<i>Initialisation of the UKF.</i>	30
4.5	<i>Summary of the UKF's equations.</i>	31
4.6	<i>Comparison of the EKF to the UKF.</i>	37
7.1	<i>Parameters and results for cube sequences.</i>	75
7.2	<i>Parameters and results for face sequence.</i>	82
A.1	<i>A sample configuration file.</i>	93
A.2	<i>A sample configuration prompt.</i>	93



Nomenclature

Constants:

- f focal length
- m number of features
- n augmented state dimension

Variables:

- s structure depth
- θ_{dim} rotation angle [†]
- ω_{dim} angular velocity
- t_{dim} translation
- d_{dim} translation velocity



Scalars, Vectors and Matrices:

- a scalar quantity
- \mathbf{n} unit vector
- \mathbf{q} quaternion
- \mathbf{x} column or row vector (context dependent)
- \mathbf{I} identity matrix
- \mathbf{P} matrix
- \mathcal{X} sigma point distribution

[†] dim denotes the dimension, e.g. x , y or z .

Functions and Operators:

- $E(\cdot)$ expectation operator
- $\mathbf{f}(\cdot)$ vector function
- $p(\cdot)$ general Probability Density Function
- $\mathcal{N}(\cdot)$ Gaussian Probability Density Function
- $O(\cdot)$ computational complexity

Subscripts:

- i i -th time-step
- j j -th sigma point
- k k -th feature



Acronyms

2D	two-dimensional
3D	three-dimensional
ANN	Artificial Neural Network
CCS	Camera Coordinate System
CMS	Critical Motion Sequence
COP	Center Of Projection
DOF	Degree Of Freedom
EKF	Extended Kalman Filter
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
HOT	Higher Order Term
KF	Kalman Filter
KLT	Kanade-Lucas-Tomasi
LKF	Linear Kalman Filter
MMSE	Minimum Mean-Square Error
OCS	Object Coordinate System
PDF	Probability Density Function



PF	Particle Filter
RMSE	Root-Mean-Square Error
RV	Random Variable
SfS	Shape from Shading
SfM	Structure from Motion
SMC	Sequential Monte Carlo
SUT	Scaled Unscented Transform
UKF	Unscented Kalman Filter
UT	Unscented Transform
VR	Virtual Reality



Chapter 1

Introduction

This report, by its very length, defends itself against the risk of being read.

Sir Winston Churchill

The human visual system is one of the greatest pieces of biological engineering, equipped with a high-resolution, high-definition colour, stereo cameras, driven by a high-speed processing unit, computing detail about lighting, perspective, structure and motion within the wink of an eye. We can fool the eye and the mind with optical illusions, but still, we struggle to mimic this incredible piece of equipment. Engineers, on a quest to conquer, arise with numerous ways. One of which is Structure from Motion (**SfM**).

1.1 Motivation

SfM suggests that an object or scene's three-dimensional (**3D**) structure can be determined from its observed two-dimensional (**2D**) motion. **SfM** is preferable to its counterparts, Shape from Shading (**SfS**) and Stereovision. **SfM** avoids the use of multiple cameras when compared to Stereovision implementations. Previous **SfM** implementations use the Extended Kalman Filter (**EKF**) as estimator (**Azarbayejani and Pentland, 1995**). We refine that approach, using the Unscented Kalman Filter (**UKF**), which is better suited to nonlinear estimation than the **EKF** (**Julier and Uhlmann, 1996**). This refined approach is tested on

(a) *Waterfall by MC Escher.*(b) *Pavement art by Julian Beaver.***Figure 1.1:** *Fool the eye and the mind with optical illusions.*

a number of sequences, including synthetic and real sequences. We conclude with a convincing experiment, performing facial feature reconstruction.

1.2 Background

Computer Vision is a broad field, which includes numerous areas of research. The area of surveillance deals with the tracking of human activities. Automatic steering of cameras is applicable to the area of video- and teleconferencing. Another area of interest is that of autonomous navigation, whether it is a vehicle or a robot. The character Gollum in the motion video *The Lord of the Rings* is an example of Human-Computer Interaction (HCI). A human plays the character and is afterwards replaced by the appropriate avatar. Some recent additions to the field of Computer Vision is the prediction and tracking of 3D trajectories in ball sports.

A SfM system has three compulsory components:

- a feature tracker
- an estimator
- structure, motion and projective models.

First in this list is the feature tracker. As the name suggests, it tracks a number of features within a motion sequence. Features within a frame may be pixels, point-wise features defined by corners or distinct patches of texture, or parametric curves, which form the outlines of an object or parts of it. Pixel-based feature tracking methods, which take all pixels within each frame of a sequence into account, are referred to as dense optical flow methods. Point-wise feature tracking methods, which track a limited number of features, instead of all pixels, are referred to as sparse optical flow methods. We opt for the Kanade-Lucas-Tomasi (KLT) feature tracker, a sparse optical flow method, to track point-wise features within a sequence. The 3D structure and motion of the tracked object are then estimated from the subsequent 2D observed features. Different flavours of the Kalman Filter (KF), such as the EKF or the UKF, are typically used for this purpose. The projective model relates the structure and motion models to the observations. In this case, it is done via a linear perspective projection model for a pinhole camera.

1.3 Literature synopsis

The full literature study is discussed in Chapter 2, but we give an overview of the literature that has been most useful to us. The SfM system presented in this thesis is centred around the UKF. The UKF, introduced by Julier & Uhlmann (1996), is one of the nonlinear extensions of the Linear Kalman Filter (LKF), originally introduced by Kalman (1960). The UKF is considered a true nonlinear extension of the LKF, as opposed to the EKF, which makes use of linearisation techniques.

Online estimation is synonymous to filtering. Filtering is the process of estimating the internal state of a dynamic system, given a set of past and present observations in the presence of noise. Previous work by Azarbayejani & Pentland (1995) is based on point-wise feature reconstruction, using the EKF. They estimate the structure by estimating only the structure depth parameter for each feature tracked, as opposed to previous methods that use the full 3D coordinates (Broida *et al.*, 1990). The motion is modelled by estimating the rotation and translation parameters of the object. One of the intrinsic parameters of a camera, the focal length, is also estimated. A linear

formulation of the perspective projection model for a pinhole camera is used. It has the advantage that the orthogonal projection model is a special case of the perspective model when compared to the traditional formulation. We exploit and extend their approach, using the **UKF** instead of the **EKF**, and incorporate additional motion parameters.

1.4 Objectives

We summarise the objectives of this thesis in the following list:

- to achieve a better understanding of quaternion mathematics and its applications (Chapter 3)
- theoretical comparison of the different **KF** topologies (Chapter 4)
- relevance of the **UKF** to the problem of **SfM** (Chapter 4)
- incorporate feature occlusion (Chapter 6)
- accurate **3D** tracking (Chapter 7)
- accurate **3D** reconstruction (Chapter 7).

The quaternion stands central to rotation estimation in **SfM**. We present a detailed discussion about the time-derivative of a quaternion. This will form part of the motion dynamics as discussed in Chapter 5. A comparison of the different **KF** topologies leads us to the conclusion that the **UKF** is superior to the **EKF**. Feature occlusion is investigated in Chapter 6. We propose two methods to overcome the occurrence of this effect. Last, but not least, are the experiments in Chapter 7, showing that accurate **3D** tracking and reconstruction is possible, using simple structure and motion modelling.

1.5 Contributions

Few solutions to handle occlusion well exist. Due to the complexity of the problem, most of the available literature propose very complex methods (Nickels and Hutchinson, 2001) or just ignore the problem (Qian and Chellappa, 2001). In Section 6.3, we propose two simple methods to deal with feature occlusion:

- predict the **3D** location of an occluded feature
- replace the occluded feature.

It is assumed in this thesis that the tracked object is rigid. Based on this assumption, the **3D** location of an occluded feature can be predicted, since its position relative to the object's centroid is fixed. Unfortunately, we find that this approach performs poorly if the last visible estimate of the occluded feature's **3D** position has not yet converged.

Another way to handle occlusion is to replace the occluded feature with another non-occluded feature, given that the feature tracker is able to find another unoccluded feature. Some post-processing is needed to incorporate all features tracked throughout the sequence. We show that this approach performs well under the assumption that the occlusion rate is low.

1.6 Overview of this work

A vast amount of information is available on the subject of Computer Vision. We give an overview in Chapter 2, covering some commonly used methods and applications, giving special attention to **SfM**. The next three chapters are of a theoretical nature. The first of these (Chapter 3) is dedicated to the quaternion and its applicability to **SfM**, especially its usefulness for representing rotations. A formula for the time-derivative of a quaternion is derived. Chapter 4 focuses on the **KF** family, giving an overview of one of the offspring, the **LKF**. We compare the nonlinear extensions of the **KF**, the **EKF** and **UKF**, based on prediction methods and modelling, computational complexity and statistical accuracy—refer to Subsection 4.1.3. In Chapter 5, the linear perspective projection model for a pinhole camera is presented. The formulation of the structure and motion models forms the second part of the chapter and we conclude with a section about the initialisation and setup of the **UKF**.

This work gains depth when some of the implementation issues encountered are discussed in Chapter 6. It includes some notes regarding the implementation and optimisation of the **UKF**. A short overview of the **KLT** feature tracker is given, which leads to one of the contributions of this thesis in Section 6.3—

a way to handle feature occlusion. The effect of radial lens distortion, as applicable to real sequences, is investigated in Section 6.4.

This thesis nears its end when the experiments are discussed in Chapter 7. A number of experiments are included, starting with a pure synthetic experiment to test our implementation of the UKF. The experiments get closer to reality as we add the KLT feature tracker and apply the SfM system as a whole to rendered and real sequences. We conclude with a display of facial feature reconstruction in Subsection 7.3.2.

In Chapter 8, we take a step back and put this work into perspective. We review how the proposed objectives were achieved and the results obtained. This work is concluded with a number of suggestions for future improvements.

Supplementary material is provided in the appendices. Appendix A serves as a user's guide for the software provided in Appendix B on CD. It contains, among other stuff, all the software and sequences used in this thesis.

1.7 Previous work

This thesis builds on the work by Venter (2002), who compared the EKF and UKF on an experimental basis to determine its suitability in a SfM system. Our motion modelling is simpler, compared to the models proposed by Venter (2002). We combine our ideas with those of Azarbayejani & Pentland (1995) and introduce a method to handle feature occlusion.

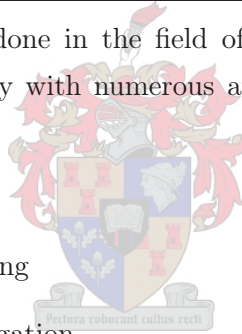
Chapter 2

In-depth exploration

Basic research is like shooting an arrow in the air and, where it lands, painting a target. *Homer Adkins*

Extensive research has been done in the field of Computer Vision. It is a field of huge interest to society with numerous applications. It includes the following:

- surveillance
- video- and teleconferencing
- autonomous vehicle navigation
- robotics
- Human-Computer Interaction
- Virtual Reality.



More recent additions to this field include the prediction and tracking of three-dimensional (3D) trajectories in ball sports, such as tennis, cricket and golf, to name only a few. There is also interest in tracking the players. Interesting deductions can be made from the collected statistics.

We discuss some of the most commonly used methods and applications in the field of Computer Vision, as applicable to this thesis. This chapter concludes with a section about Structure from Motion (SfM).

2.1 Filtering methods

We first discuss two of the most widely used filtering methods in the field of Computer Vision. Filtering is the process of estimating the internal state of a dynamic system. A distinction between parametric and nonparametric methods is made. The Kalman Filter (**KF**) is by nature a parametric method, since it estimates the parameters of a Probability Density Function (**PDF**) (Kalman, 1960; Sherlock and Herbst, 2003). Nonparametric methods include Sequential Monte Carlo (**SMC**) methods, such as the Particle Filter (**PF**), which alternatively estimates the unknown **PDF**, using a number of discrete samples or particles from which the necessary statistics can be calculated (Isard and Blake, 1998; van der Merwe *et al.*, 2000).

2.1.1 Kalman Filters

The control systems society favours the use of **KFs**, due to their ease of implementation, robustness and low computational complexity. The **KF** family consists of three well-known members, such as the Linear Kalman Filter (**LKF**), to handle linear problems, and the Extended Kalman Filter (**EKF**) and Unscented Kalman Filter (**UKF**), to handle nonlinear problems. **KFs** operate on the principle that statistical modelling using a Gaussian **PDF** is adequate. The parameters estimated by the **KF** are the first two moments of the **PDF** — the mean and the covariance which are the only non-zero moments.

The **EKF** approximates nonlinear models by assuming a first order Taylor series expansion of the modelling equations. On the other hand, the **UKF** uses a set of deterministically chosen sigma points to represent the **PDF** exactly. Sigma points are discrete n -dimensional samples, chosen in such a way that it contains the same statistical information as the original **PDF**. These sigma points are propagated using the nonlinear model itself, instead of using a first order approximation. We discuss **KFs** in detail in Chapter 4.

2.1.2 Particle Filters

The Particle Filter is a **SMC** method that uses a set of weighted, randomly drawn samples or particles to represent the prior and posterior **PDFs**. It is

common to use the **EKF** or **UKF** to propose the prior **PDF**. A set of particles are sampled from the prior **PDF** and each particle is evaluated to calculate the particles that estimate the posterior **PDF**. Given an adequate number of particles, the estimated **PDF** will converge to the true **PDF**. In theory, **PFs** deal with any kind of nonlinearity or **PDF**. Any of the required statistics, such as the mean or covariance, can be calculated from the particles. Some implementations of the **PF** suffer from particle degeneration—this means that one particle tends to gain all the weight during propagation, while the rest of the particles lose theirs. However, there are methods to overcome this. **PFs** also tend to be computationally expensive—the number of particles needed grows exponentially with the state dimension. More detailed discussions can be found in the literature of [Isard & Blake \(1998\)](#) and [van der Merwe *et al.* \(2000\)](#).

2.2 Motion estimation

When discussing literature about tracking, one has to distinguish between two-dimensional (**2D**) and **3D** tracking. Parametric curves, using splines or snakes for example, are usually applied to **2D** problems—the curve provides a segmentation of the image, defining the region to be tracked. Point-wise features are commonly used in **3D** problems. In this case, the projected **2D** motion of the features is correlated in such a way that the **3D** information can be recovered. Parametric curves can act as an aid to **3D** problems to facilitate the localisation of features.

The **2D** tracking of multiple objects with the possibility of occlusion in moderately complex scenes is a problem of interest in surveillance applications. Conventional methods, tracking only one object at a time, work well. The more complicated problem of tracking multiple objects is attempted by [Dockstader & Tekalp \(2001\)](#). Novel modifications to the standard **KF** are introduced. They track each object using frame differencing, each combined with its own modified **KF**. They notice that the observations of the different objects are no longer independent during occlusion and introduced a near real-time method of probabilistic weighting to let the **KFs** interact.

[LaViola \(2003\)](#) developed a system which accurately tracks **3D** human mo-

tion for use with Virtual Reality (VR) applications. They are particularly interested in real-time head and hand motion. A comparison between the **EKF** and **UKF** is performed, based only on the orientation of the tracked object—the filters estimate the orientation in terms of quaternions and angular velocities. They conclude that the **EKF** performs slightly better than the **UKF** and argue that the **UKF** will only achieve better estimates when the higher order moments, such as the kurtosis, are significant.

Another **2D** tracking method by **Isard & Blake (1998)** tackles the challenging problem of tracking curves in the presence of dense visual clutter. They claim that **KFs** are inefficient, because they rely on unimodal Gaussian modelling. Thus, they introduce a member of the **PF** family called Condensation (conditional density propagation). The object is located approximately in the first frame and tracked in subsequent frames. B-splines are used to parameterise the shape's curves. Their near real-time approach is capable of highly robust tracking of erratic motion. They developed impressive demonstrations, for example tracking a camouflaged leaf in the presence of gusts of wind.

Li & Zhang (2002) attempt the identical problem, but rather use the **UKF**. They prefer the **UKF** over Condensation, due to the large number of particles of the Condensation algorithm. Performance is enhanced by training their algorithm with a sequence of a hand moving against a clutter-free background, before testing it against a cluttered background. Even so, their algorithm gets distracted by the background clutter and never recovers. **Chen et al. (2002)** reach a more satisfying conclusion when using a Hidden Markov Model (**HMM**) to identify the object to be tracked.

Object tracking is not limited to visual tracking only. **Ward et al. (2003)** use a **PF** to track an acoustic source in a reverberant environment. This kind of problem is applicable to camera steering for videoconferencing. Their approach uses an array of microphones to create a beamformer. Time-delay estimation is done, using the cross-correlation among the microphones. Spurious peaks, with a greater value than that of the correct peak due to reverberation, may exist. The **PF**'s task is to track the correct peak. Their approach seems successful within moderately reverberant environments.

Rui & Chen (2001) combine the audio and visual techniques of **Ward et al. (2003)** and **Isard & Blake (1998)** to perform tracking. They specifically use the

UKF to generate the prior **PDF** and propagate it using Condensation. Their results are superior to those obtained by [Isard & Blake \(1998\)](#).

2.3 Structure estimation

There are several methods to estimate an object or scene's **3D** structure, three of which are discussed here.

2.3.1 Shape from Shading

In their paper, [Atick *et al.* \(1996\)](#) suggest that the human visual system gains information about **3D** shapes, using the shading patterns in a **2D** image. They back this statement by arguing that shading is often used by painters to convey **3D** perception, contributing to the realism of their work. Shading is the variation in brightness from one point to another in an image, due to the amount of light a surface patch reflects. This is affected by the texture properties of the patch and its orientation relative to the incident light. They also suggest that the human brain classifies objects into lower object classes according to their shape. They speculate that they are able to recover the **3D** surface from a single **2D** image of a face, but do not include any conclusive results.

2.3.2 Stereovision

Stereovision uses a disparity map to relate a feature within one frame to its image in another frame. The two frames are obtained from two differently positioned and calibrated cameras. If the intrinsic parameters, such as the focal length and lens distortion, and extrinsic parameters, such as the location and orientation, of each camera are known, the disparity map can be used to reconstruct the scene under observation—an absolute **3D** model can be constructed from this. This is a very accurate, but computationally expensive method.

2.3.3 Structure from Motion

SfM is a method to reconstruct an object or scene's **3D** structure from its observed **2D** motion. **SfM** is similar to Stereovision in the sense that it ex-

exploits the interframe differences to obtain the structure. It is preferred over its Stereovision counterpart, due to lower computational complexity, although SfM processes many frames, compared to Stereovision's two.

Accurate tracking is central to SfM. Models for the relative motion of the camera to the object and the camera's projective geometry have to be developed. These are respectively referred to as the state transition model and observation model. The state transition model contains the structure and motion parameters and predicts the new state based on the previous state. The observation model uses the predicted state to create a predicted observation. The difference between the real and predicted observations is then minimised to obtain a better estimate of the state. It is typical to use the UKF or PF for this purpose. The object is normally assumed to be rigid or at least composed of several rigid parts.

Detailed research have been done by Azarbayejani & Pentland (1995), Jebara & Pentland (1996) and Jebara *et al.* (1999). They formulate a method for recursive recovery of motion, point-wise structure and one of the camera's intrinsic parameters, the focal length. In addition to this, they redefine the camera's linear projective geometry in such a way that the orthographic camera model becomes a special case of the perspective camera model. Their approach is based on the EKF—the results show that their formulation is more stable and accurate than earlier formulations (Broida *et al.*, 1990). We fully discuss our adapted approach in Chapter 5.

The work of Qian *et al.* (2001) fuses inertial kinematics, additional to the rotation and translation parameters, into a general SfM framework. They show that the inertial data play an important role in improving resistance to tracking noise. In another paper by Qian & Chellappa (2001) the problem of SfM using SMC methods is addressed. Errors in feature tracking are modelled and they are capable of dealing with feature occlusion.

Two kinds of ambiguities exist in SfM, namely structure ambiguity and motion ambiguity. Structure ambiguity is the effect where differently scaled versions of the same object yield the same projection. Thus, the true structure can only be determined up to a scale factor. Fortunately, if one feature's true location is known, the rest can be scaled accordingly to recover the absolute structure (Szeleski and Kang, 1996). On the other hand, motion ambiguities

pose an unsolvable problem: these are motions that cannot be uniquely determined by any algorithm. Such motions are referred to as Critical Motion Sequences (CMSs) as discussed by [Sturm \(1997\)](#). We do not investigate this.

2.4 Summary

Some of the most important literature in the field of Computer Vision, as related to SfM, including recent additions, was covered. We gave an overview of some of the filtering methods used and some of the applications in this field. In this thesis, we use the UKF to perform SfM.

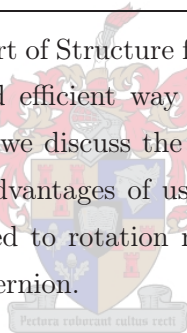


Chapter 3

The quaternion

To everything, turn, turn, turn. *The Byrds*

Quaternions form an essential part of Structure from Motion (SfM) estimation, since they provide a simple and efficient way to describe the rotation of a tracked object. In this chapter, we discuss the fundamental mathematics, as well as the advantages and disadvantages of using quaternions. We will also show how quaternions are related to rotation matrices and derive a formula for the time-derivative of a quaternion.



3.1 Turning towards history

William Rowan Hamilton invented the quaternions in 1843 in an effort to construct hypercomplex numbers or higher dimensional generalisations of the complex numbers. Failing to construct a generalisation in three dimensions in such a way that division would be possible, he considered systems with four complex units and arrived at the quaternions. He realised that each one of his complex units could also be associated with a rotation in space. Vectors were introduced by Hamilton for the first time as pure quaternions and vector calculus was at first developed as part of this theory. Maxwell's electromagnetism equations were first written using quaternions (Coutsias and Romero, 1999).

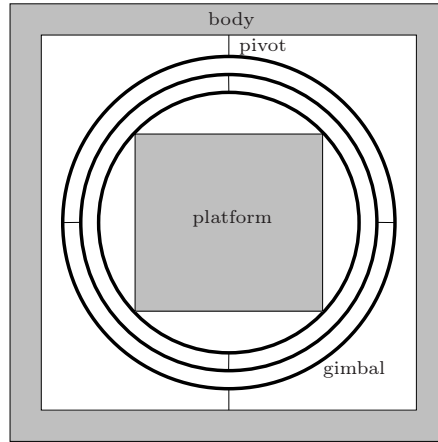
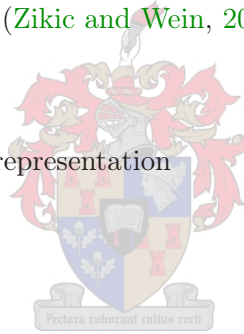


Figure 3.1: An old-style guidance system using three gimbals.

3.2 Why quaternions?

Quaternions have many advantages over their matrix, Euler axis-and-angle and Euler angles counterparts (Zikic and Wein, 2004):

- intuitive and simple
- minimum and sufficient representation
- unambiguous
- no gimbal lock
- easier rotation interpolation (Eberly, 2002).



Section 3.4 shows that one can easily relate the quaternion representation to the Euler axis-and-angle representation, making it intuitive and simple to understand. Unfortunately, the Euler axis-and-angle representation is ambiguous, since a specific rotation can be represented in more than one way. It also suffers from gimbal lock.

Gimbal lock nearly became a troublesome complication when Apollo 13 had control difficulties after their tank rupture on their unsuccessful mission to the moon. Old-style inertial guidance systems used a platform, held in a fixed orientation in three-dimensional (3D) space by gyroscopes, using only three gimbals, the minimum for free movement in any direction. The gimbals

between the platform and the spacecraft enable rapid movement between these two parts. Gimbals are rings of different sizes that are held together concentrically by two pivots per ring, where two subsequent rings' pivots are 90° apart as depicted in Figure 3.1. Gimbal lock is what happens when two of these pivots line up during a sequence of rotations. This means that the number of gimbals are effectively reduced to two which results in a loss of stability. Adding a fourth gimbal overcomes this problem. Gimbal lock is not only a physical phenomenon, but also a mathematical one, causing singularities in the equations of Euler angles. Gimbal angles are the physical manifestation of the mathematical Euler angles. It is not the inability of Euler angles to represent a specific orientation, but rather a discontinuity problem when rotating by small angles across mathematical boundaries.

The quaternion, rotation matrix, Euler axis-and-angle and Euler angles representations of rotations all have three **DOFs**, due to normality constraints, but the quaternion representation is the only one that does not suffer from the mentioned problems.

3.3 Quaternion fundamentals

A quaternion is defined as a four-dimensional vector over the quaternion space \mathcal{Q} . The elements q_0, q_1, q_2 and q_3 in (3.3.1) are real, while the second of the two definitions uses a scalar v and a **3D** vector \mathbf{w}

$$\mathbf{q} = [q_0, q_1, q_2, q_3] \quad (3.3.1)$$

$$= [v, \mathbf{w}]. \quad (3.3.2)$$

Being the most explicit, a quaternion can also be defined in terms of its basis elements as

$$\mathbf{q} = q_0\mathbf{e}_0 + q_1\mathbf{e}_1 + q_2\mathbf{e}_2 + q_3\mathbf{e}_3, \quad (3.3.3)$$

where

$$\mathbf{e}_0 = [1, 0, 0, 0]$$

$$\mathbf{e}_1 = [0, 1, 0, 0]$$

$$\mathbf{e}_2 = [0, 0, 1, 0]$$

$$\mathbf{e}_3 = [0, 0, 0, 1].$$

Given two quaternions $\mathbf{q}_1 = [v_1, \mathbf{w}_1]$ and $\mathbf{q}_2 = [v_2, \mathbf{w}_2]$, we recall the following definitions (Coutsias and Romero, 1999).

Definition 1 *Addition,*

$$\mathbf{q}_1 + \mathbf{q}_2 = [(v_1 + v_2), (\mathbf{w}_1 + \mathbf{w}_2)].$$

Definition 2 *Multiplication,*

$$\mathbf{q}_1 \mathbf{q}_2 = [(v_1 v_2 - \mathbf{w}_1 \cdot \mathbf{w}_2), (v_1 \mathbf{w}_2 + v_2 \mathbf{w}_1 + \mathbf{w}_1 \times \mathbf{w}_2)],$$

where $\mathbf{w}_1 \cdot \mathbf{w}_2$ is the dot product and $\mathbf{w}_1 \times \mathbf{w}_2$ is the cross product of the two 3D vectors.

It should be noted that the multiplication of two quaternions is not commutative, since the cross product is not commutative. Addition of two quaternions, however, is commutative. Also, the associative and distributive laws hold for addition as well as multiplication. Alternatively, the product of two quaternions can be expressed in terms of a matrix-vector multiplication. This is a very useful representation for computational purposes. Manipulating Definition 2 yields

$$\mathbf{q}_1 \mathbf{q}_2 = \mathbf{Q}_1 \mathbf{q}_2 \tag{3.3.4}$$

$$= \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}_1 \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}_2. \tag{3.3.5}$$

Note that the matrix \mathbf{Q}_1 is antisymmetric ($\mathbf{Q}_1^T = -\mathbf{Q}_1$) if \mathbf{q}_1 is a pure quaternion (see Definition 5), which is the case if $q_0 = 0$.

Definition 3 *The conjugate of a quaternion $\mathbf{q} = [v, \mathbf{w}]$,*

$$\mathbf{q}^c = [v, -\mathbf{w}].$$

Definition 4 *A unit quaternion $\mathbf{q} = [v, \mathbf{w}]$,*

$$\mathcal{Q}_1 := \{\mathbf{q} | N(\mathbf{q}) = 1\},$$

where the norm is calculated via the dot product $N(\mathbf{q}) = \mathbf{q} \mathbf{q}^c = \mathbf{q}^c \mathbf{q}$.

Note that the norm $N(\mathbf{q})$ of a quaternion is the square of its euclidean distance

$$N(\mathbf{q}) = q_0^2 + q_1^2 + q_2^2 + q_3^2. \quad (3.3.6)$$

However, the magnitude of a quaternion is defined as its euclidean distance.

Definition 5 A pure quaternion $\mathbf{q} = [v, \mathbf{w}]$, where

$$\mathcal{Q}_0 := \{\mathbf{q} | \mathbf{q} = [0, \mathbf{w}]\}$$

forms the set of all pure quaternions.

A quaternion is pure in the sense that \mathbf{w} forms a 3D vector over \mathbb{R}^3 . In this literature we denote this by using the subscript p , for example, \mathbf{r} is a 3D vector embedded in a quaternion \mathbf{r}_p , where $\mathbf{r}_p = [0, \mathbf{r}]$.

Definition 6 The reciprocal of a quaternion $\mathbf{q} = [v, \mathbf{w}]$,

$$\mathbf{q}^{-1} = \frac{[v, -\mathbf{w}]}{N(\mathbf{q})}.$$

From this it follows that the product $\mathbf{q}\mathbf{q}^{-1} = \mathbf{q}^{-1}\mathbf{q} = [1, 0, 0, 0]$.

3.4 Rotations using quaternions

Attention is now given to the use of quaternions as related to the SfM problem. The equations for the SfM models are in Section 5.1 derived in terms of a rotation matrix. On the other hand, in Section 5.2 we model the rotation with a quaternion. Thus, we need to define this relationship. To rotate a 3D vector \mathbf{r} , using a quaternion \mathbf{q} , we first write it as a pure quaternion $\mathbf{r}_p = [0, \mathbf{r}]$ and form the expression

$$\mathbf{r}'_p = \mathbf{q}\mathbf{r}_p\mathbf{q}^c, \quad (3.4.1)$$

where \mathbf{q} is a unit quaternion. It follows from (3.4.1) that the length of \mathbf{r}_p remains unaltered. Applying the definitions in Section 3.3, it can be rewritten in terms of a rotation matrix as

$$\mathbf{r}' = \mathbf{R}\mathbf{r}. \quad (3.4.2)$$

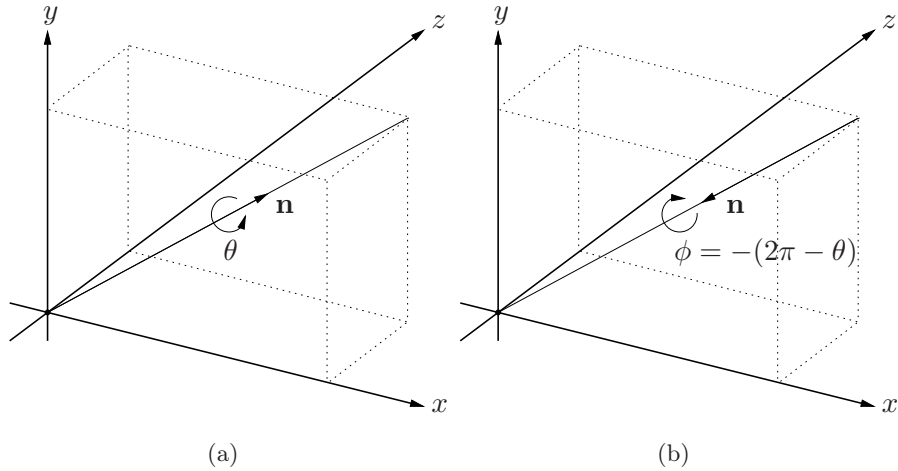


Figure 3.2: A quaternion is unique, except for its sign.

A straight forward, but tedious derivation yields

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_2q_1 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_3q_2 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \quad (3.4.3)$$

As a consequence of the normality constraint placed on \mathbf{q} , such that $N(\mathbf{q}) = 1$, the matrix \mathbf{R} will be an orthonormal matrix

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} = \mathbf{R}^T\mathbf{R}. \quad (3.4.4)$$

Also, a rotation matrix is constrained not to allow a reflection

$$\det(\mathbf{R}) = 1. \quad (3.4.5)$$

There exists a useful relationship between unit quaternions and the Euler axis-and-angle representation, where $\mathbf{n} = [n_x, n_y, n_z]$ is the unit axis of rotation and θ the angle of rotation about \mathbf{n}

$$\mathbf{q} = \pm \left[\cos \frac{\theta}{2}, \mathbf{n} \sin \frac{\theta}{2} \right]. \quad (3.4.6)$$

Humans prefer this representation, since it is easy to visualise. It is important to note that a quaternion representing a specific rotation is unique, except for its sign. Consider a quaternion \mathbf{q} representing a rotation about an axis \mathbf{n} by an angle θ . Fixing \mathbf{n} and rotating in the opposite direction ($\phi = -(2\pi - \theta)$)

result in an equivalent rotation, but a change in the sign of \mathbf{q} . We illustrate it graphically as in Figure 3.2 and summarise it mathematically

$$\begin{aligned}\mathbf{q} &= \left[\cos \frac{\phi}{2}, \mathbf{n} \sin \frac{\phi}{2} \right] \\ &= \left[\cos \frac{-(2\pi-\theta)}{2}, \mathbf{n} \sin \frac{-(2\pi-\theta)}{2} \right] \\ &= \left[\cos(-\pi + \frac{\theta}{2}), \mathbf{n} \sin(-\pi + \frac{\theta}{2}) \right] \\ &= - \left[\cos \frac{\theta}{2}, \mathbf{n} \sin \frac{\theta}{2} \right].\end{aligned}$$

3.5 Derivative of a quaternion

The time-derivate of a quaternion is of utmost importance in this thesis, since it will form part of the model for the motion dynamics of the object being tracked. We base our discussion on the assumptions made by [Wertz \(1986\)](#).

Let the quaternions $\mathbf{q}(t)$ and $\mathbf{q}''(t + \Delta t)$ represent the orientation of a rigid body with respect to a reference system at subsequent time-steps. Let $\mathbf{q}'(\Delta t)$ be the quaternion that relates $\mathbf{q}(t)$ to $\mathbf{q}''(t + \Delta t)$

$$\mathbf{q}''(t + \Delta t) = \mathbf{q}'(\Delta t)\mathbf{q}(t). \quad (3.5.1)$$

We express $\mathbf{q}'(\Delta t)$ as an Euler axis-and-angle—recall (3.4.6)

$$\mathbf{q}'(\Delta t) = \left[\cos \frac{\Delta\theta}{2}, \mathbf{n} \sin \frac{\Delta\theta}{2} \right]. \quad (3.5.2)$$

The change in \mathbf{n} over the time interval Δt leads to second order terms that can be ignored—the error caused by this assumption is discussed in [Wertz \(1986\)](#). We rewrite (3.5.1) in terms of the Euler axis-and-angle, using the fact that a quaternion product can be written as a matrix-vector multiplication as in (3.3.4)

$$\mathbf{q}''(t + \Delta t) = \left\{ \cos\left(\frac{\Delta\theta}{2}\right)\mathbf{I} + \sin\left(\frac{\Delta\theta}{2}\right)\mathbf{N} \right\} \mathbf{q}(t), \quad (3.5.3)$$

where

$$\mathbf{N} = \begin{bmatrix} 0 & -n_x & -n_y & -n_z \\ n_x & 0 & -n_z & n_y \\ n_y & n_z & 0 & -n_x \\ n_z & -n_y & n_x & 0 \end{bmatrix} \quad (3.5.4)$$

is an antisymmetric matrix. In the case where Δt is infinitesimal, we define $\Delta\theta = \omega\Delta t$, where ω is the magnitude of the instantaneous angular velocity of the rigid body. The angular velocity vector is given by

$$\boldsymbol{\omega} = \omega \mathbf{n} \quad (3.5.5)$$

$$= [\omega_x, \omega_y, \omega_z]. \quad (3.5.6)$$

To simplify (3.5.3), we use the following small angle approximations

$$\cos \frac{\Delta\theta}{2} \approx 1 \quad (3.5.7)$$

and

$$\sin \frac{\Delta\theta}{2} \approx \frac{1}{2}\omega\Delta t. \quad (3.5.8)$$

These substitutions yield the approximation

$$\mathbf{q}(t + \Delta t) \approx \left\{ \mathbf{I} + \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\Delta t \right\} \mathbf{q}(t), \quad (3.5.9)$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & -\omega_z & \omega_y \\ \omega_y & \omega_z & 0 & -\omega_x \\ \omega_z & -\omega_y & \omega_x & 0 \end{bmatrix} \quad (3.5.10)$$

too is an antisymmetric matrix. This can be rewritten as the definition for a derivative as

$$\frac{d}{dt}\mathbf{q}(t) \equiv \lim_{\Delta t \rightarrow 0} \frac{\mathbf{q}(t + \Delta t) - \mathbf{q}(t)}{\Delta t}. \quad (3.5.11)$$

Finally, from (3.5.9) and (3.5.12) follows

$$\dot{\mathbf{q}}(t) = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q}(t). \quad (3.5.12)$$

This equation represents a homogeneous system of linear first order differential equations. We can solve it exactly if the angular velocity is constant and for a given initial condition $\mathbf{q}_0 = \mathbf{q}(t_0)$ as

$$\mathbf{q}(t) = e^{\frac{1}{2}(t-t_0)\boldsymbol{\Omega}(\boldsymbol{\omega})}\mathbf{q}_0. \quad (3.5.13)$$

The matrix exponential for an arbitrary matrix \mathbf{A} is defined as

$$e^{t\mathbf{A}} = \sum_{n=0}^{\infty} \frac{(t\mathbf{A})^n}{n!} \quad (3.5.14)$$

$$= \mathbf{I} + t\mathbf{A} + \frac{t^2}{2}\mathbf{A}^2 + \frac{t^3}{3!}\mathbf{A}^3 + \dots \quad (3.5.15)$$

This concludes the derivation of the time-derivative of a time-dependent quaternion under the assumption that the angular velocity is constant.

3.6 Summary

We have shown in this chapter that quaternions provide a simple and efficient way to represent **3D** rotations as a hypersphere in four dimensions. The conversion from a rotation matrix to a quaternion and vice versa is an easy procedure. We have also shown how to calculate the time-derivative of a quaternion as applicable to this thesis. Quaternions prove to be very useful in **SfM** applications.



Chapter 4

The Kalman Filter paradigm

Declare the past, diagnose the present, foretell the future. *Hippocrates*

The Linear Kalman Filter (**LKF**) was introduced by **Kalman (1960)** and aimed at the control systems society. Subsequently, it has evolved into a tool used in many other fields of engineering, for example object tracking, audio restoration and of course Structure from Motion (**SfM**). It is even used in fields other than engineering, such as economics, for making financial predictions (**van der Merwe et al., 2000**). The Kalman Filter (**KF**) is particularly useful when implementing real-time or online systems, since it executes at low computational cost. In this chapter, we give an overview of the **LKF** and two of its nonlinear extensions, the Extended Kalman Filter (**EKF**) and the Unscented Kalman Filter (**UKF**). The **UKF** is the preferred method in this thesis.

4.1 Evolution of the Kalman Filter

In this section, we track the development of the **KF**, but let us first define what is meant by filtering: filtering is the process of estimating the internal state of a dynamic system, given a set of past and present observations in the presence of noise — **KFs** are Minimum Mean-Square Error (**MMSE**) estimators. The Probability Density Functions (**PDFs**) involved are generally modelled as Gaussian distributions, although it is possible to use distributions other than

these. Underlying to the **KF** is the first order Markov assumption, which states that the current state only depends on the previous one

$$p(\mathbf{x}_i | \mathbf{x}_{i-1}). \quad (4.1.1)$$

Similarly, the current observation depends only on the current state

$$p(\mathbf{y}_i | \mathbf{x}_i). \quad (4.1.2)$$

We use the notation

$$\bar{\mathbf{x}}_{i|i-1} = E(\mathbf{x}_i | \mathbf{y}_1, \dots, \mathbf{y}_{i-1}) \quad (4.1.3)$$

to indicate that the predicted state mean $\bar{\mathbf{x}}_{i|i-1}$ is the expected value of \mathbf{x}_i , given the observations $\mathbf{y}_1, \dots, \mathbf{y}_{i-1}$. Similarly, the notation

$$\bar{\mathbf{y}}_{i|i-1} = E(\mathbf{y}_i | \mathbf{y}_1, \dots, \mathbf{y}_{i-1}) \quad (4.1.4)$$

indicates that the predicted observation mean $\bar{\mathbf{y}}_{i|i-1}$ is the expected value of \mathbf{y}_i , given the observations $\mathbf{y}_1, \dots, \mathbf{y}_{i-1}$.

4.1.1 Linear Kalman Filter

The **LKF**, being the simplest of all, is not commonly used in practice, since real-world problems are seldom linear. However, there are exceptions and it does give us insight on the basic steps involved. Based on the assumptions (4.1.1) and (4.1.2), we define the linear state transition equation

$$\mathbf{x}_i = \mathbf{F}_{i-1}\mathbf{x}_{i-1} + \mathbf{B}_{i-1}\mathbf{u}_{i-1} + \boldsymbol{\mu}_{i-1} \quad (4.1.5)$$

and the linear observation transformation

$$\mathbf{y}_i = \mathbf{h}_i\mathbf{x}_i + \boldsymbol{\nu}_i, \quad (4.1.6)$$

where i is the current time-step. \mathbf{F}_{i-1} , \mathbf{B}_{i-1} and \mathbf{H}_{i-1} are matrices and \mathbf{u}_{i-1} is an external control vector. The noise vectors $\boldsymbol{\mu}_{i-1}$ and $\boldsymbol{\nu}_i$ have the following properties:

- Gaussian distributed
- white ($E(\boldsymbol{\mu}_k\boldsymbol{\mu}_l^T) = 0$ and $E(\boldsymbol{\nu}_k\boldsymbol{\nu}_l^T) = 0 \forall k \neq l$)

State transition equation

$$\mathbf{x}_i = \mathbf{F}_{i-1}\mathbf{x}_{i-1} + \boldsymbol{\mu}_{i-1}$$

Observation transform

$$\mathbf{y}_i = \mathbf{h}_i\mathbf{x}_i + \boldsymbol{\nu}_i$$

Predict state

$$\begin{aligned}\bar{\mathbf{x}}_{i|i-1} &= \mathbf{F}_i\bar{\mathbf{x}}_{i-1|i-1} \\ \mathbf{P}_{\mathbf{x}_i|i-1} &= \mathbf{F}_i\mathbf{P}_{\mathbf{x}_{i-1}|i-1}\mathbf{F}_i^T + \mathbf{Q}_{i-1}\end{aligned}$$

Predict observation

$$\begin{aligned}\bar{\mathbf{y}}_{i|i-1} &= \mathbf{H}_i\bar{\mathbf{x}}_{i|i-1} \\ \mathbf{P}_{\mathbf{y}_i|i-1} &= \mathbf{H}_i\mathbf{P}_{\mathbf{x}_i|i-1}\mathbf{H}_i^T + \mathbf{R}_i\end{aligned}$$

Kalman gain

$$\mathbf{K}_i = \mathbf{P}_{\mathbf{x}_i|i-1}\mathbf{H}_i^T\mathbf{P}_{\mathbf{y}_i|i-1}^{-1}$$

Corrected state

$$\begin{aligned}\bar{\mathbf{x}}_{i|i} &= \bar{\mathbf{x}}_{i|i-1} + \mathbf{K}_i(\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_{i|i-1}) \\ \mathbf{P}_{\mathbf{x}_i|i} &= \mathbf{P}_{\mathbf{x}_i|i-1} - \mathbf{K}_i\mathbf{H}_i\mathbf{K}_i^T\end{aligned}$$

Table 4.1: Summary of the LKF's equations.

- additive
- zero-mean ($E(\boldsymbol{\mu}) = 0$ and $E(\boldsymbol{\nu}) = 0$)
- uncorrelated ($E(\boldsymbol{\mu}\boldsymbol{\nu}^T) = 0$).

We use the notation

$$\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \quad (4.1.7)$$

and

$$\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \quad (4.1.8)$$

to indicate that $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are drawn from a zero-mean Gaussian PDF \mathcal{N} with respective covariance matrices \mathbf{Q} and \mathbf{R} . Typically, a single iteration of the filter is as follows:

- predict the state $\bar{\mathbf{x}}_{i|i-1}$, given the best previous state estimate $\bar{\mathbf{x}}_{i-1|i-1}$
- predict the observation $\bar{\mathbf{y}}_{i|i-1}$, given the predicted state $\bar{\mathbf{x}}_{i|i-1}$
- calculate the Kalman gain \mathbf{K}
- correct the predicted state from the difference between the actual observation $\bar{\mathbf{y}}_i$ and the predicted observation $\bar{\mathbf{y}}_{i|i-1}$ to obtain the best possible estimate of the state $\bar{\mathbf{x}}_{i|i}$ (Schwardt, 2003).

The state and observation can each be fully described by its mean vector and covariance matrix if unimodal Gaussian PDFs are assumed. The advantage of using Gaussian PDFs is that it remains Gaussian after a linear transformation. If the actual distribution is unknown, a Gaussian PDF is the safest bet, because it has the highest entropy. The state and observation uncertainties are described by their respective covariance matrices \mathbf{Q}_{i-1} and \mathbf{R}_i as indicated in Table 4.1. We have dropped the control terms from the equation summaries, since it is not applicable to this thesis. For more details on the LKF, see for example Sherlock & Herbst (2003) and Schwardt (2003).

4.1.2 Extended Kalman filter

The EKF is a natural extension of the LKF to nonlinear systems. The state transition equation in nonlinear form now becomes

$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) + \boldsymbol{\mu}_{i-1} \quad (4.1.9)$$

State transition equation

$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1}) + \boldsymbol{\mu}_{i-1}$$

Observation transform

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) + \boldsymbol{\nu}_i.$$

Predict state

$$\begin{aligned}\bar{\mathbf{x}}_{i|i-1} &= \mathbf{f}(\bar{\mathbf{x}}_{i-1|i-1}) \\ \mathbf{P}_{\mathbf{x}_i|i-1} &= \mathbf{F}_{i|i-1} \mathbf{P}_{\mathbf{x}_{i-1}|i-1} \mathbf{F}_{i|i-1}^T + \mathbf{Q}_{i-1} \\ \mathbf{F}_{i|i-1} &= \left. \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}} = \bar{\mathbf{x}}_{i|i-1}}\end{aligned}$$

Predict observation

$$\begin{aligned}\bar{\mathbf{y}}_{i|i-1} &= \mathbf{h}(\bar{\mathbf{x}}_{i|i-1}) \\ \mathbf{P}_{\mathbf{y}_i|i-1} &= \mathbf{H}_{i|i-1} \mathbf{P}_{\mathbf{x}_i|i-1} \mathbf{H}_{i|i-1}^T + \mathbf{R}_i \\ \mathbf{H}_{i|i-1} &= \left. \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}} = \bar{\mathbf{x}}_{i|i-1}}\end{aligned}$$

Kalman gain

$$\mathbf{K}_i = \mathbf{P}_{\mathbf{x}_i|i-1} \mathbf{H}_{i|i-1}^T \mathbf{P}_{\mathbf{y}_i|i-1}^{-1}$$

Corrected state

$$\begin{aligned}\bar{\mathbf{x}}_{i|i} &= \bar{\mathbf{x}}_{i|i-1} + \mathbf{K}_i (\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_{i|i-1}) \\ \mathbf{P}_{\mathbf{x}_i|i} &= (\mathbf{I} - \mathbf{K}_i \mathbf{H}_{i|i-1}) \mathbf{P}_{\mathbf{x}_i|i-1}\end{aligned}$$

Table 4.2: Summary of the EKF's equations.

and the observation transformation

$$\mathbf{y}_i = \mathbf{h}(\mathbf{x}_i) + \boldsymbol{\nu}_i. \quad (4.1.10)$$

The **EKF** uses a first order Taylor series expansion to approximate respectively the system and observation models, evaluated at the point of interest — this enables us to proceed as in the linear case. It seems more accurate to refer to the **EKF** as a first order **EKF**, since higher orders are indeed possible, although hardly if ever used. A Taylor series expansion yields

$$\mathbf{x}_i = \mathbf{f}(\mathbf{x}_{i-1|i-1}) + \mathbf{f}'(\mathbf{x}_{i-1|i-1})(\mathbf{x}_{i-1} - \mathbf{x}_{i-1|i-1}) + \text{HOTs} + \boldsymbol{\mu}_{i-1}, \quad (4.1.11)$$

where

$$\mathbf{f}'(\mathbf{x}_{i-1|i-1}) = \left. \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_{i-1|i-1}}. \quad (4.1.12)$$

The same procedure holds for (4.1.10). The Higher Order Terms (**HOTs**) may be expanded to obtain a higher order **EKF**. The linearisation, as applicable to the **KF** equations, is achieved by calculating the Jacobian matrices of the models, as shown in Table 4.2. For more details, see [Sherlock & Herbst \(2003\)](#) and [Morrell \(2003\)](#), among other's.

4.1.3 Unscented Kalman Filter

We now come to the focus of this chapter. The **UKF** was first introduced by [Julier & Uhlmann \(1996\)](#) to address the shortcomings of the **EKF**. Firstly, if the system and observation models are highly nonlinear, the **EKF**'s linearisation fails. The **UKF** does not approximate the nonlinear models, but rather approximates their distributions using the nonlinear models itself. Figure 4.1 shows a simple two-dimensional (**2D**) example, comparing the predictions of the **EKF** and the **UKF**. The **EKF** predicts the state along the tangent to the curve, resulting in an erroneous mean and covariance. To compensate for this, the covariance needs to be adjusted to an unrealistic extent. The **UKF**, using its sigma points, correctly predicts the state mean and covariance. Secondly, the **EKF** is capable of modelling only the first and second moments of a Gaussian **PDF**, whereas the **UKF** is capable of up to fourth moment modelling.

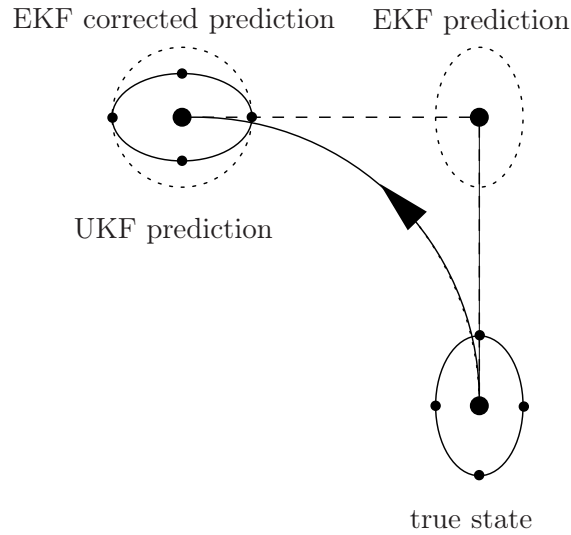


Figure 4.1: The UKF correctly predicts the state, whereas the EKF needs additional correction.

$$\begin{aligned}
 w_0^{(m)} &= \frac{\lambda}{(n + \lambda)} \\
 w_0^{(c)} &= \frac{\lambda}{(n + \lambda)} + (1 - \alpha^2 + \beta) \\
 w_j^{(m)} &= w_j^{(c)} = \frac{1}{2(n + \lambda)}, \quad j = 1..2n \\
 \lambda &= \alpha^2(n + \kappa) - n
 \end{aligned}$$

Table 4.3: Weights needed by the UKF.

Algorithm

Julier & Uhlmann (1996) claims that their exposition of the UKF is a more direct generalisation of the LKF than the EKF. In particular, they suggest that the new approach is a true extension of the KF paradigm, whereas the so-called EKF is more of a recipe for pounding nonlinear models into linear holes (sic.). The same state transition equation and observation transformation as for the EKF are assumed—see (4.1.9) and (4.1.10).

The algorithm relies on the weights given in Table 4.3 and the filter is ini-

$$\begin{aligned}
\bar{\mathbf{x}}_0 &= E[\mathbf{x}_0] \\
\mathbf{P}_0 &= E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T] \\
\bar{\mathbf{a}}_0 &= E[\mathbf{a}_0] = [\bar{\mathbf{x}}_0^T, \mathbf{0}^T, \mathbf{0}^T]^T \\
\mathbf{A}_0 &= E[(\mathbf{a}_0 - \bar{\mathbf{a}}_0)(\mathbf{a}_0 - \bar{\mathbf{a}}_0)^T] = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}
\end{aligned}$$

Table 4.4: *Initialisation of the UKF.*

tialised using the equations from Table 4.4. The weights are used to calculate the statistics of the state and observation. Table 4.5 shows the filter loop—it involves steps similar to the LKF (Table 4.1) and the EKF (Table 4.2). Augmented vectors and matrices are used for ease of computation. The dimension of the augmented state vector as the statistical mean of the sigma points $\bar{\mathbf{a}}$ is

$$n = n_{\mathbf{x}} + n_{\boldsymbol{\mu}} + n_{\boldsymbol{\nu}}, \quad (4.1.13)$$

where $n_{\mathbf{x}}$, $n_{\boldsymbol{\mu}}$ and $n_{\boldsymbol{\nu}}$ denote the dimensions of the vectors \mathbf{x} , $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. Thus, the augmented state covariance \mathbf{A} has dimensions $n \times n$, where $n_{\mathbf{x}} \times n_{\mathbf{x}}$, $n_{\boldsymbol{\mu}} \times n_{\boldsymbol{\mu}}$ and $n_{\boldsymbol{\nu}} \times n_{\boldsymbol{\nu}}$ denote the dimensions of the matrices \mathbf{P} , \mathbf{Q} and \mathbf{R} . The sigma points are formed by first calculating a matrix square root of the scaled augmented state covariance \mathbf{A} —more details on the actual computation follows in the next section. These values are assembled into an augmented sigma point matrix \mathcal{A} , where its first column is the augmented state mean $\bar{\mathbf{a}}$. The next n columns are formed by adding $\bar{\mathbf{a}}$ to each column of the calculated matrix square root and the last n columns by subtracting $\bar{\mathbf{a}}$. The final structure has a dimension of $n \times (2n + 1)$, where $2n + 1$ denotes the number of sigma points. The augmented sigma point matrix \mathcal{A} is related to the sigma point state matrix $\boldsymbol{\mathcal{X}}$, the sigma point state uncertainty matrix $\boldsymbol{\mathcal{V}}$ and the sigma point observation uncertainty matrix $\boldsymbol{\mathcal{N}}$ via

$$\mathcal{A} = \begin{bmatrix} \boldsymbol{\mathcal{X}} \\ \boldsymbol{\mathcal{V}} \\ \boldsymbol{\mathcal{N}} \end{bmatrix} \quad (4.1.14)$$

Augmented structures

$$\bar{\mathbf{a}} = \begin{bmatrix} \bar{\mathbf{x}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \mathbf{A} = \begin{bmatrix} \mathbf{P} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}, \mathcal{A} = \begin{bmatrix} \boldsymbol{\chi} \\ \boldsymbol{\nu} \\ \mathcal{N} \end{bmatrix}$$

Calculate sigma points

$$\mathcal{A}_{i-1|i-1} = [\bar{\mathbf{a}}_{i-1|i-1}, \bar{\mathbf{a}}_{i-1|i-1} \pm \sqrt{(n + \lambda)\mathbf{A}_{i-1|i-1}}]$$

Predict state

$$\begin{aligned} \boldsymbol{\chi}_{i|i-1} &= \mathbf{f}(\boldsymbol{\chi}_{i-1|i-1}, \boldsymbol{\nu}_{i-1}) \\ \bar{\mathbf{x}}_{i|i-1} &= \sum_{j=0}^{2n} w_j^{(m)} \boldsymbol{\chi}_{j,i|i-1} \\ \mathbf{P}_{\mathbf{x}_{i|i-1}} &= \sum_{j=0}^{2n} w_j^{(c)} [\boldsymbol{\chi}_{j,i|i-1} - \bar{\mathbf{x}}_{i|i-1}][\boldsymbol{\chi}_{j,i|i-1} - \bar{\mathbf{x}}_{i|i-1}]^T \end{aligned}$$

Predict observation

$$\begin{aligned} \boldsymbol{\nu}_{i|i-1} &= \mathbf{h}(\boldsymbol{\chi}_{i|i-1}, \mathcal{N}_{i-1}) \\ \bar{\mathbf{y}}_{i|i-1} &= \sum_{j=0}^{2n} w_j^{(m)} \boldsymbol{\nu}_{j,i|i-1} \\ \mathbf{P}_{\mathbf{y}_{i|i-1}} &= \sum_{j=0}^{2n} w_j^{(c)} [\boldsymbol{\nu}_{j,i|i-1} - \bar{\mathbf{y}}_{i|i-1}][\boldsymbol{\nu}_{j,i|i-1} - \bar{\mathbf{y}}_{i|i-1}]^T \end{aligned}$$

Kalman gain

$$\begin{aligned} \mathbf{P}_{\mathbf{x}_{i|i-1} \mathbf{y}_{i|i-1}} &= \sum_{j=0}^{2n} w_j^{(c)} [\boldsymbol{\chi}_{j,i|i-1} - \bar{\mathbf{x}}_{i|i-1}][\boldsymbol{\nu}_{j,i|i-1} - \bar{\mathbf{y}}_{i|i-1}]^T \\ \mathbf{K}_{i|i-1} &= \mathbf{P}_{\mathbf{x}_{i|i-1} \mathbf{y}_{i|i-1}} \mathbf{P}_{\mathbf{y}_{i|i-1}}^{-1} \end{aligned}$$

Corrected state

$$\begin{aligned} \bar{\mathbf{x}}_{i|i} &= \bar{\mathbf{x}}_{i|i-1} + \mathbf{K}_{i|i-1}(\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_{i|i-1}) \\ \mathbf{P}_{\mathbf{x}_{i|i}} &= \mathbf{P}_{i|i-1} - \mathbf{K}_{i|i-1} \mathbf{P}_{\mathbf{y}_{i|i-1}} \mathbf{K}_{i|i-1}^T \end{aligned}$$

Table 4.5: Summary of the UKF's equations.

as needed by the nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$.

Scaled Unscented Transform

The **UKF** relies on what is referred to as sigma points. It is by definition the minimum number of deterministic points to fully represent a given **PDF**. In general, $2n + 1$ sigma points are needed for a distribution of dimension n . One way to find these points for a Gaussian **PDF** is to calculate the square root of its covariance matrix, using Choleski decomposition. It is numerically stable and efficient. The Choleski decomposition of an $n \times n$ matrix \mathbf{A} is defined as

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T, \quad (4.1.15)$$

where

$$\text{chol}(\mathbf{A}) = \mathbf{L} \quad (4.1.16)$$

and \mathbf{L} is a lower triangular matrix. Each column of \mathbf{L} corresponds to one n -dimensional principal point along one principal axis. These principal points are assembled into a matrix \mathbf{A} to form a symmetrical set of sigma points as explained in the previous section.

The Unscented Transform (**UT**) is a method for calculating the statistics of a nonlinearly transformed Random Variable (**RV**) or, as in this case, the statistics of the state and observation. It was originally proposed by **Julier & Uhlmann (1996)**, but modified by (**Julier, 1999**) to address the problem of positive semi-definiteness of the covariance matrices and to include prior knowledge about the **PDF**. **Julier (2002)** also proposed a method to calculate a reduced set of sigma points, referred to as simplex sigma points. This approach uses only $n + 1$ asymmetrically distributed sigma points. The downside to this is the small bias and covariance error it introduces. In this thesis, we use the Scaled Unscented Transform (**SUT**) with $2n + 1$ sigma points to fully represent a Gaussian **PDF**.

The **SUT** has three parameters to manipulate the posterior **PDF** (**van der Merwe et al., 2000**). The prior **PDF** is defined by the set of sigma points from which the posterior **PDF** is calculated via appropriate weighting (Table 4.3). The parameter α is constrained by $0 \leq \alpha \leq 1$ and it determines the spread of the sigma points around the mean. It is set to $\alpha = 1$ by default and only used

if the nonlinearities are strong. The parameter β , where $\beta \geq 0$, influences the higher order moments by adding more weight to the mean via $w_0^{(c)}$ if such knowledge is available. This parameter can also be used to control the heaviness of the tails of the posterior distribution. For a Gaussian PDF, the choice $\beta = 2$ is recommended (Julier and Uhlmann, 1996; van der Merwe *et al.*, 2000). To guarantee positive semi-definiteness of the covariance matrices, set the secondary scaling parameter $\kappa \geq 0$. We set $\kappa = 0$ by default.

Computational complexity is an important consideration when implementing online algorithms. How does the EKF compare to the UKF in this respect? According to van der Merwe & Wan (2001) and van der Merwe & Wan (2003), both the EKF and UKF run at a computational complexity of $O(n^3)$. They state that implementations of the UKF with a complexity of $O(n^2)$ are possible, but that applies only to parameter estimation and not to state estimation, which we are dealing with. Parameter estimation refers to methods using the UKF to train an Artificial Neural Network (ANN), for example. Determining the computational complexity of an algorithm is in reality a much more complicated process, where atomic operations must be defined in order to make a fair comparison.

We compare the statistical accuracy of the SUT to the linearisation method of the EKF by using an example from Julier & Uhlmann (1996). Let us assume a simple one-dimensional nonlinear transformation

$$y = x^2, \tag{4.1.17}$$

where x and y are RVs. We define x_{true} , a Gaussian RV, as the sum of its mean \bar{x} and w , a zero-mean Gaussian RV with variance σ_x^2 , as

$$x_{true} = \bar{x} + w. \tag{4.1.18}$$

We transform x_{true} with (4.1.17), to obtain

$$y_{true} = \bar{x}^2 + 2\bar{x}w + w^2. \tag{4.1.19}$$

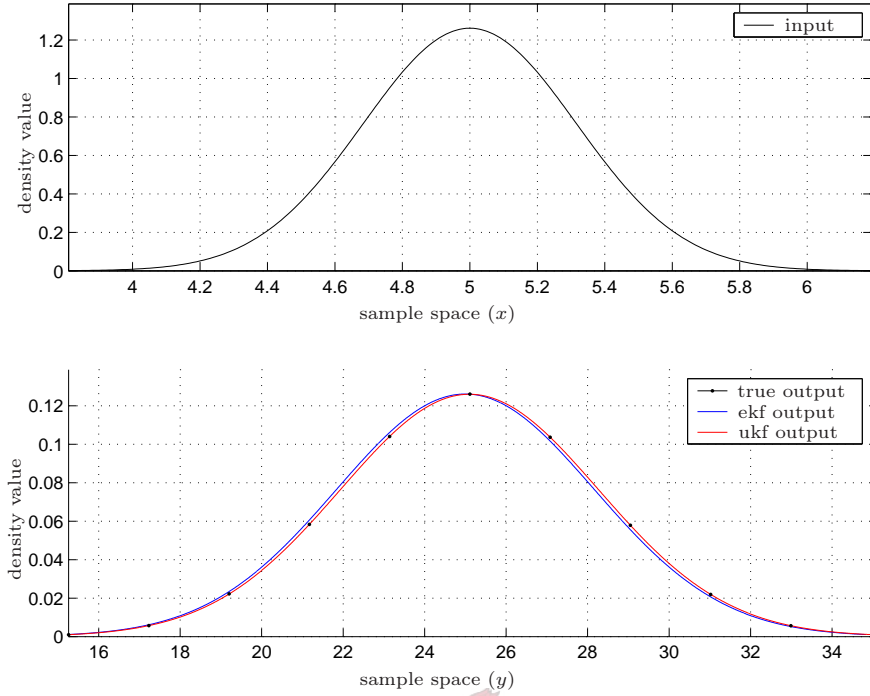


Figure 4.2: The EKF's and UKF's output states vs. the true output state for the input state with mean $\bar{x} = 5$ and variance $\sigma_x^2 = 0.1$.

The true transformed mean and variance are calculated by taking expectations:

$$\bar{y}_{true} = E(y_{true}) \quad (4.1.20)$$

$$= \bar{x}^2 + \sigma_x^2 \quad (4.1.21)$$

$$\sigma_{y_{true}}^2 = E((y_{true} - \bar{y}_{true})^2) \quad (4.1.22)$$

$$= 2\sigma_x^4 + 4\bar{x}^2\sigma_x^4. \quad (4.1.23)$$

If the statistics are calculated using the linearisation method of the EKF as in Table 4.2, we obtain the transformed mean

$$\bar{y}_{ekf} = \bar{x}^2 \quad (4.1.24)$$

and the transformed variance

$$\sigma_{y_{ekf}}^2 = (2\bar{x})(\sigma_x^2)(2\bar{x}) \quad (4.1.25)$$

$$= 4\bar{x}^2\sigma_x^4, \quad (4.1.26)$$

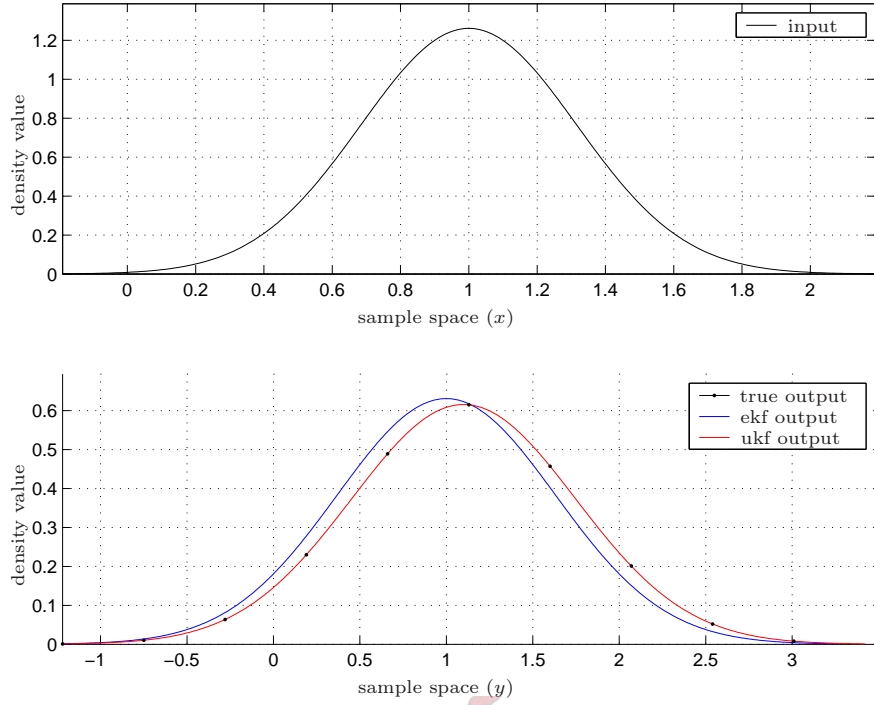


Figure 4.3: The EKF's and UKF's output states vs. the true output state for the input state with mean $\bar{x} = 1$ and variance $\sigma_x^2 = 0.1$.

where the nonlinear transformation is linearised as

$$\left. \frac{dy}{dx} \right|_{x=\bar{x}} = 2\bar{x}. \quad (4.1.27)$$

We now use the **SUT** as used by the **UKF** to calculate the same statistics. We take $\alpha = 1$, $\beta = 0$ and $\kappa = 2$, which yields $\lambda = 2$ for a dimension of $n = 1$. From this, we obtain the weights, using Table 4.3, as

$$w_0^{(m)} = w_0^{(c)} = \frac{2}{(1+2)} \quad (4.1.28)$$

and

$$w_{1,2}^{(m)} = w_{1,2}^{(c)} = \frac{1}{2(1+2)}. \quad (4.1.29)$$

The three sigma points are calculated as

$$\mathcal{X} = [\bar{x}, \bar{x} + \sqrt{(1+\lambda)\sigma_x^2}, \bar{x} - \sqrt{(1+\lambda)\sigma_x^2}]. \quad (4.1.30)$$

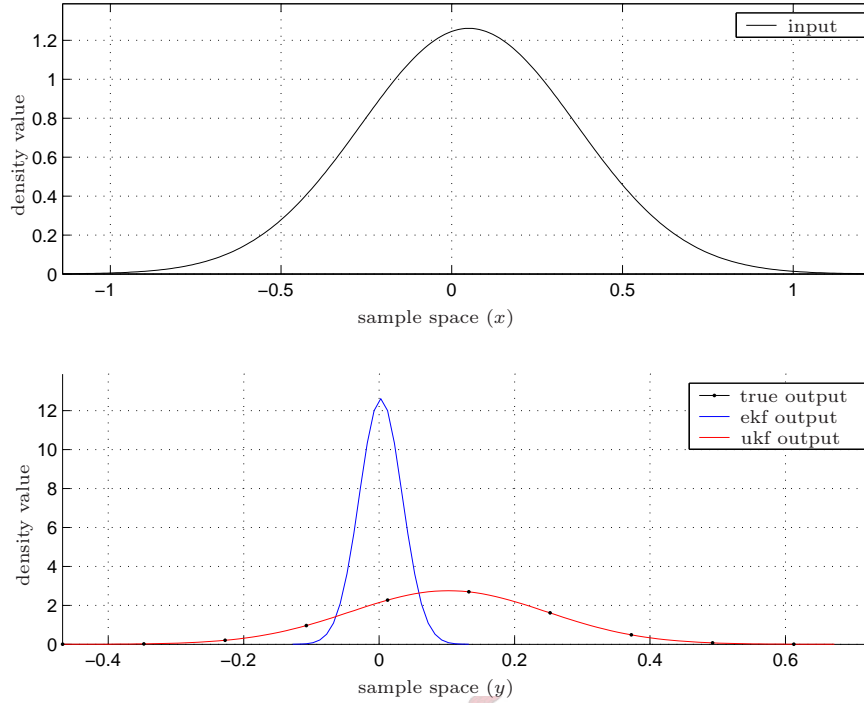


Figure 4.4: The EKF's and UKF's output states vs. the true output state for the input state with mean $\bar{x} = 0.05$ and variance $\sigma_x^2 = 0.1$.

Using all of these, we obtain the transformed mean and variance:

$$\bar{y}_{ukf} = \bar{x}^2 + \sigma_x^2 \quad (4.1.31)$$

$$\sigma_{y_{ukf}}^2 = 2\sigma_x^4 + 4\bar{x}^2\sigma_x^4. \quad (4.1.32)$$

It is evident from these equations that the **SUT**'s results are identical to that of the true mean and variance, while the linearisation method introduces errors in both the mean and variance. We include a few figures which illustrate the effect of these errors on the given input **PDF**. In Figure 4.2 the linearisation method's **PDF** is very close to that of the true one. In Figure 4.3 the error is more prominent and in Figure 4.4 it is significant. When $\bar{x} = 0$, the linearisation method's **PDF** becomes an impulse, because its transformed variance $\sigma_{y_{ekf}}^2 = 0$, due to its dependence on the mean. The **SUT**, which yields results identical to the true output **PDF**, is clearly superior to the linearisation method. As a final remark: the **SUT** is not influenced negatively by discontinuous or piecewise continuous functions, since the sigma points are propagated

Property	EKF	UKF
Method	linearisation	sigma points
Modelling	first order Taylor series	original nonlinear models
Complexity	$O(n^3)$	$O(n^3)$
Statistical accuracy	second moment	up to fourth moment

Table 4.6: Comparison of the EKF to the UKF.

directly. On the other hand, the linearisation method is, since the first derivative is undefined at discontinuous or sharp points. The reader is referred to Appendix A of [Julier & Uhlmann \(1996\)](#) for a proof and full discussion.

4.2 Summary

The EKF's inability to correctly predict the state and observation of nonlinear models was exploited in this chapter. It is clearly an inferior choice when compared to the UKF. The UKF's strength lies in the propagation of its sigma points using the nonlinear models itself. Up to fourth moment information of the PDFs can be incorporated by adjusting the parameters α , β and κ . The comparison of the EKF to the UKF is summarised in Table 4.6.



Chapter 5

Structure from Motion

The best material model of a cat is another, or preferably the same, cat.

A. Rosenblueth and Norbert Wiener

Structure from Motion (**SfM**) is a method to reconstruct an object or scene's three-dimensional (**3D**) structure from its observed two-dimensional (**2D**) motion. The performance of **SfM** strongly depends on the state and observation modelling. The state transition model predicts the new state in terms of its previous state and the observation model transforms this new state into a predicted observation. The Unscented Kalman Filter (**UKF**) acts as Minimum Mean-Square Error (**MMSE**) agent, minimising the error between the real observation and the predicted observation by estimating the **3D** structure and motion of the rigid object.

We will discuss our framework based on the concepts used by [Azarbayejani & Pentland \(1995\)](#) and [Venter \(2002\)](#). Our aim is to construct models that are robust, not depending on any prior knowledge about the structure or motion. In the following sections we describe the observation model and structure and motion model, rounding it off with a section about proper initialisation of the **UKF** and the models.

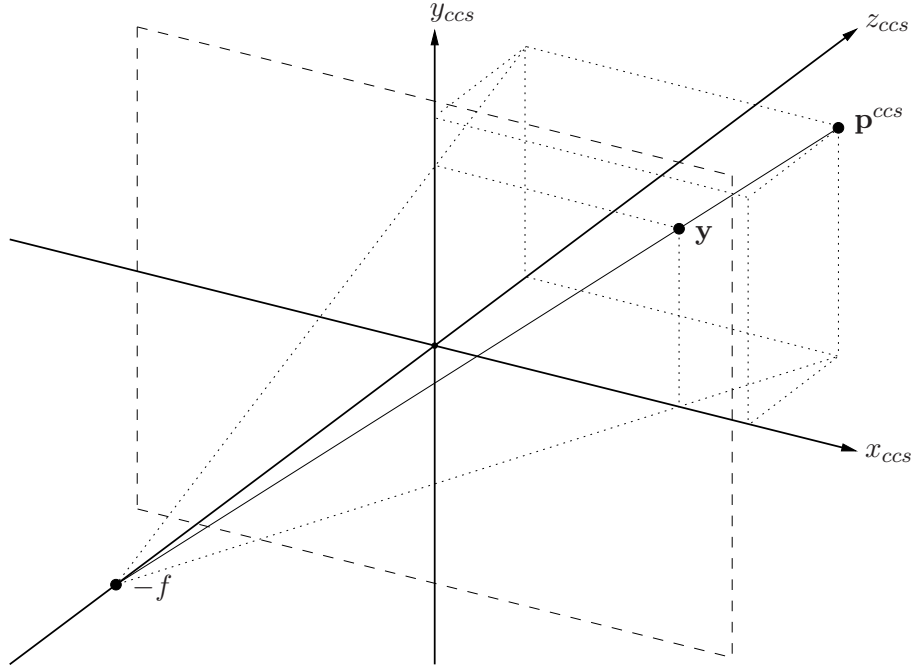


Figure 5.1: Linear perspective projection of a point \mathbf{p} onto the image plane as \mathbf{y} at a focal length of f .

5.1 Observation modelling

We assume a linear perspective projection model for the pinhole camera. Consider a **3D** coordinate in the Camera Coordinate System (**CCS**)

$$\mathbf{p}^{ccs} = \begin{bmatrix} p_x^{ccs} \\ p_y^{ccs} \\ p_z^{ccs} \end{bmatrix} \quad (5.1.1)$$

that undergoes a linear perspective projection. The projected coordinate \mathbf{y} lies on the image plane, where the image plane passes through the origin of the **CCS**—this is illustrated in Figure 5.1. The focus lies at $z_{ccs} = -f$, referred to as the Center Of Projection (**COP**). The mathematical relationship between the **3D** coordinate \mathbf{p}^{ccs} and its projected **2D** coordinate \mathbf{y} is described by

$$\mathbf{y} = \begin{bmatrix} p_x^{ccs} \\ p_y^{ccs} \end{bmatrix} \left(\frac{1}{1 + \frac{p_z^{ccs}}{f}} \right). \quad (5.1.2)$$

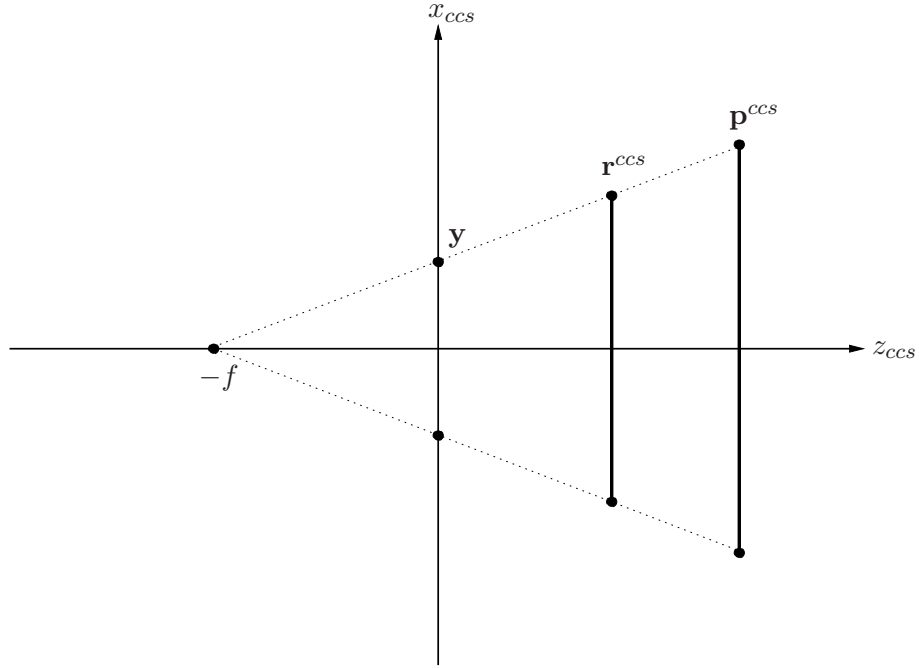


Figure 5.2: The points \mathbf{r}^{ccs} and \mathbf{p}^{ccs} yield the same observation.

Altering f in this model, alters the projected coordinate independent of p_z^{ccs} (Azarbayejani and Pentland, 1995). Note that it is impossible to tell whether we have changed p_z^{ccs} or f when either p_z^{ccs} or f is kept constant — it is possible that different values yield the same ratio $\frac{p_z^{ccs}}{f}$. This model also enables us to express the orthographic camera model as a special case of the perspective camera model as

$$\mathbf{y} = \begin{bmatrix} p_x^{ccs} \\ p_y^{ccs} \end{bmatrix}, \quad (5.1.3)$$

when $f \rightarrow \infty$. We are actually interested in the inverse process of (5.1.2)

$$\mathbf{p}^{ccs} = \begin{bmatrix} y_x \left(1 + \frac{p_z^{ccs}}{f}\right) \\ y_y \left(1 + \frac{p_z^{ccs}}{f}\right) \\ p_z^{ccs} \end{bmatrix}, \quad (5.1.4)$$

where \mathbf{p}^{ccs} is written in terms of its projection \mathbf{y} and the focal length f . From this we see that the only information we need to reconstruct \mathbf{p}^{ccs} up to a scale factor is \mathbf{y} and the ratio $\frac{p_z^{ccs}}{f}$. In this thesis, we assume that the focal length

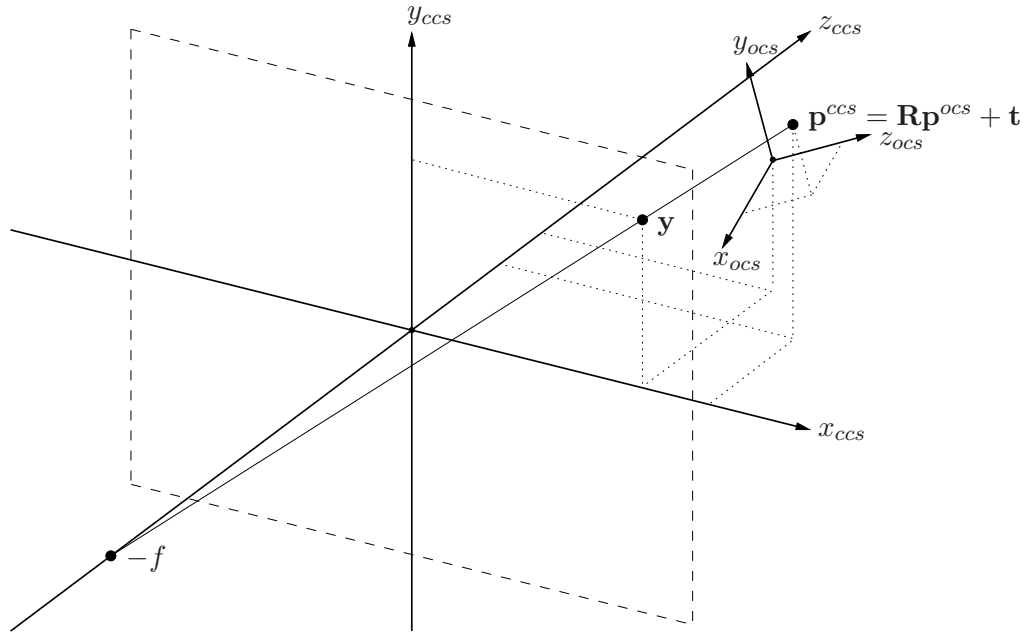


Figure 5.3: Linear perspective projection of a point \mathbf{p} in terms of the OCS onto the image plane as \mathbf{y} in the CCS at a focal length of f .

is known, which leaves us with only one unknown parameter per coordinate— p_z^{ccs} . Compared to previous methods that estimate the 3D point \mathbf{p}^{ccs} as a whole (Broida *et al.*, 1990), this is a reduction of two thirds. This enables us to keep the estimated state’s dimension low, since the dimension influences the computational complexity of the UKF, as stated in Subsection 4.1.3.

The issue of scale is an important one in SfM. Consider a feature \mathbf{p}^{ccs} on an arbitrary object to be a scaled version of the same feature \mathbf{r}^{ccs} on the unscaled object. Figure 5.2 shows that \mathbf{r}^{ccs} yields the same observation as \mathbf{p}^{ccs} . The significance of this is that the structure can only be estimated up to a scale value, but this does not pose a problem. If one feature’s real coordinate is known, the scale factor can be determined and the whole structure scaled accordingly, since it is assumed to be rigid.

Consider now a rigid object in 3D space under linear perspective projection as depicted in Figure 5.3. We refer to the coordinate system attached to the object as the Object Coordinate System (OCS). The structure is estimated relative to the CCS. The two origins of the respective coordinate systems are related via the translation vector \mathbf{t} . The orientation of the OCS relative to the

CCS is related via the rotation matrix \mathbf{R} . We express this mathematically as

$$\mathbf{p}^{ccs} = \mathbf{R}\mathbf{p}^{ocs} + \mathbf{t}. \quad (5.1.5)$$

Again, we rewrite this as the inverse relationship

$$\mathbf{p}^{ocs} = \mathbf{R}^{-1}(\mathbf{p}^{ccs} - \mathbf{t}) \quad (5.1.6)$$

$$= \mathbf{R}^{-1} \begin{bmatrix} y_x(1 + \frac{p_z^{ccs}}{f}) - t_x \\ y_y(1 + \frac{p_z^{ccs}}{f}) - t_y \\ p_z^{ccs} - t_z \end{bmatrix}, \quad (5.1.7)$$

where we substituted (5.1.4). As discussed in Chapter 3, the rotation is represented using quaternions. The solution to the differential equation (3.5.13) implies that we must choose a reference. Thus, we initialise the rotation by aligning the OCS with the CCS

$$\mathbf{R}_0^{-1} = \mathbf{I}. \quad (5.1.8)$$

We define a point-wise feature in terms of its image location in the first frame it appears. The translation components t_x and t_y are initialised with the mean of the initial 2D observation. This is a valid assumption: given an adequate number of features, the mean of the observation tends to coincide with the centroid of the object. The component t_z is set to zero

$$\mathbf{t}_0 = \begin{bmatrix} \bar{y}_{x,0} \\ \bar{y}_{y,0} \\ 0 \end{bmatrix}. \quad (5.1.9)$$

If we incorporate the initial conditions into (5.1.7) and write it in a time-dependent form, we obtain our perspective projection camera model. The equation for a single feature is expressed as

$$\mathbf{p}_{i|i-1}^{ocs} = \begin{bmatrix} y_{x,0}(1 + \frac{p_{z,i|i-1}^{ccs}}{f}) - \bar{y}_{x,0} \\ y_{y,0}(1 + \frac{p_{z,i|i-1}^{ccs}}{f}) - \bar{y}_{y,0} \\ p_{z,i|i-1}^{ccs} \end{bmatrix} \quad (5.1.10)$$

$$= \begin{bmatrix} y_{x,0}(1 + \frac{s_{i|i-1}}{f}) - \bar{y}_{x,0} \\ y_{y,0}(1 + \frac{s_{i|i-1}}{f}) - \bar{y}_{y,0} \\ s_{i|i-1} \end{bmatrix}, \quad (5.1.11)$$

where s is substituted for p_z^{ccs} to simplify the notation and to link it to the state modelling in Section 5.2. Note that (5.1.11) depends on the initial observation \mathbf{y}_0 and its corresponding mean $\bar{\mathbf{y}}_0$ — any errors introduced at initialisation will be propagated to all future estimates. We assume that the observations are corrupted by additive, zero-mean, white, Gaussian noise

$$\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad (5.1.12)$$

where $\mathbf{0}$ denotes a zero-mean and \mathbf{R} the observation covariance matrix.

5.2 Structure and motion modelling

Our goal is to estimate point-wise structure. It is assumed that no prior knowledge about the structure of the tracked object is available. We assume additive, zero-mean, white, Gaussian noise throughout this section to characterise uncertainties, expressed as

$$\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad (5.2.1)$$

where $\mathbf{0}$ denotes a zero-mean and \mathbf{Q} the state covariance matrix. The simplest form of the structure state transition is given by

$$\mathbf{s}_{i|i-1} = \mathbf{s}_{i-1|i-1}, \quad (5.2.2)$$

where i is the current time step. This equation acts on the structure vector

$$\mathbf{s} = [s_0, \dots, s_k, \dots, s_{m-1}]^T. \quad (5.2.3)$$

It contains the estimated structure parameters, where m is the number of features and $s_k = p_{z,k}^{ocs}$, where k is the feature index. This structure transition equation simply states that the current estimate of the structure is similar to the estimate at the previous step. This is a valid approach if the tracked object is moving slowly or the frame-rate is high. This way, we rely on the Kalman gain to adjust the structure — the modelling error is absorbed by its modelling uncertainty $\boldsymbol{\mu}_s$.

We describe the rotation of the object relative to the CCS using quaternions. Quaternions were fully discussed in Chapter 3. In particular, we use (3.5.10) and (3.5.12), which yields

$$\mathbf{q}_{i|i-1} = \mathbf{q}_{i-1|i-1} + \Delta t \frac{1}{2} \boldsymbol{\Omega}(\boldsymbol{\omega}_{i|i-1}) \mathbf{q}_{i-1|i-1}. \quad (5.2.4)$$

We have to incorporate the object's angular velocity, also relative to the **OCS**, into the state, since $\boldsymbol{\Omega}(\boldsymbol{\omega}_{i|i-1})$ is a function of the angular velocity. The angular velocity state transition is modelled as

$$\boldsymbol{\omega}_{i|i-1} = \boldsymbol{\omega}_{i-1|i-1}. \quad (5.2.5)$$

These equations act on the vectors

$$\mathbf{q} = [q_0, q_1, q_2, q_3]^T \quad (5.2.6)$$

and

$$\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T. \quad (5.2.7)$$

The translation transition equations follow a similar procedure. The translation relates the origin of the **OCS** relative to the **CCS**. It is modelled as the translation at the previous step plus the change in translation, denoted by

$$\mathbf{t}_{i|i-1} = \mathbf{t}_{i-1|i-1} + \Delta t \mathbf{d}_{i|i-1}. \quad (5.2.8)$$

Thus, the model for the translation velocity is given by

$$\mathbf{d}_{i|i-1} = \mathbf{d}_{i-1|i-1} \quad (5.2.9)$$

and is related to the translation \mathbf{t} via

$$\mathbf{d} = \frac{d\mathbf{t}}{dt}, \quad (5.2.10)$$

which states that the velocity is the time-derivative of the translation. The vectors on which these equations operate are

$$\mathbf{t} = [t_x, t_y, t_z]^T \quad (5.2.11)$$

and

$$\mathbf{d} = [d_x, d_y, d_z]^T. \quad (5.2.12)$$

It is clear that all these equations are of first order. Using higher order modelling results in higher computational complexity. To conclude, we concatenate the respective state column vectors to construct the full state column vector with dimension $m + 13$

$$\mathbf{x} = [\mathbf{s}^T, \mathbf{q}^T, \boldsymbol{\omega}^T, \mathbf{t}^T, \mathbf{d}^T]^T, \quad (5.2.13)$$

where \mathbf{s} contains the z -coordinates of all features, \mathbf{q} is the rotation of the **OCS** with respect to the **CCS**, $\boldsymbol{\omega}$ is the angular velocity of the **OCS** with respect to the **CCS**, \mathbf{t} is the translation of the **OCS** with respect to the **CCS** and \mathbf{d} is the translation velocity of the **OCS** with respect to the **CCS**.

This modelling approach differs from the one by [Azarbayejani & Pentland \(1995\)](#)—they estimate focal length as an additional parameter, but do not estimate the translation velocity. We opted for a simpler form of the translation transition equations than those proposed by [Venter \(2002\)](#). This trade-off does not seem to influence the performance in a negative way as we will see from the results in Chapter 7.

5.3 Initialisation and setup

The **UKF** strongly depends on suitable initial conditions and model uncertainties—if not set up properly, the algorithm fails to converge. Some of the initialisation choices have been mentioned earlier. Recall that since it is assumed that we have no prior knowledge available about the structure, it is set to a zero-vector

$$\mathbf{s}_0 = \mathbf{0}, \quad (5.3.1)$$

which corresponds to a plane coincident with the image plane. The estimated rotation is only a relative measure of orientation and we chose in Section 5.1 to align the **OCS** with the **CCS**. Thus, the initial quaternion is set to

$$\mathbf{q}_0 = [1, 0, 0, 0], \quad (5.3.2)$$

which corresponds to an identity matrix. Accordingly, the initial angular velocity vector is set to

$$\boldsymbol{\omega}_0 = \mathbf{0}. \quad (5.3.3)$$

We have only partial information available to initialise the translation vector. The components t_x and t_y are initialised as the **2D** mean of the initial set of observations, since the translation of the tracked object is defined by the translation of its centroid

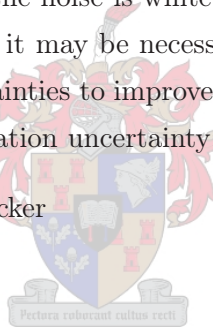
$$\mathbf{t}_0 = \begin{bmatrix} \bar{y}_{x,0} \\ \bar{y}_{y,0} \\ 0 \end{bmatrix}, \quad (5.3.4)$$

where t_z is set to zero. Thus, our initial structure coincides with the image plane with its centroid at the **2D** mean of the initial set of observations. This is also the origin of the **OCS** and the point about which rotation occurs. It must be stressed that if these values do not coincide with the real centroid, due to the distribution of features, it causes difficulties in the estimation of rotation. The effect of a biased centroid will be observed in Subsection 7.2.3. Analogous to the angular velocity, we set the initial translation velocity to

$$\mathbf{d}_0 = \mathbf{0}. \quad (5.3.5)$$

The state covariance matrix \mathbf{P}_0 is initialised with a spherical covariance matrix, containing the same value for each entry on its main diagonal. The matrices \mathbf{Q} and \mathbf{R} are spherical matrices too and the entries on the main diagonal are fixed for a particular sequence. Thus, these matrices are each fully described by a variance. Spherical covariance matrices are an appropriate choice, since it is assumed that the noise is white and uncorrelated—refer to Subsection 4.1.1. In some cases, it may be necessary to differentiate between the structure and motion uncertainties to improve convergence. There are two factors that influence the observation uncertainty:

- accuracy of the feature tracker
- resolution of the frames.



The accuracy of the Kanade-Lucas-Tomasi (**KLT**) feature tracker depends on its setup and the quality of the sequence and therefore may vary from sequence to sequence. We can, however, partially estimate the observation noise from the resolution of the frames for a particular sequence. It should be noted that all frames are scaled by the resolution height h and width w to create a normalised frame with unity width and height. Thus, if we assume one pixel deviation, the observation noise variance can be based on the mean of the respective reciprocals of the resolution width and height

$$\sigma_r^2 = \frac{1}{2} \left(\frac{1}{w} + \frac{1}{h} \right). \quad (5.3.6)$$

As a final remark, we take a look at the dimensionality of the vectors and matrices involved, since it influences the computational complexity of the **UKF**.

The state, state noise and observation noise vectors \mathbf{x} , $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ respectively have dimensions $m + 13$, $m + 13$ and $2m$ if expressed in terms of the number of features m . Thus, the augmented state vector dimension n is given by

$$n = 2(m + 13) + 2m \quad (5.3.7)$$

$$= 4m + 26. \quad (5.3.8)$$

The matrices \mathbf{P} , \mathbf{Q} and \mathbf{R} are square matrices with dimensions $(m + 13) \times (m + 13)$, $(m + 13) \times (m + 13)$ and $2m \times 2m$ with an augmented dimension of $n \times n$. We saw in Subsection 4.1.3 that the number of sigma points are given by

$$2n + 1 = 2(4m + 26) + 1 \quad (5.3.9)$$

$$= 8m + 43 \quad (5.3.10)$$

It is clear that a large number of features result in a large number of sigma points, but more importantly, it also results in very large matrices for a hundred or so features. This slows down the **UKF** dramatically.

5.4 Summary

We have established a framework for the observation, structure and motion models. It may seem inappropriate to use the **UKF**, which is a nonlinear estimator, in combination with our linear models. However, it should be kept in mind that although the motion may seem linear over a small number of steps, the overall process may contain a number of nonlinear effects. It is these unpredictable events that we need the **UKF** to model, for example occlusion. Thus, using linear models do not restrict us to linear processes only.

Chapter 6

Implementation issues

In theory, there is no difference between theory and practice. In practice, there is. *Chuck Reid*

In this chapter, we discuss some of the implementation issues encountered during the course of this thesis. These include decisions about the setup and initialisation of the Unscented Kalman Filter (**UKF**) and use of the Kanade-Lucas-Tomasi (**KLT**) feature tracker. A contribution to the field is made with two methods to handle feature occlusion (Section 6.3). Another implementation issue discussed is that of radial lens distortion, which has a significant effect on locally recorded sequences. Finally, we look at the real-time performance of our Structure from Motion (**SfM**) system.

6.1 Unscented Kalman Filter

6.1.1 Initialisation and setup

Care must be taken when setting up the **UKF**—it is very sensitive to the initial choice of the initial state uncertainty \mathbf{P}_0 . We prefer to set the initial entries of $\mathbf{P}_0 \ll 1$ to ensure convergence of the **UKF**. There exists a trade-off for the choice of the state transition uncertainty \mathbf{Q} . If it is set too large, it causes the **UKF** to oscillate, but if it is set too small, the **UKF** converges slowly. This corresponds with the findings of Bizup & Brown (2003). The

values as indicated in Table 7.1 and Table 7.2 for \mathbf{P}_0 and \mathbf{Q} were found empirically. Note that a distinction between the structure uncertainty \mathbf{Q}_s and the motion uncertainty $\mathbf{Q}_{r,t}$ is made—in some cases it is necessary to increase the structure uncertainty to improve convergence. The downside to this is that the UKF can use this freedom to minimise the error between the real and predicted observations, leading to an erroneous estimated structure. Also, if the ratio of the entries of \mathbf{Q}_s to $\mathbf{Q}_{r,t}$ is large, it will cause the Choleski decomposition to fail, due to ill-conditioning, which will lead to numerical instability. One implementation which uses the Extended Kalman Filter (EKF) sets one of the structure parameters to a known value and fix its corresponding uncertainty at zero (Azarbayejani and Pentland, 1995). If this approach is applied to an implementation based on the UKF, the diagonal state covariance will be singular, causing the Choleski decomposition to fail.

6.1.2 Optimisation

The UKF can be optimised by noting that the Choleski decomposition of \mathbf{Q} and \mathbf{R} need to be calculated only once if its entries are constant over time. Furthermore, if each of these matrices contain identical entries only on the main diagonal, the square root of the entries are equal to the Choleski root for each matrix—the calculation of a single square root is computationally much more efficient. We express this mathematically for the matrix \mathbf{Q} as

$$\text{chol}(\mathbf{Q}) = \text{chol}(k\mathbf{I}) \quad (6.1.1)$$

$$= \text{chol}(k) \text{chol}(\mathbf{I}) \quad (6.1.2)$$

$$= \sqrt{k}\mathbf{I}, \quad (6.1.3)$$

where \mathbf{I} is the identity matrix and k a positive constant.

6.1.3 Sigma structure

Consider the previous state of the UKF at a given time-step. Following the procedure described in Section 4.1.3 and Table 4.5, retrieve the state sigma points $\mathcal{X}_{i-1|i-1}$ from the augmented sigma point matrix $\mathcal{A}_{i-1|i-1}$. These sigma points are then propagated using the state transition model to yield the predicted sigma points $\mathcal{X}_{i|i-1}$, from which the predicted state mean $\bar{\mathbf{x}}_{i|i-1}$ and

state covariance $\mathbf{P}_{i|i-1}$ are calculated. The next step is to create a predicted observation $\bar{\mathbf{y}}_{i|i-1}$ from the sigma points $\mathcal{X}_{i|i-1}$ — refer to the observation prediction step in Table 4.5. To do this, we need to reconstruct the object in the Object Coordinate System (OCS), but closer inspection of (5.1.11) raises the question: how do we reconstruct the object from a set of sigma points, if the initial observation \mathbf{y}_0 and its mean $\bar{\mathbf{y}}_0$ do not form a set of sigma points? Each sigma point need a matching initial observation sigma point. The answer is that we have to do without it. There is no way to create a set of sigma points from the initial observation, because that implies that we know the state sigma points that produced it. We instead use the initial observation and its mean as sigma points. Thus, we rewrite (5.1.11) as

$$\mathcal{P}_{j,i|i-1}^{ocs} = \begin{bmatrix} y_{x,0}(1 + \frac{\mathcal{X}_{j,i|i-1}^s}{f}) - \bar{y}_{x,0} \\ y_{y,0}(1 + \frac{\mathcal{X}_{j,i|i-1}^s}{f}) - \bar{y}_{y,0} \\ \mathcal{X}_{j,i|i-1}^s \end{bmatrix}, \quad j = 0..2n, \quad (6.1.4)$$

where $\mathcal{X}_{j,i|i-1}^s$ is the j -th predicted sigma structure parameter. We continue to work toward a predicted observation, by now rotating and translating the newly created sigma feature $\mathcal{P}_{j,i|i-1}^{ocs}$ to get its location in the Camera Coordinate System (CCS). Care must be taken when rotating each reconstructed sigma feature with its corresponding sigma quaternion $\mathcal{X}_{j,i|i-1}^q$. A quaternion must have unit norm to yield to a valid rotation (Section 3.4). Thus, each sigma quaternion must be normalised to prevent spurious scaling of the sigma structure. Finally, the sigma structure is projected onto the image plane, from which the predicted observation mean $\bar{\mathbf{y}}_{i|i-1}$ and covariance $\mathbf{P}_{\mathbf{y}\mathbf{y}_{i|i-1}}$ are calculated.

6.2 Feature tracking

Although feature tracking is beyond the scope of this thesis, we consider it necessary to review some of the issues encountered. The quality of our results strongly depends on the quality of the tracked features. There are a few factors that influence the quality:

- setup of the feature tracker

- feature selection method(s) used by the feature tracker
- feature tracking method(s) used by the feature tracker.

We opted for the **KLT** feature tracker for this thesis. It is one of the most widely used algorithms to perform the task of point-wise feature tracking in a sequence. It was originally developed by [Lucas & Kanade \(1981\)](#) and later refined by [Tomasi & Kanade \(1991\)](#). We do not discuss the algorithm itself—for more details, see [Wagener & Herbst \(2002\)](#) and [\(Birchfield, 2003\)](#) among other.

A feature is defined by its window size. A good feature is one that is easily tracked in subsequent frames. How are features selected by the algorithm? Areas containing rich textures, such as corners or distinct patches, are considered good and determined by calculating gradients. Another question comes to mind: how are features tracked in subsequent frames? Small inter-frame change is modelled as a translation plus a residue term. This proves to be problematic for the **UKF** when estimating small subsequent rotations, which will incorrectly be estimated as small subsequent translations, as will be shown for the POV-Ray experiment in [Subsection 7.2.3](#). However, their approach is fast and simple, given a moderate number of features. Unfortunately, for the face sequence used in [Subsection 7.3.2](#), where 180 features were tracked, the **KLT** feature tracker's speed hamper the real-time performance of our algorithm.

6.2.1 Feature window size

The feature window size is specific to a particular sequence. According to [Tomasi & Kanade \(1991\)](#), smaller windows are more sensitive to noise. The advantage of smaller windows is that it is less likely to span discontinuities or to be affected by a change in viewpoint. We typically set the window size to 7×7 pixels. See the documentation of the **KLT** feature tracker by [Birchfield \(2003\)](#) on how to fine-tune it.

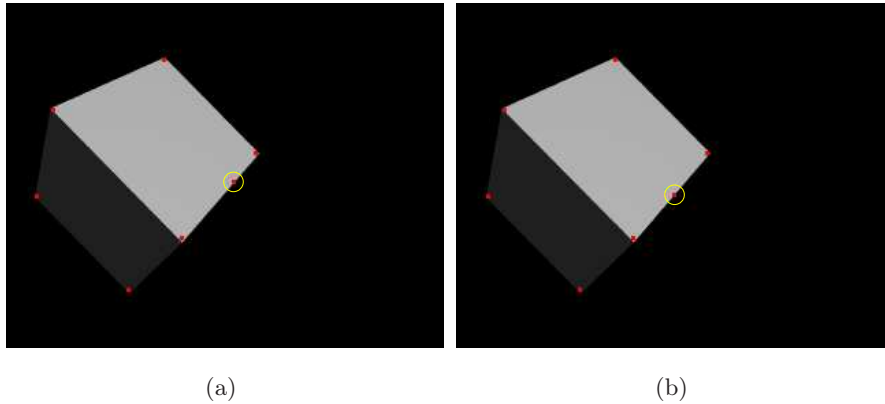


Figure 6.1: *A feature tends to slide along an edge of an object.*

6.2.2 False features

A false feature exists where two edges of two objects of different perspective seem to intersect in the projection. The motion of such a feature is not in accord with the motion of the true features and influences the estimation negatively.

Another example of a false feature is one that is tracked on an edge of an object. Such a feature slides along the edge, creating an inconsistent observation. This usually happens when the feature tracker cannot find the requested number of features. The POV-Ray cube sequence in Subsection 7.2.3 serves as example. During the sequence, only six good features (the vertices) can be tracked. If we request one more feature and the **KLT** feature tracker is unable to find another good feature, it selects one on the edge. Figure 6.1 shows a sliding feature in two subsequent frames.

6.3 Feature occlusion

Feature occlusion is the event of one or more features becoming obstructed. This may be due to the object being rotated in such a way that previously unobstructed features become occluded by the object itself or due to the relative orientation or position of another object.

One approach is to predict the occluded feature based on the best estimate of its three-dimensional (**3D**) orientation and position. Since we assume the

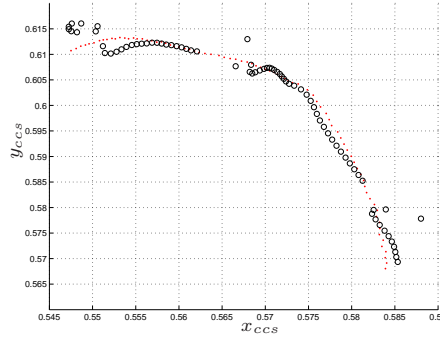


Figure 6.2: *Motion of one feature of the hotel.*

object to be rigid, this should work well, in theory at least. Unfortunately, if the **UKF** has not yet converged, this wrongly predicted feature will skew the structure for as long as that feature is occluded, creating biased estimates. This raises another problem: how do we match a previously occluded feature with its predicted observation and do that without adding too much overhead? It remains an open question. This approach did not yield any meaningful results.

One way to continue accurate tracking during the event of occlusion, is to simply replace the occluded feature with another non-occluded feature. We assume that the object retains its angular velocity and translation velocity and leave the associated parameters unaltered. Thus, the only altered parameter is the replaced feature's structure depth, which may be re-initialised to any value—we choose $s_{new} = 0$. Thus, the structure will appear skewed for a few iterations. This approach works only if the occlusion rate is low and given that the **KLT** feature tracker is able to replace the occluded feature. All tracked features can be incorporated into the final estimated structure as a post-processing step.

Results are included to show the feature replacement method in action. Figure 6.2 shows one of the non-occluded tracked features, where the red dot indicates a **KLT** feature and the black circle the feature tracked by the **UKF**. The **UKF** loses track for two to five iterations at time-steps of occlusion. Although we do not have the ground truth data available for this sequence, inspection of Figure 6.4 shows that the rotation continues almost without interruption. The same conclusion follows from Figure 6.5. The only exception is the trajectory of t_z , which is influenced by the replaced feature's depth,

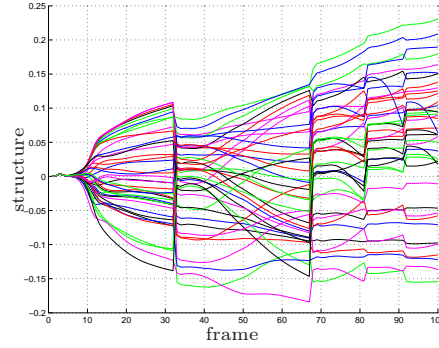
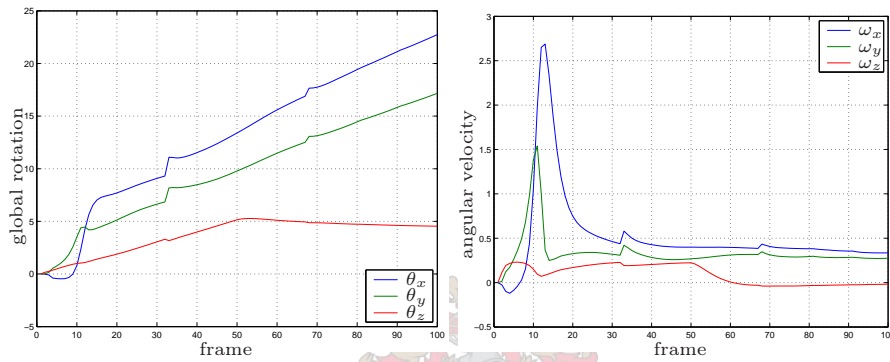


Figure 6.3: *Estimated structure parameters of the hotel.*



(a) *Global rotation.*

(b) *Angular velocity.*

Figure 6.4: *Estimated rotation of the hotel.*

which is re-initialised to zero. Occlusion occurred at frames 33, 68, 82 and 92. Figure 7.22 shows two views of the reconstructed hotel, showing that reconstruction is possible under feature replacement. We conclude that our approach handles feature occlusion well under the condition that the feature occlusion rate is low.

6.4 Radial lens distortion

The assumption made in this thesis (Section 5.1) is that the linear perspective projection model is an accurate description of a pinhole camera. A linear projective geometry projects 3D lines onto the image plane as two-dimensional (2D) lines. However, this is not true in practice, where the most

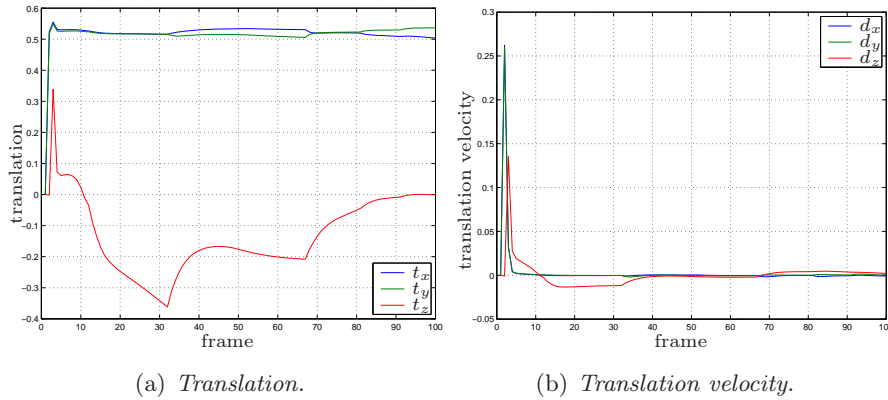


Figure 6.5: Estimated translation of the hotel.

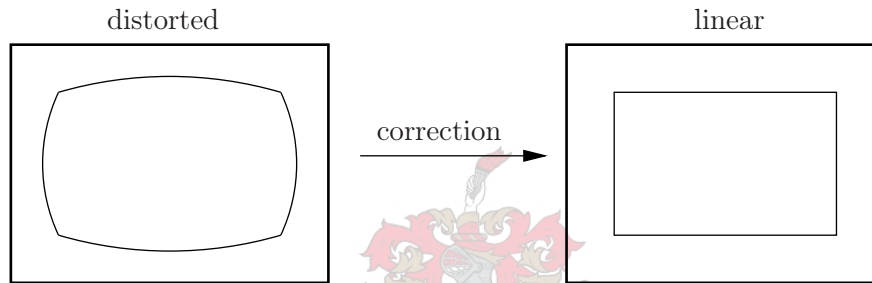


Figure 6.6: The effect of radial lens distortion on straight lines.

common distortion is radial lens distortion. The distortion becomes more severe as the focal length decreases. Figure 6.6 shows the effect of radial lens distortion. The face sequence used in Subsection 7.3.2 suffers from this effect. Thus, to find the true structure and to calculate the Root-Mean-Square Error (RMSE), when compared to the 3D scanned data as for the face experiment in Subsection 7.3.2, the distortion needs to be eliminated.

We relate the actual projected coordinate to the ideal coordinate via a radial displacement. According to Hartley & Zisserman (2001), radial lens distortion is modelled as

$$\mathbf{p}_{corr} = \mathbf{p}_{cen} + L(r)(\mathbf{p}_{rd} - \mathbf{p}_{cen}), \quad (6.4.1)$$

where $\mathbf{p}_{corr} = [x_{corr}, y_{corr}]$ is the corrected (ideal) coordinate in terms of the radially distorted (observed) coordinate $\mathbf{p}_{rd} = [x_{rd}, y_{rd}]$ and the centre of radial distortion $\mathbf{p}_{cen} = [x_{cen}, y_{cen}]$. The function $L(r)$ determines the

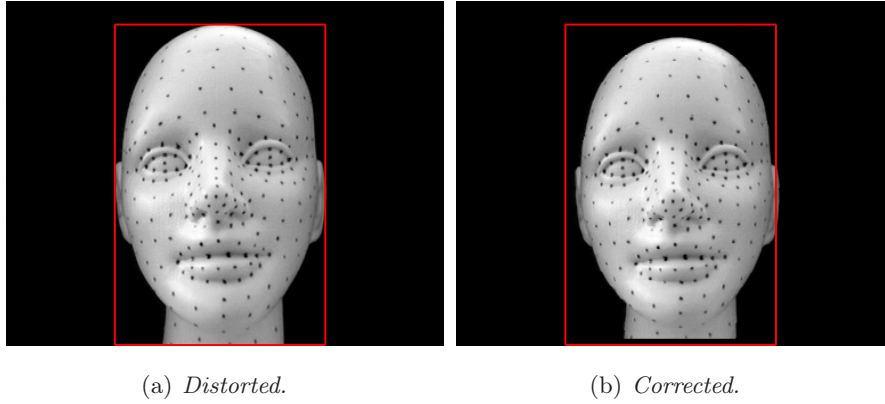


Figure 6.7: Radially distorted and corrected frame from the face sequence.

distortion factor at a radial distance from the centre of distortion, defined as

$$r = \sqrt{(x_{rd} - x_{cen})^2 + (y_{rd} - y_{cen})^2}. \quad (6.4.2)$$

The function $L(r)$ is defined only for positive values of r , where $L(0) = 1$. An arbitrary function may be approximated using a Taylor series expansion, given by

$$L(r) = 1 + k_1 r + k_2 r^2 + \dots, \quad (6.4.3)$$

where the k -coefficients and \mathbf{p}_{cen} are considered part of the intrinsic parameters of the camera. According to the number of coefficients estimated, an equal number of distorted and ideal coordinates must be chosen to yield a unique solution of (6.4.1). In this case, MATLAB's Levenberg-Marquardt method was used to minimise the nonlinear problem in a least squares sense. This procedure is carried out as a post-processing step. We show the effect on one frame in Figure 6.7 from the face sequence. The bounding box is added to show the original position of the face in Figure 6.7(a) in Figure 6.7(b).

6.5 Real-time performance

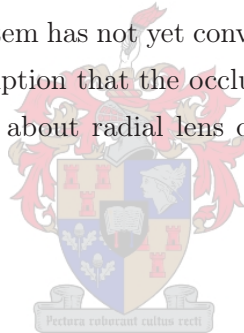
All experiments were run on a 1.8 GHz PC with 1280 MB RAM. The **UKF** is implemented as part of a locally developed software library written in C++. We use an implementation of the **KLT** feature tracker written in C, available from <http://www.ces.clemson.edu/~stb/klt/>.

Our implementation of the **UKF** processes more than 60 frames per second, but the speed is significantly reduced when used in combination with the **KLT** feature tracker. Given a moderate number of features, typically less than 20, it processes 5-15 frames per second. The face sequence's high number of features is processed at 1 frame per minute.

6.6 Summary

The most significant implementation issues encountered, such as the initialisation and setup of the **UKF**, were discussed in this chapter. Some optimisation notes, concerning the Choleski decomposition step as applicable to the **UKF**, was made.

We also gave an overview of the **KLT** feature tracker and the parameters and issues that influence the tracking of good features. Two simple methods to handle feature occlusion were introduced. We concluded that feature prediction performs poorly if the system has not yet converged. Feature replacement performs well under the assumption that the occlusion rate is low. The chapter is concluded with sections about radial lens distortion and the real-time performance of our algorithm.



Chapter 7

Experimental investigation

An experiment is a question which science poses to Nature, and a measurement is the recording of Nature's answer. *Max Planck*

We begin this chapter by defining our measure of error to compare the results from different sequences, before we move on to the actual experiments and results. An extensive number of sequences is used to show the versatility of our approach. The structure of this chapter is summarised in the following list to indicate the flow:

- **Cube sequences**
 - Pure synthetic sequence
 - Noisy synthetic sequence
 - Quasi-real sequence
- **Real sequences**
 - Hotel sequence
 - Face sequence

We begin with a pure synthetic, MATLAB generated cube sequence, simulating the motion of its eight vertices. This is to test our implementation of the Unscented Kalman Filter (**UKF**) and Structure from Motion (**SfM**) models. Next, we corrupt this sequence with observation noise and rerun the

experiment. The next step is to integrate the Kanade-Lucas-Tomasi (**KLT**) feature tracker, which detects point-wise features within each frame—we gave a short overview of the **KLT** feature tracker in Section 6.2. This experiment is performed on a POV-Ray (Persistence of Vision Raytracer) rendered cube sequence and is occlusion-free. The real challenge is to test on real sequences for which ideal conditions seldom hold—practical issues, such as occlusion and radial lens distortion come into play. The first of these experiments is performed on a toy hotel sequence. The final and main experiment is our attempt to perform facial feature reconstruction from a locally recorded sequence.

Just a note on how the rendered reconstructed models presented in this chapter is produced. The three-dimensional (**3D**) reconstructed point-wise features are calculated from the estimated structure parameters in the state vector at a given time-step. These features are then interpolated, using MATLAB's Delaunay triangulation. The choice of interpolation method, linear or cubic, is based on prior knowledge about the structure. The final step is to apply the initial frame of the applicable sequence as a texture map to the interpolated reconstructed model to produce the final reconstructed model.

7.1 Error calculation

It is necessary to define a measure of error to compare the results of the different sequences. We are in particular interested in the accuracy of the reconstruction. Let us now define the distance between the k -th estimated structure parameter and its reference as

$$\Delta s_k = |as_k - s_k^{ref}|, \quad (7.1.1)$$

where a is a scalar used to scale the estimated structure parameters. We use the well-known definition for the Root-Mean-Square Error (**RMSE**) to calculate the reconstruction error

$$e_s = \sqrt{\frac{1}{m} \sum_{k=0}^{m-1} \Delta s_k^2}, \quad (7.1.2)$$

where m is the number of features. This is calculated at a time-step of our choice after the filter has converged. The smaller this value, the more accurate

the reconstruction. The bias-corrected sample variance of the structure is defined as

$$\sigma_s^2 = \frac{1}{m-1} \sum_{k=0}^{m-1} (\Delta s_k - \Delta \bar{s}_k)^2, \quad (7.1.3)$$

where the mean is defined as

$$\Delta \bar{s}_k = \frac{1}{m} \sum_{k=0}^{m-1} \Delta s_k. \quad (7.1.4)$$

This result is an indication of the distribution of the expected error. The smaller the variance, the less the deviation from the true structure.

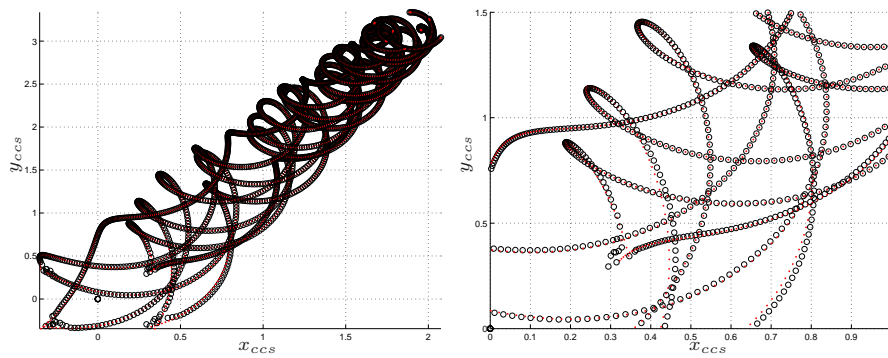
7.2 Cube sequences

This section covers three experiments. The first one is performed on a pure synthetic sequence created with MATLAB. The motion of the eight vertices (point-wise features) acts as observations. The second experiment is performed on the same sequence, but with observation noise added. These two experiments utilise only the **UKF**. The **UKF** is then combined with the **KLT** feature tracker. This combination, referred to as the **SfM** system, is then tested on a software rendered cube created with POV-Ray. The results of this section is summarised in Subsection 7.2.4.

7.2.1 Pure synthetic sequence

This MATLAB generated sequence of a cube serves as sanity check. Its purpose is to test our implementation of the **UKF**. No real feature tracking is performed. Instead, we generated the projected motion of a cube's eight vertices and fed that into the **UKF**. The cube is assumed to be transparent to prevent occlusion. The cube rotates at a constant angular velocity and translates at a constant velocity which suits our linear state transition model. A two-dimensional (**2D**) projection of the **3D** motion is performed in accord with the observation model for a linear perspective projection camera. Thus, the observations are ideal and noise free.

Figure 7.1(a) shows the projected motion for the whole sequence and Figure 7.1(b) a detailed view of the first part. Although the different vertices



(a) Full sequence of observations.

(b) Detailed observations.

Figure 7.1: Motion of the eight vertices of the synthetic cube.

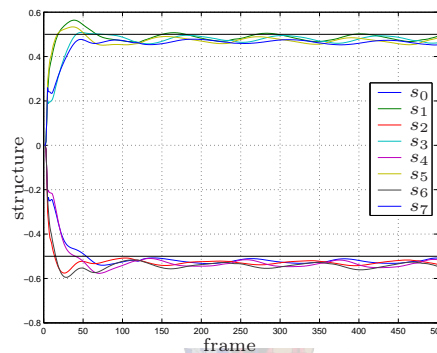
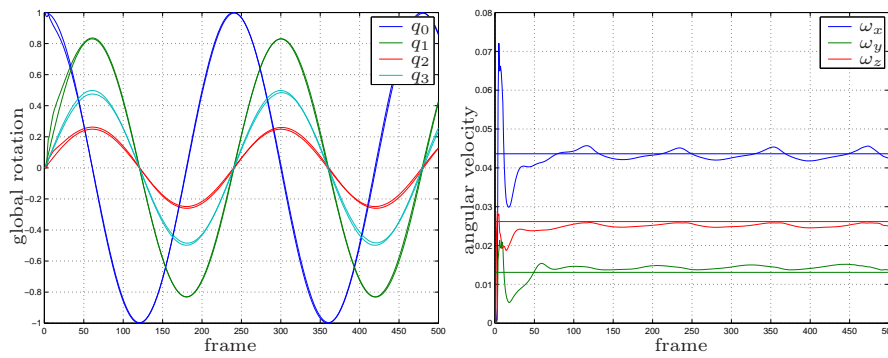


Figure 7.2: Estimated structure parameters of the synthetic cube.



(a) Global rotation.

(b) Angular velocity.

Figure 7.3: Estimated rotation of the synthetic cube.

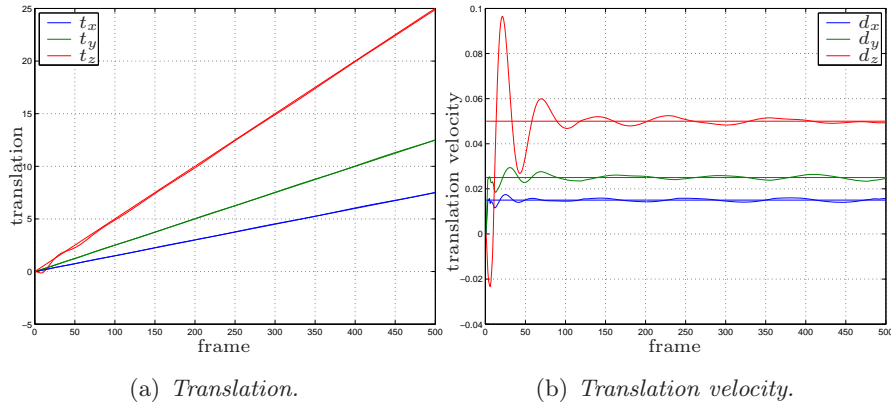


Figure 7.4: Estimated translation of the synthetic cube.

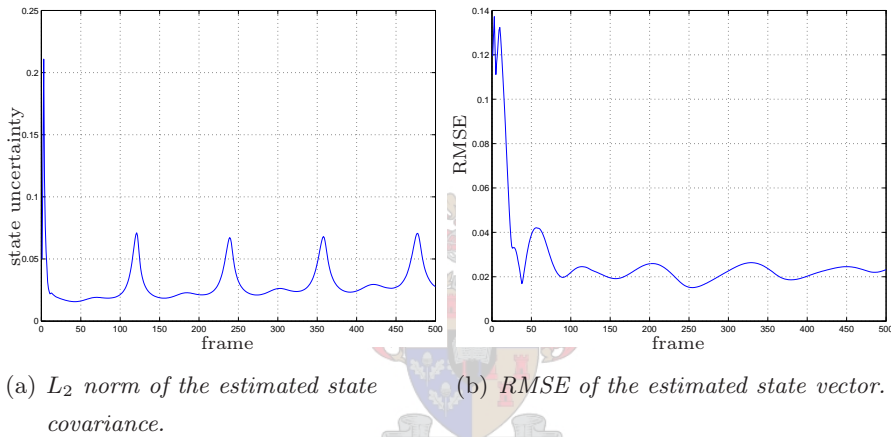


Figure 7.5: Convergence for the synthetic cube sequence.

follow quite different 2D projected trajectories, the UKF manages to track it well—the red dots indicate the true motion and the black circles the filtered motion.

Figure 7.2 shows the estimated structure parameters \mathbf{s} over time. The initial orientation of the cube’s faces is aligned with the axes of the Object Coordinate System (OCS) and the Camera Coordinate System (CCS), implying that half the vertices should lie on either the front or back face. This is observed from Figure 7.2—the structure parameters converge within 100 frames to the true values of ± 0.5 .

Figure 7.3 depicts the true and estimated rotation of the cube. Fig-

Figure 7.3(a) shows the four quaternion parameters. The cube rotates at a constant angular velocity and we expect the quaternion parameters to be sinusoidal. This is similar to the parameterised sinusoidal and cosinusoidal functions that represent circular motion in 2D. Figure 7.3(b) confirms that the angular velocity is constant.

The estimated translation is accurate, showing little deviation from the true translation as shown in Figure 7.4. However, we note in Figure 7.4(b) that the translation model is more sensitive to translation along the z_{ccs} -axis. This sensitivity is also reflected by the estimated structure parameters in Figure 7.2. The reason for the sensitivity is that all the structure parameters \mathbf{s} and the translation parameter t_z encode information about depth.

The UKF takes less than 100 iterations to converge when Figure 7.5(a) and Figure 7.5(b) are considered in combination. Figure 7.5(a) shows the L_2 norm of the state covariance over time, which is an indication of the UKF's uncertainty—the smaller the norm, the more confident the UKF. Notice the peaks in the norm during the course of the sequence. We deduce from this that the motion is highly nonlinear during those sections, which justifies the need for a nonlinear estimator such as the UKF. Although the uncertainty increases, the RMSE of the state vector remains small as indicated in Figure 7.5(b).

An extract from the reconstruction sequence relative to the OCS is shown in Figure 7.6—it is clear that the scaled reconstructed model indicated in black is a cube. The reconstruction error at frame 175 is $e_s = 0.0167$ with a variance of $\sigma_s^2 < 0.0001$.

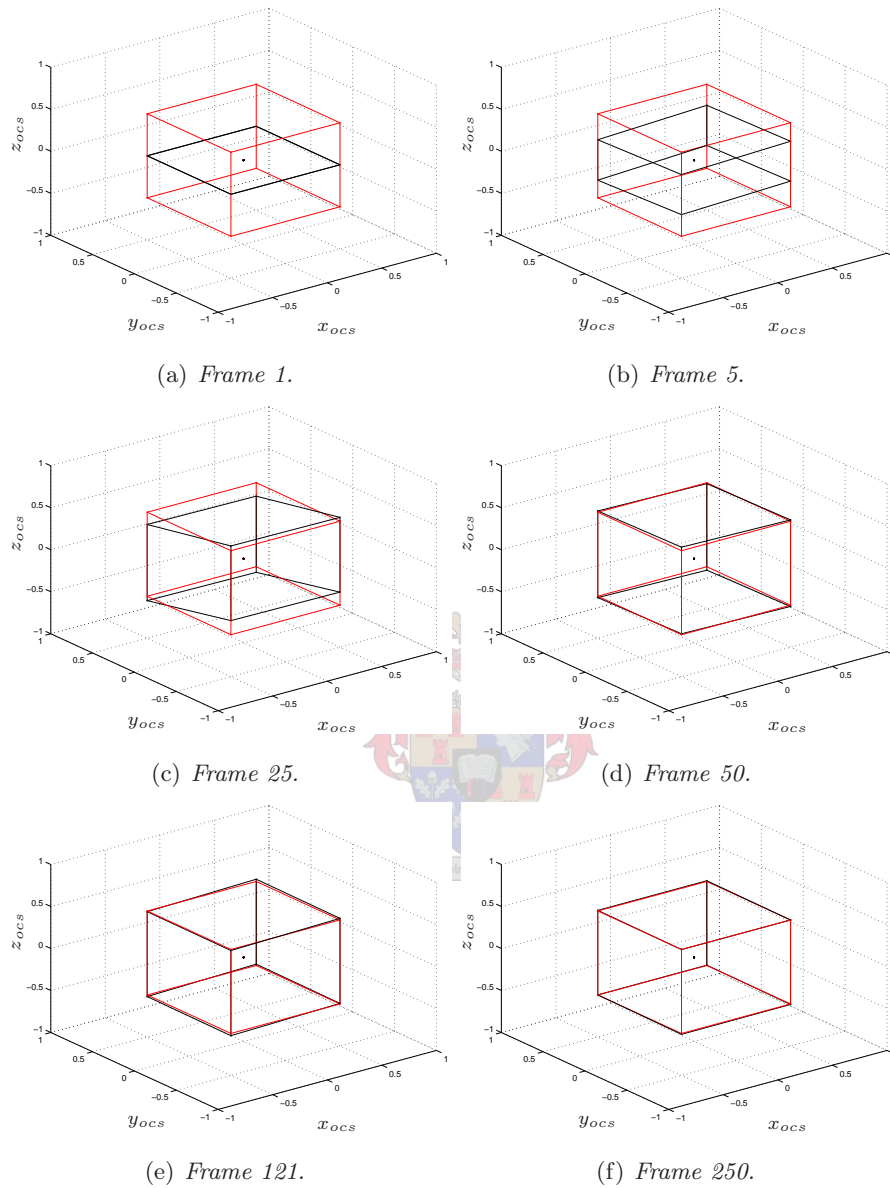


Figure 7.6: Extract from the reconstruction sequence of the synthetic cube where the reconstructed cube is indicated in black and the original cube in red.

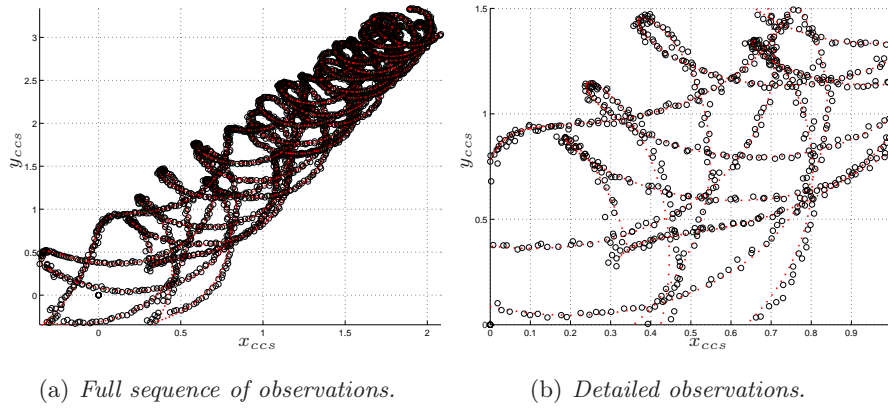


Figure 7.7: Motion of the eight vertices of the noisy synthetic cube.

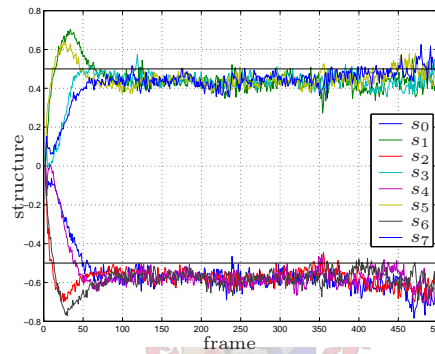


Figure 7.8: Estimated structure parameters of the noisy synthetic cube.

7.2.2 Noisy synthetic sequence

We use the same sequence as in Subsection 7.2.1, but this time it is corrupted with observation noise. The only change in setup of the UKF is the adjusted observation uncertainty. The added zero-mean noise is Gaussian distributed with a variance of 0.0001.

It is apparent from Figure 7.7 that the UKF survives this test—the red dots indicate the true and noise-free motion and the black circles the filtered motion. The noisy estimated structure parameters depicted in Figure 7.8 are on average near the correct values. An interesting deduction can be made from Figure 7.9 and Figure 7.10. Note that the estimated rotation and translation parameters are very smooth—the angular velocity and translation velocity parameters absorb the noise. The only exception is t_z , which is more noisy.

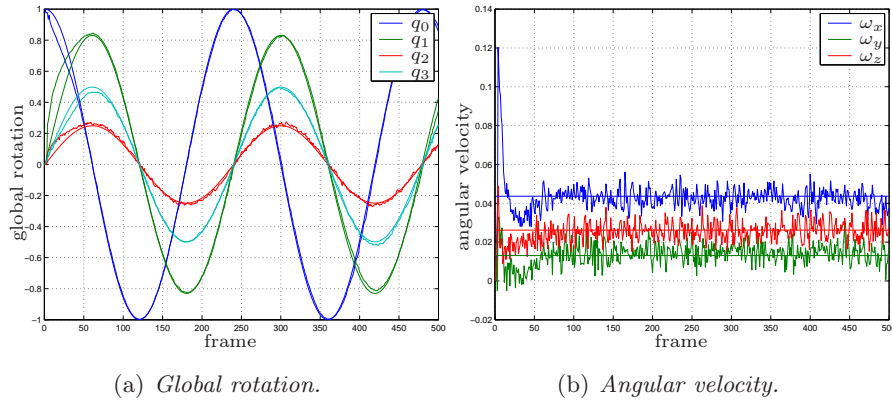


Figure 7.9: Estimated rotation of the noisy synthetic cube.

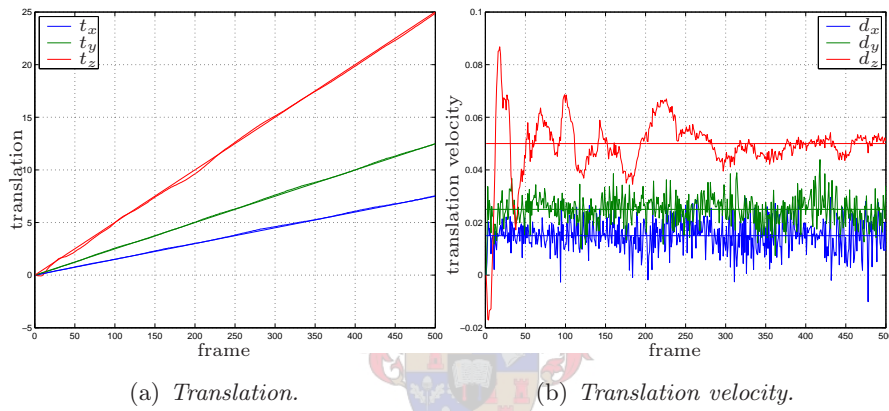


Figure 7.10: Estimated translation of the noisy synthetic cube.

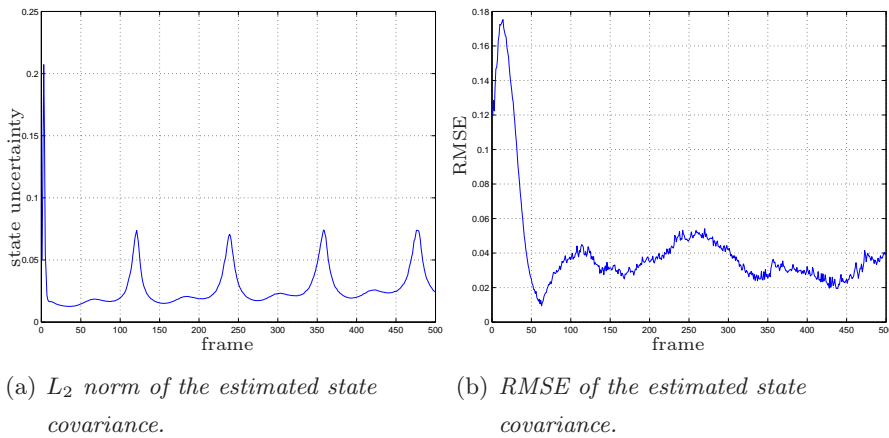
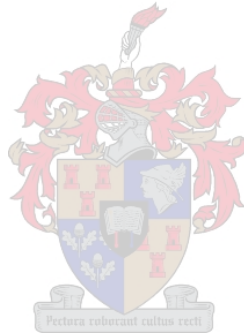


Figure 7.11: Convergence for the noisy synthetic cube sequence.

In the view of our argument about sensitivity along the z_{ccs} -axis in Subsection 7.2.1, we conclude that the noise is absorbed by the structure parameters, since all these parameters encode information about depth.

An extract from the reconstruction sequence is depicted in Figure 7.12. The state uncertainty and RMSE results are comparable to that of Subsection 7.2.1. The reconstruction error at frame 370 is $e_s = 0.0488$ with a variance of $\sigma_s^2 = 0.0014$. As expected, the results are less accurate than the results for the pure synthetic case, due to the added noise. This experiment demonstrates the reconstruction algorithm's dependence on good features—observation noise influences the reconstruction negatively.



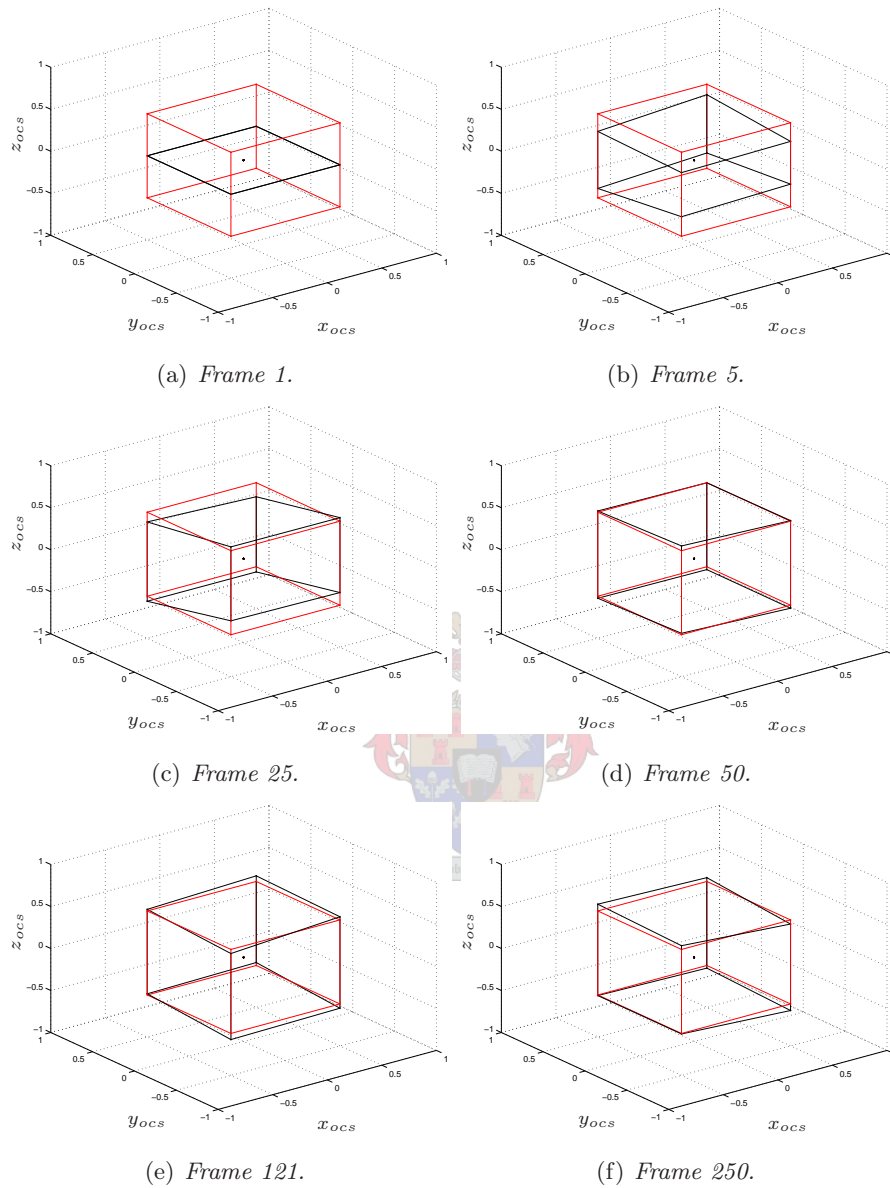
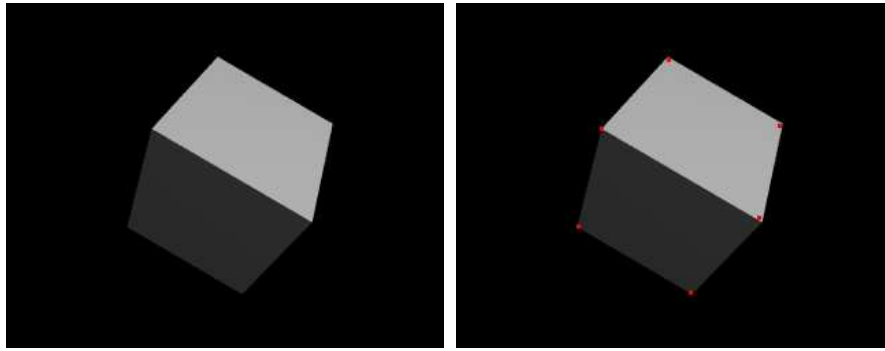
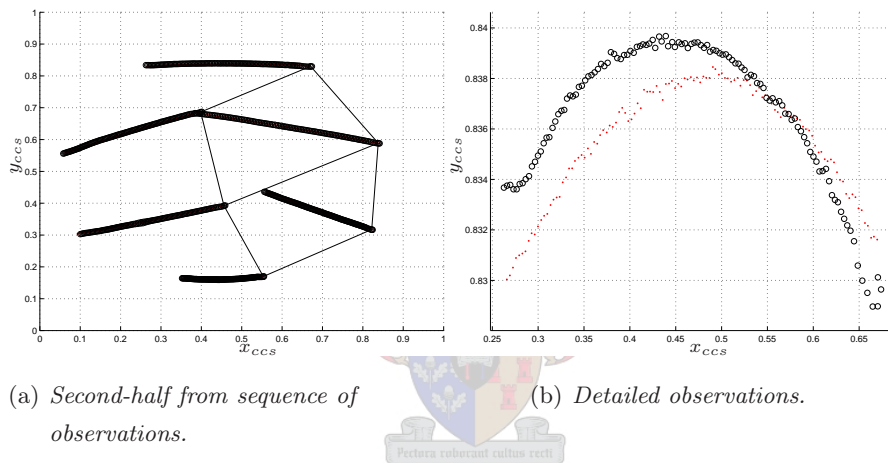


Figure 7.12: Extract from the reconstruction sequence of the noisy synthetic cube where the reconstructed cube is indicated in black and the original cube in red.



(a) Frame 80.

(b) Detected features.

Figure 7.13: One frame from the POV-Ray sequence.

(a) Second-half from sequence of observations.

(b) Detailed observations.

Figure 7.14: Motion of the six features of the POV-Ray cube.

7.2.3 Quasi-real sequence

The **KLT** feature tracker is brought into action in this experiment. The **KLT** feature tracker together with the **UKF** forms the complete **SfM** system. The feature tracker's purpose is to detect and track point-wise features within each frame—the **KLT** feature tracker is discussed in Section 6.2. This experiment acts as an intermediate step before testing on real sequences. We apply the **KLT** feature tracker to a software rendered cube sequence, created with POV-Ray. As in the synthetic cases in Subsection 7.2.1 and Subsection 7.2.2, the projection is performed via a linear perspective projection model, but with the

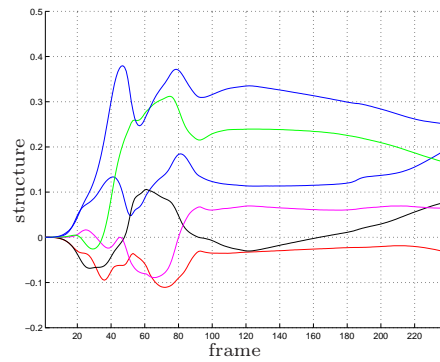
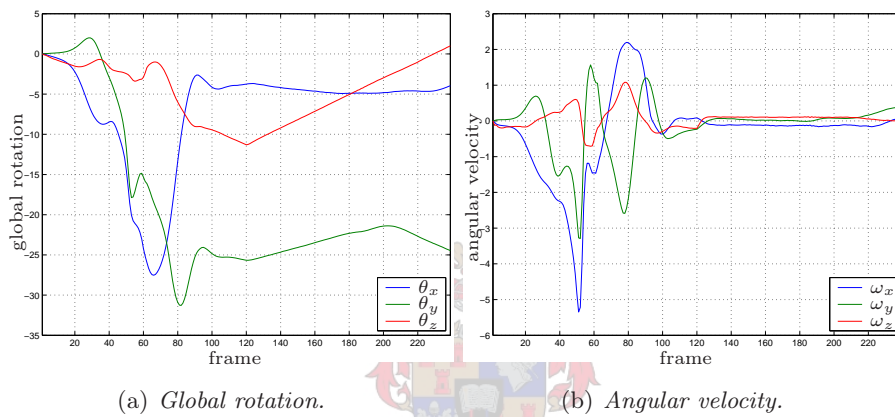


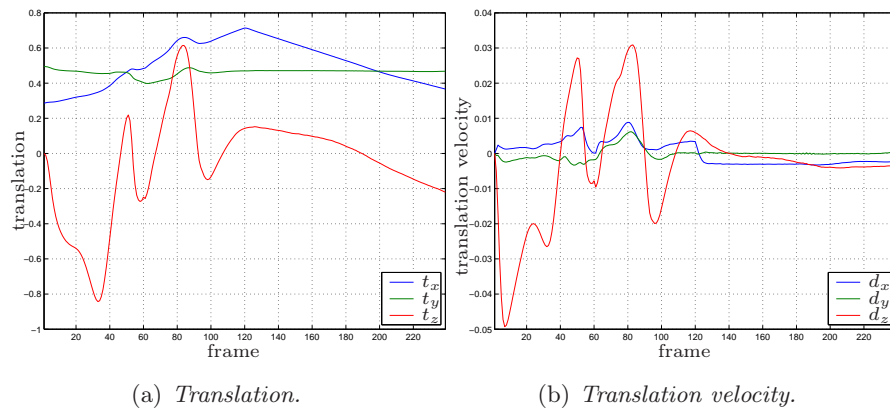
Figure 7.15: Estimated structure parameters of the POV-Ray cube.



(a) Global rotation.

(b) Angular velocity.

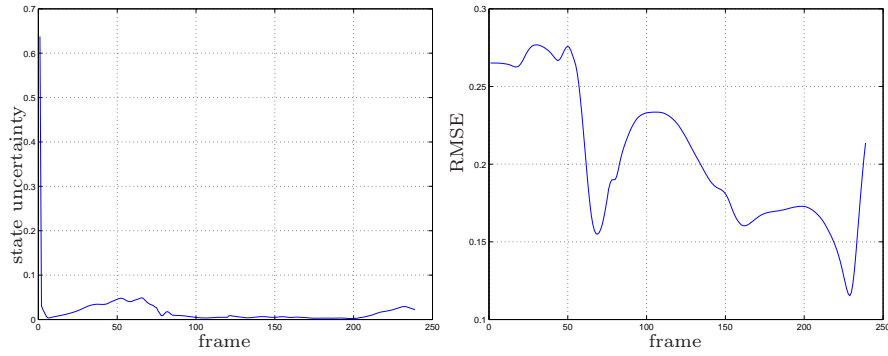
Figure 7.16: Estimated rotation of the POV-Ray cube.



(a) Translation.

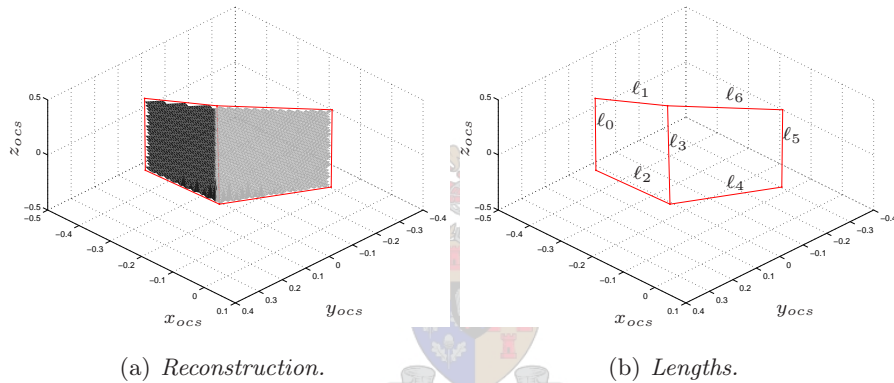
(b) Translation velocity.

Figure 7.17: Estimated translation of the POV-Ray cube.



(a) L_2 norm of the estimated structure covariance. (b) RMSE of the estimated structure.

Figure 7.18: Convergence for the POV-Ray cube sequence.



(a) Reconstruction.

(b) Lengths.

Figure 7.19: The reconstructed POV-Ray cube and its lengths used to calculate the RMSE.

difference that only visible features are projected. Thus, feature occlusion is now a possibility. However, we choose a sequence that is occlusion-free—the matter of occlusion is discussed in Section 6.3.

Figure 7.13 shows two identical frames from the sequence—Figure 7.13(a) shows the rendered input frame and Figure 7.13(b) the frame processed by the KLT feature tracker. The cube’s six visible vertices are tracked throughout the sequence. It translates from left to right while rotating about its z_{ocs} -axis over the first 120 frames. To complete the sequence, this motion is reversed over the second half.

We show the tracked features for the sequence in Figure 7.14—note the observation noise, indicated by the red dots, in Figure 7.14(b). We pointed out in Section 5.3 that two factors influence the quality of the observations:

- accuracy of the feature tracker
- resolution of the frames.

The resolution of this sequence is 320×240 pixels, which yields an estimated noise variance from (5.3.6) of $\sigma_r^2 \approx 0.003$, assuming pixel accuracy for the observations.

Figure 7.15 depicts the estimated structure parameters. Although most of the transients seem to have passed, the structure still changes—the UKF uses the structure parameters to minimise the error between the real and predicted observation. A closer inspection of Figure 7.16 and Figure 7.17 show some more deviations. The cube’s z_{ocs} -parameter of its orientation (Figure 7.16(a)) at frame 120 is $\theta_z = -12^\circ$ and should be $\theta_z = -24^\circ$ —the global rotation is expressed in terms of Euler angles instead of quaternions to facilitate the interpretation of the motion. This incorrect estimate is compensated for by t_z and leads to more erroneous estimates due to this interaction. The reason is that a small interframe rotation can be modelled as a translation and is the reason that this kind of error is not reflected by the RMSE of the structure in Figure 7.18(b). More specific, the KLT feature tracker makes use of this method (Section 6.2, Tomasi & Kanade (1991)), causing the UKF to incorrectly estimate the motion. The UKF does this with great confidence as shown in Figure 7.18(a). Another factor that contributes to the error is the small number of features, causing an underdetermined system, but this is only a suggestion. Once again, note the sensitivity of our algorithm to translation along the z_{ccs} -axis.

The reconstruction error is calculated from the normalised lengths (Figure 7.19(b)) between features along the edges of the cube

$$\ell = [0.9211, 1.0510, 1.0255, 1.2475, 0.8824, 0.9487, 0.9230].$$

Calculating the RMSE of the structure from these values at frame 229 yield $e_s = 0.1153$ with a variance of $\sigma_s^2 < 0.0089$. This does not compares very

well with the results of the previous two subsections. A more representative error gives about 0.2 when calculated over the length of the sequence. The POV-Ray sequence has a greater **RMSE** than the previous results, but the variance is quite low. We investigate the reasons for the larger reconstruction error. The culprits are $\ell_3 = 1.2475$ and $\ell_4 = 0.8824$ —the associated lengths are shown in Figure 7.19(b) together with the reconstruction in Figure 7.19(a). The erroneous estimates of the rotation and translation parameters contribute to this error. A more subtle factor that contributes to the reconstruction error is the calculation of the initial centroid \mathbf{t}_0 from the initial observation \mathbf{y}_0 via (5.3.4) as

$$\mathbf{t}_0 = \begin{bmatrix} \bar{y}_{x,0} \\ \bar{y}_{y,0} \\ 0 \end{bmatrix} \quad (7.2.1)$$

$$= \begin{bmatrix} 0.2859 \\ 0.4951 \\ 0.0000 \end{bmatrix}. \quad (7.2.2)$$

The actual centroid is given by

$$\mathbf{t}_0^{ref} = \begin{bmatrix} 0.2500 \\ 0.5000 \\ 0.0000 \end{bmatrix}. \quad (7.2.3)$$

Calculating the **RMSE** yields a value of 0.0256, which is quite significant. The initial observation itself may be inaccurate. If the **RMSE** between the detected vertices and the true vertices is calculated, it yields an error of 0.0073, which also contributes to the reconstruction error. The initial observation \mathbf{y}_0 is never updated for the length of a sequence—refer to Section 5.1. Consequently, any initial observation error is propagated to all future estimates. Due to this, the translation is now biased and the rotation occur about an incorrect origin. The reconstruction results are none the less considered good.

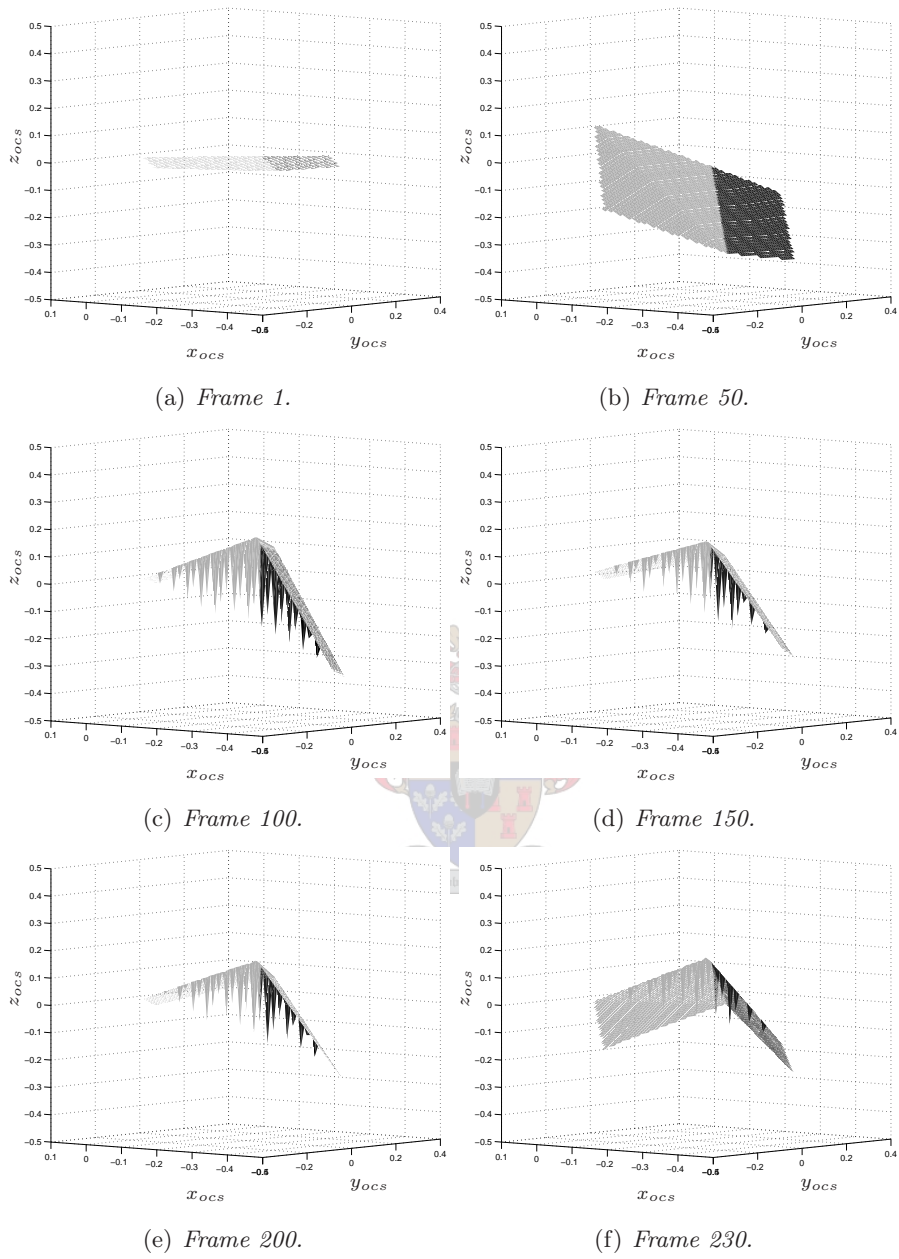


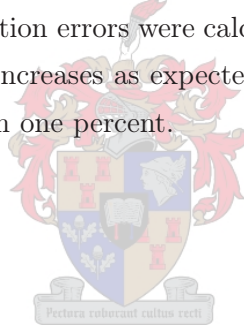
Figure 7.20: Extract from the reconstruction sequence of the POV-Ray cube.

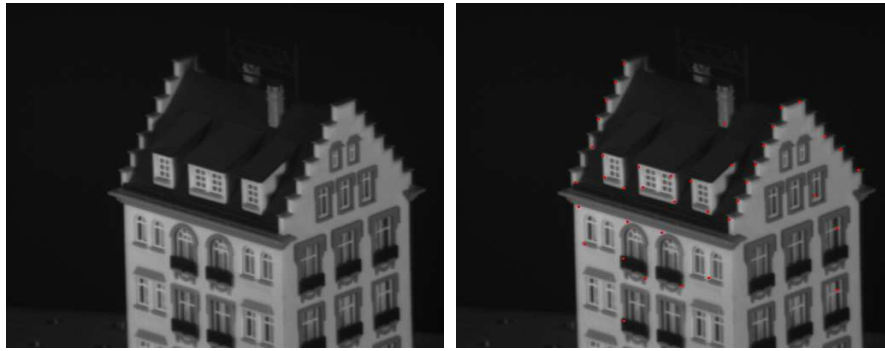
Parameter/Result	Synthetic	Noisy synthetic	Quasi-real
Initial state covariance \mathbf{P}_0	0.0500	0.0500	0.0500
Structure covariance \mathbf{Q}_s	0.0100	0.0100	0.0010
Motion covariance $\mathbf{Q}_{r,t}$	0.0010	0.0010	0.0010
Observation covariance \mathbf{R}	0.0010	0.0010	0.0030
Number of features m	8	8	6
Frame of evaluation	175	370	229
RMSE of structure e_s	0.0167	0.0488	0.1153
Variance of structure σ_s^2	<0.0001	0.0014	0.0089

Table 7.1: Parameters and results for cube sequences.

7.2.4 Results

The results of the experiments of this section is summarised in Table 7.1 for easy comparison. We include the exact setup, number of features and the frames at which the reconstruction errors were calculated. As the experiments become less ideal, the **RMSE** increases as expected. The respective variances are satisfactory, being less than one percent.



(a) *Frame 50.*(b) *Detected features.***Figure 7.21:** *One frame from the hotel sequence.*

7.3 Real sequences

This section covers two more experiments on real sequences, using the complete SfM system. The first experiment is performed on a sequence of a toy hotel. The purpose of this experiment is to test on a sequence not obtained by us to show the versatility of our algorithm. The final experiment is our attempt to perform facial feature reconstruction and is the main experiment of this thesis. We show that facial feature reconstruction is indeed possible using simple modelling.

7.3.1 Hotel sequence

This experiment is performed on a toy hotel sequence. The original can be obtained from Carnegie Mellon University’s Computer Vision Homepage <http://www-2.cs.cmu.edu/~cil/vision.html>. No ground truth data is available for this sequence and we give only a rendered representation of the reconstructed model. This experiment serves as basis for the discussion about feature occlusion—refer to Section 6.3 for more details.

As for the previous sequences, we show one of the frames from the sequence in Figure 7.21. Two different views of the reconstructed model is presented in Figure 7.22. The roof and walls can clearly be identified—we rely on the results of the previous experiments to assume that the reconstruction is accurate. A total of 40 features is tracked throughout this sequence.

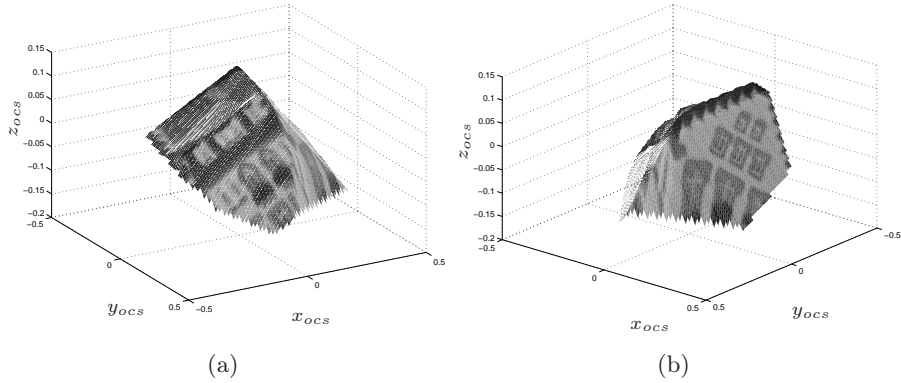
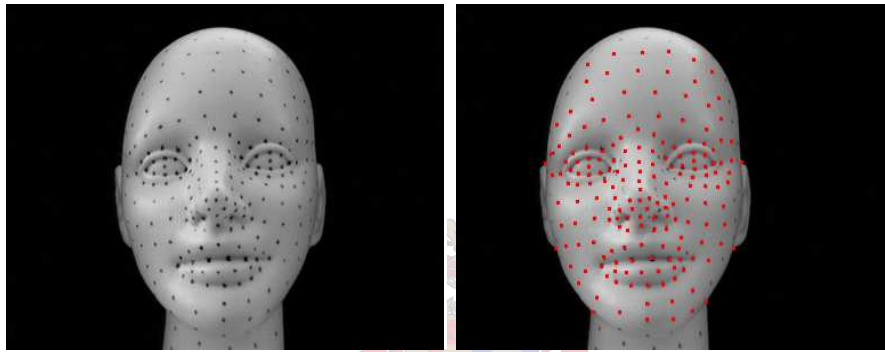


Figure 7.22: Two different views of reconstructed hotel.



(a) Frame 20.

(b) Detected features.

Figure 7.23: One frame from the face sequence.

7.3.2 Face sequence

This experiment forms the focus of this chapter. We test our algorithm on a sequence of a face in an attempt to accurately reconstruct the 3D model from its 2D motion.

A total of 180 features are tracked throughout this 120 frame sequence—the estimated structure parameters are shown in Figure 7.24. A number of markers were placed on the face using a black marker to ease the detection of point-wise features by the KLT feature tracker. The frames has a resolution of 384×288 pixels. The motion is from left to right and back to the point of origin. This translation is confirmed by Figure 7.26—the parameters t_y and t_z are constant as expected. We note that the estimated translation is less

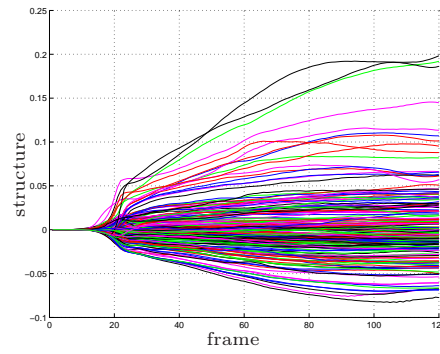
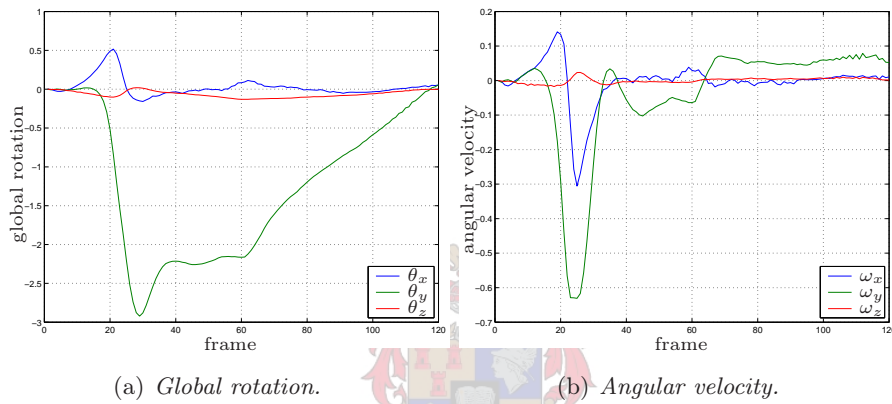


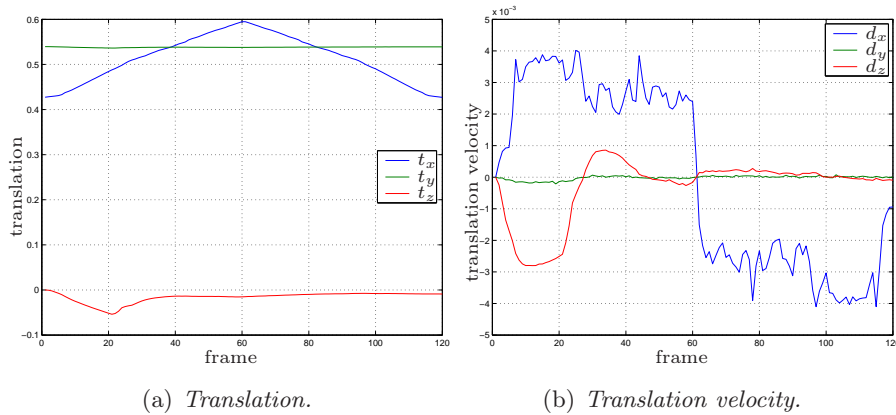
Figure 7.24: Estimated structure parameters of the face.



(a) Global rotation.

(b) Angular velocity.

Figure 7.25: Estimated rotation of the face.



(a) Translation.

(b) Translation velocity.

Figure 7.26: Estimated translation of the face.

sensitive to motion along the z_{ccs} -axis than in the previous experiments. A small rotation error is observed in Figure 7.25. The parameter θ_y reaches a maximum deviation of -3° , although the estimated rotation parameters should ideally be zero throughout the sequence, since the true motion contains only a translation component.

Let us now discuss the reconstruction results. A head-on view of the 3D reconstructed face is presented in Figure 7.27. We are aware that the human brain interprets the 2D reference image, adding 3D information not necessarily present in the reconstructed model. Thus, the usual texture map is not applied to the reconstructed model for this figure. Instead, we added a light source and applied a reflective material as texture map. This is done to convince the reader from a visual point of view that what is observed really is a face. The most prominent facial regions such as the forehead, eyes, nose and mouth can clearly be identified. Figure 7.28 shows two more views of the reconstructed face. An extract from the reconstruction sequence is depicted in Figure 7.29. The RMSE of the reconstruction is calculated as $e_s = 0.1132$ with a variance of $\sigma_s^2 = 0.0129$.

The ground truth structure is obtained from a mechanically scanned version of the face at a horizontal and vertical step-size of 1 mm, containing about 220 000 reference points.

A subtle factor that contributes to the reconstruction error is the effect of radial lens distortion. This kind of distortion is more severe at short focal lengths, which is the case for this experiment. It causes the actual projection to differ from the ideal projection — radial lens distortion is discussed in Section 6.4. Thus, the reconstructed model differs from the real structure. We conclude that our generalised structure model prevents us from achieving a better reconstruction result.

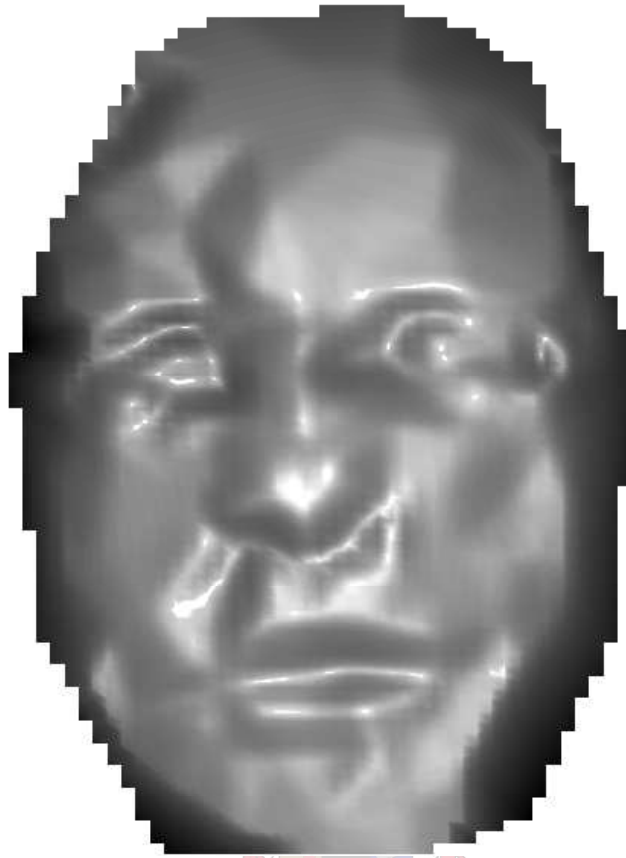


Figure 7.27: Head-on view of the reconstructed face without the reference frame as texture map.

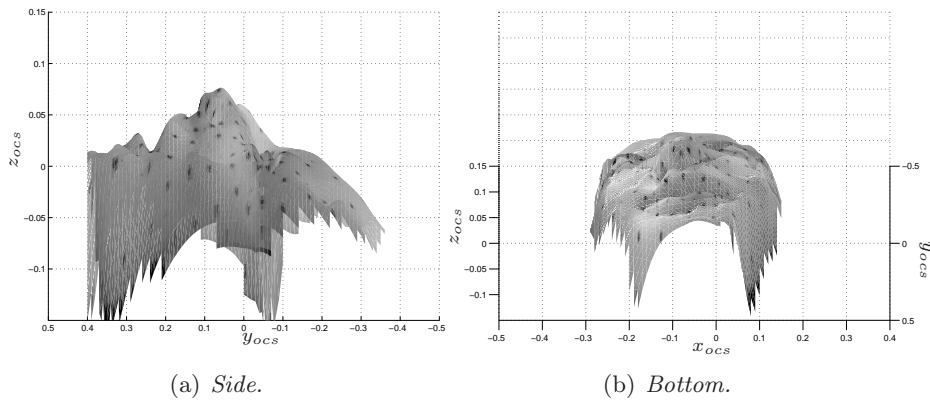


Figure 7.28: Two different views of reconstructed face.

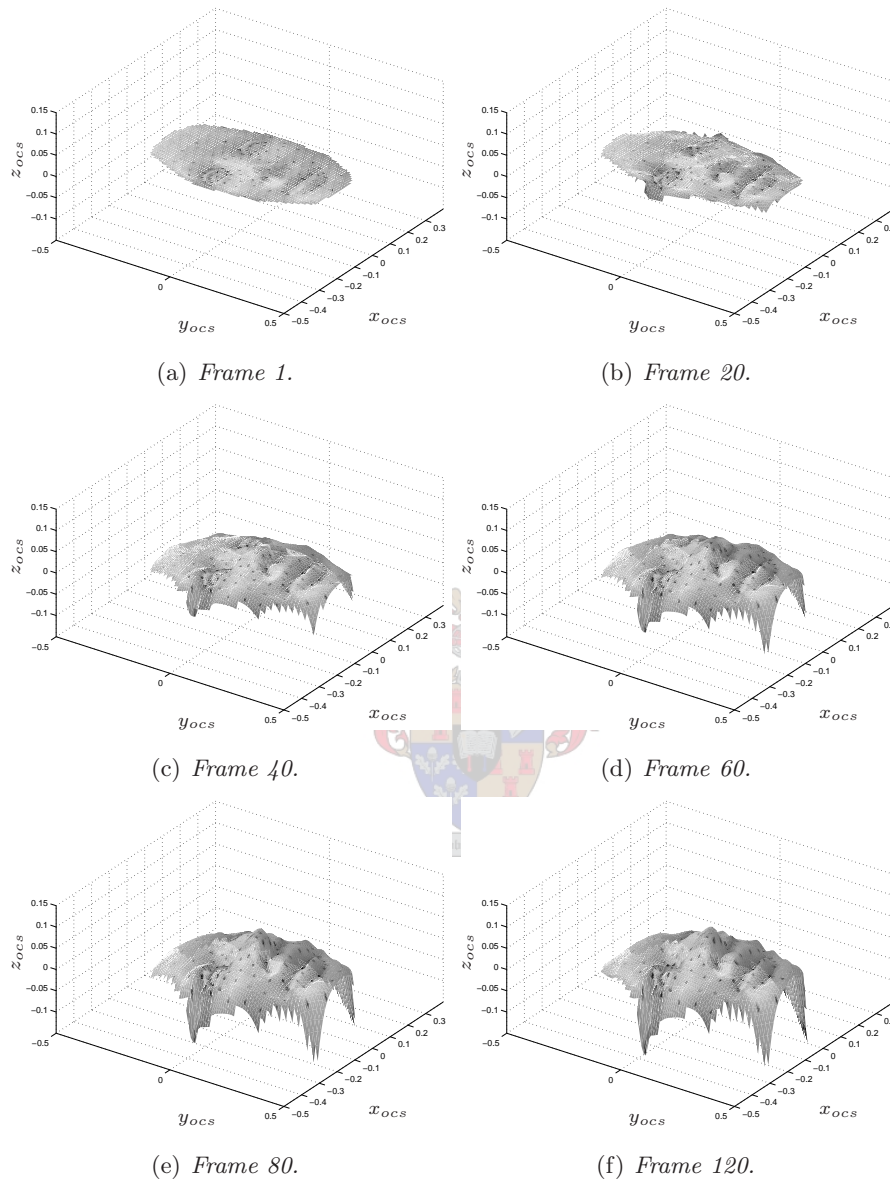


Figure 7.29: Extract from the reconstruction sequence of the face.

Parameter/Result	Face sequence
Initial state covariance \mathbf{P}_0	0.0010
Structure covariance \mathbf{Q}_s	0.0010
Motion covariance $\mathbf{Q}_{r,t}$	0.0001
Observation covariance \mathbf{R}	0.0001
Number of features m	180
Frame of evaluation	120
RMSE of structure e_s	0.1132
Variance of structure σ_s^2	0.0129

Table 7.2: Parameters and results for face sequence.

7.3.3 Results

The results and setup for the face sequence is summarised in Table 7.2. The results of the POV-Ray cube sequence in Table 7.1 is comparable to these results, since both experiments utilise the full SfM system. The only difference is the effect of radial lens distortion.



7.4 Summary

Several aspects of the inner workings of our SfM system were covered in this chapter, using a variety of experiments. We focused on facial feature reconstruction. The system is particularly sensitive to translation along the z_{ccs} -axis. Future improvements should consider the decoupling of \mathbf{s} and t_z to prohibit or limit the interaction between these two parameters. The system's greatest limitation lies in its structure model, which may be improved by incorporating prior knowledge about the structure.

The system strongly depends on the initial observation, which influences the estimation of the structure and motion. The structure parameters rely on the initial observation to calculate the best estimate of the structure as calculated within the observation model. More consequences of a initial observation error is an incorrect centre of rotation and a biased translation.

The system's performance can potentially be influenced by the performance of the KLT feature tracker, since it may introduce observation noise additional to the discretisation of each frame. However, if configured well, it should not pose a problem.

Some of the practical issues involved when implementing a system like this were discussed in Chapter 6. Supplementary material is provided on CD in Appendix B. It contains the reconstruction sequences as short video clips, as well as the 3D models, including the POV-Ray cube, the toy hotel and especially the reconstructed face, among other results and demonstrations.

Chapter 8

Conclusions

Education is what survives when what has been learnt has been forgotten.

Burrrhus Frederoc Skinner

Given a set of two-dimensional (2D) observations, we have shown that accurate three-dimensional (3D) structure and motion estimation can be achieved, using the popular Structure from Motion (SfM) approach. Previous implementations (Azarbayejani and Pentland, 1995; Jebara *et al.*, 1999; Jebara and Pentland, 1996; Ström *et al.*, 1999) are based on the Extended Kalman Filter (EKF), but we opted for the Unscented Kalman Filter (UKF), due to its ease of implementation and higher statistical accuracy. Several aspects were considered during the course of this thesis. We summarise in terms of the theory discussed, the results achieved and the system’s limitations. This thesis is concluded with some improvements and extensions for future research.

Quaternions provide a simple and efficient way to represent rotations. A formula, based on the derivation by Wertz (1986), for the time-derivative of a quaternion was derived in Section 3.5 for use in the motion dynamics. We gave special attention to the theoretical comparison of the EKF to the UKF — refer to Subsection 4.1.3. The UKF proves to be superior to the EKF, based on prediction methods and modelling, computational complexity and statistical accuracy (Table 4.6). The UKF’s only drawback is the increased overhead due to an increase in the number of sigma points. This is induced by an increase in the number of features, which consequently increases the state dimension,

but that is a small price to pay for improved accuracy. The final theoretical chapter covered the motion dynamics of a rigid object and the formulation of the linear perspective projection model for a pinhole camera (Chapter 5). This formulation decouples the focal length from the structure depth.

Chapter 6 covered the implementation issues, regarding the implementation of the **UKF** and its integration with the Kanade-Lucas-Tomasi (**KLT**) feature tracker. We also discussed feature occlusion. One way to handle feature occlusion is to predict the occluded feature's **3D** location, based on its best visible estimate of its structure parameter. We came to the conclusion that this approach only works after the **UKF** has converged. Another method is to reject the occluded feature and replace it with another non-occluded feature (Section 6.3). Radial lens distortion has a significant effect on real sequences at short focal lengths (Section 6.4). Correction of this kind of distortion is compulsory to find the true structure.

The experiments in Chapter 7 covered a wide range of sequences, ranging from pure synthetic sequences to real sequences. The pure synthetic sequences tested only our implementation of the **UKF** and showed the dependence on good features. The structure and motion results for the synthetic sequences are extremely accurate, even under noisy conditions. Recall that our approach is sensitive to motion perpendicular to the image plane.

The next natural step was to integrate the **KLT** feature tracker with the **UKF** to form the complete **SfM** system. The system was tested on a quasi-real sequence, which proved to be successful, except that the reconstruction is skewed due to initialisation errors: any errors introduced by the initial observation, which is used to do the reconstruction, propagates to all future estimates, since this information is never updated for the length of a sequence—this is one of the limitations of our approach. One factor that contributes to the reconstruction error is the feature measurement error. Another factor is an incorrect centroid, due to the distribution of the initially observed features. This results in a biased translation and an incorrect centre of rotation, as discussed in Subsection 7.2.3. The choice of the initial state covariance may prevent the **UKF** from converging. The final and most significant experiment to put the **SfM** system to the real test, was to perform facial feature reconstruction. In our view, the reconstruction proved to be successful, with a reconstruc-

tion error of about 10%—refer to Figure 7.27 for the reconstructed face and Table 7.2 for the numerical results.

The approach presented in this thesis is subjected to the following constraints:

- tracking of a single object only
- the object is assumed to be rigid
- tracking is performed in a clutter-free environment.

Thus, this thesis leaves much space for future research. We make a few suggestions for improvement, regarding these limitations.

The motion modelling can be improved in a number of ways:

- Particle Filter hybrid systems
- integration of inertial parameters
- improve feature tracking
- segmentation using HMMs and parametric curves
- segment deformable object into smaller rigid objects

The Particle Filter (PF) provides better statistical modelling when compared to the UKF, since it is not limited to unimodal Gaussian Probability Density Functions (PDFs) only. The PF is a nonparametric method and has the ability to estimate a nonparametric PDF, defined by a discrete set of particles. It is common to integrate a PF with a Kalman Filter (KF). It must be stressed that PFs are computationally expensive. The motion model can be extended by integrating inertial parameters. This will of course only enhance the performance if inertia plays a significant role. By suggesting to improve feature tracking, we imply that better tracking can be achieved if the estimator and the feature tracker are more tightly integrated in terms of feedback. Parametric curves are often used to segment an image to distinguish an object from its cluttered background. A trained Hidden Markov Model (HMM) can be used to identify the object to be tracked. Point-wise features for structure estimation can then be selected within this region. Object tracking is not limited to rigid

objects only—a deformable object can in some cases be defined by smaller interconnected objects.

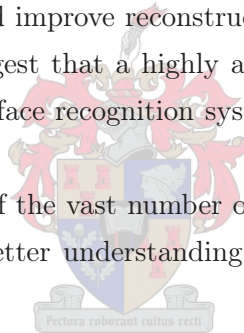
The following suggestions are made to improve reconstruction:

- initialisation using prior structure knowledge
- prohibit or limit the interaction between the structure parameters and the translation’s z -parameter.

If prior knowledge about the structure is available, it can be used to initialise the structure parameters. For example, the reconstructed face presented in this thesis can be used as a generic model for a face—this should improve convergence if this properly initialised system is used to reconstruct a human face. We saw in Chapter 7 that our approach suffers from interaction between the structure parameters and the z -parameter of the translation, since both encodes information about depth. If this interaction can in any way be prohibited or limited, it should improve reconstruction results.

As a final remark, we suggest that a highly accurate SfM system can be combined with a high quality face recognition system to perform 3D recognition.

This thesis covered some of the vast number of possibilities in the field of Computer Vision to gain a better understanding of the inner structure and motion of SfM.



Appendix A

Supplementary information

The supplementary information provided in this appendix acts as guide for the project as supplied on CD in Appendix B. The CD contains all resources, including the sequences and literature used in this thesis (Section A.1). The most important MATLAB scripts are discussed in Section A.2. Section A.3 discusses the libraries needed to run the filter software discussed in Section A.4. Section A.5 explains how to setup and invoke an experiment, using the facial feature reconstruction experiment as example. The directory structure of the project is shown in Figure A.1 for easy navigation.

This thesis, typeset with $\text{\LaTeX} 2_{\epsilon}$, is located under `latex/` as `thesis.tex` and can be compiled by invoking the command `make` from a terminal on a Unix or Linux system.

A.1 Resources

The literature used in this thesis is provided under `papers/` in electronic format. Additional material is provided under `html/`.

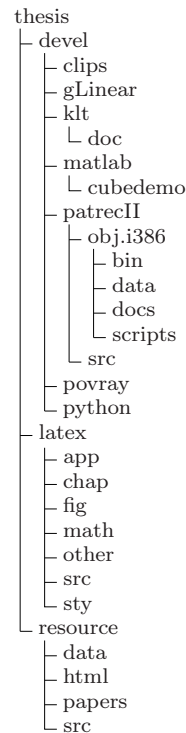


Figure A.1: *Directory structure of the project.*

The hotel and face sequences are located under `resource/data/` as short video clips. The separate frames used by the respective experiments are located under `obj.i386/data/`. The scripts necessary to render the POV-Ray cube sequence is located under `povray/`. The following version of POV-Ray was used:

```
Persistence of Vision(tm) Ray Tracer Version 3.5.0c-9
(Debian i386-linux-gcc)
This is an unofficial version compiled by:
Jeroen van Wolffelaar <jeroen@wolffelaar.nl> for Debian
(www.debian.org)
The POV-Ray Team(tm) is not responsible for supporting this
version.
Copyright 1991-2002 POV-Ray Team(tm).
```

Invoke POV-Ray from a terminal with `povray +ldata.pov data.ini`. This will produce a set of png images. The **KLT** feature tracker software accepts only pgm images. A robust Python script to do batch image conversions, using ImageMagick's `convert` utility, is located under `python/`. Use `./bconvert -e png -f pgm` to convert from png to pgm, where the `-e` switch indicates the input format and the `-f` switch the output format.

A.2 MATLAB scripts

The following MATLAB scripts are used to convert between various rotation representations:

- `quattorotm.m` converts a quaternion to a rotation matrix
- `rotR.m` creates a rotation matrix from Euler angles in degrees
- `quattoeuler3.m` converts a quaternion to Euler angles in radians
- `eulertoquat.m` converts an Euler axis-and-angle to a quaternion.

The graphical comparison of the **EKF** to the **UKF** in Subsection 4.1.3 was created with the script `sigma.m`.

The following function reconstructs point-wise features from a set of structure parameters (`beta`), given the focal length `f`, the initial observation `yInit` and a centroid `cen`.

```
function obj3D = sfmreconstruct(f, yInit, beta, cen, varargin)
```

Additional parameters may be supplied via `varargin`, such as a rotation matrix, translation vector and a scale factor.

The directory `kalman/` contains a 2D **UKF** demonstration (`ukf.m`), which shows the **UKF**'s ability to track a number of different curves. The data sets are created within this script.

The synthetic cube scripts are located under `cubedemo/`. Use the script `rot_cube.m` to simulate the linear motion of the cube's vertices. The translation and rotation parameters are set within. Invoke the **UKF** with `ukf_cube.m`. This script relies on the state transition and observation models.

```
function ptsup = ffun_cube_n(pts, noise)
function ptsm = hfun_cube_n(pts, noise, yInit, cen, f)
```

The arguments `pts` and `noise` denotes the sigma point matrices as depicted by the algorithm in Table 4.5. The results are viewed via `ukf_res.m`. A reconstruction video clip, `cube500_xvid.avi`, is located under `clips/`.

A.3 Libraries

The following version of the `gcc` compiler was used to compile the different libraries and software.

```
gcc (GCC) 3.3.4 (Debian 1:3.3.4-3)
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.
```

The filter software relies on the following C/C++ libraries:

- `libgLinear.so` or `libgLinear.a`

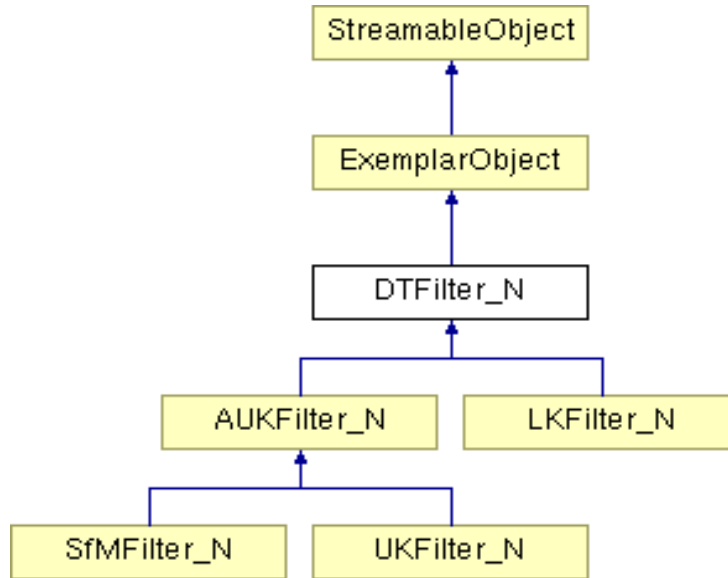


Figure A.2: *DTFilter_N* class hierarchy.

- `libgpatrecII.so` or `libgpatrecII.a`
- `libklt.a`

Invoke `make` from `gLinear/` to create the necessary matrix algebra library. The **KLT** feature tracker library is built by invoking `make` from `klt/`—its documentation is available under `klt/doc/`. Invoke `make lib` from `obj.i386/` to build the `patrecII` library. The mentioned libraries should all reside under `lib/`.

The newest source of the **KLT** feature tracker can be obtained from the website <http://www.ces.clemson.edu/~stb/klt/>. The newest source of `gLinear` and `patrecII` can be obtained via CVS. Note that the Stellenbosch University retains copyright on all software.

A.4 Filter software

The filter hierarchy is depicted in Figure A.2—it forms part of the greater `patrecII` hierarchy. The discrete-time n -dimensional filter class `DTFilter_N` is pure virtual. Both the classes `LKFilter_N` and `AUKFilter_N` derive from it. The class `AUKFilter_N` is an abstract class and has two children. The

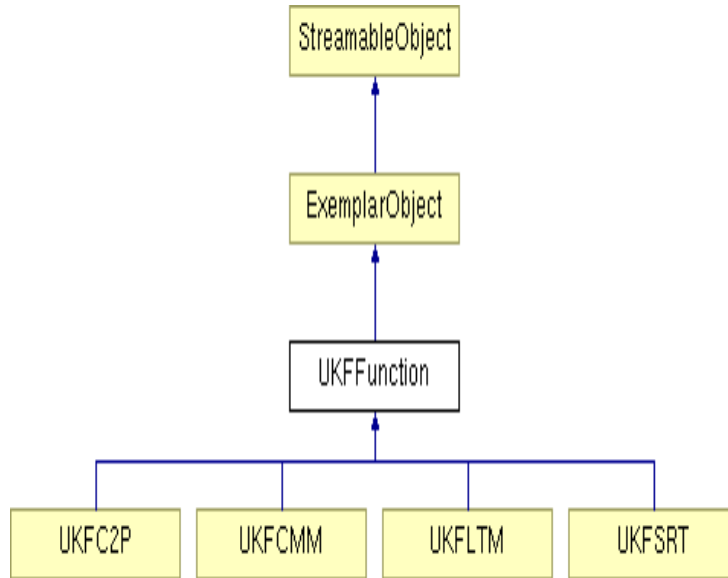


Figure A.3: *UKFFunction* class hierarchy.

class `UKFilter_N` is a generalised **UKF** and the class `SfmFilter_N` contains an instance of `UKFilter_N` and the state transition and observation models. These models are derived from `UKFFunction` as depicted in Figure A.3. The class `UKFSRT` implements the structure, rotation and translation submodels and the class `UKFCMM` implements the camera model. The full `patrecII` documentation is available under `patrecII/docs/html/`.

The source of the complete **SfM** system is provided under `patrecII/src/` as `sfmnew.cc`. It combines the `SfmFilter_N` with the **KLT** feature tracker. To build the application, invoke `make sfmnew` from `obj.i386/`. The application should reside under `bin/`.

The application assumes that the sequence is available as a set of `pgm` images located in `obj.i386/data/`, named `data001.pgm` to `data999.pgm`. Uncompress one of the subdirectories under `obj.i386/data/`. The application can be executed interactively or setup via a configuration file. To use the interactive option, execute `./bin/sfmnew` from `obj.i386/` in a terminal. Table A.2 shows an example. To use a configuration file, pass the configuration file as argument to the application, for example `./bin/sfmnew ./data/data.txt`. A typical configuration file is shown in Table A.1. Note that the comments are to describe each field and should not be part of the file. The `tf` field has no effect

```

SFMIO_v0    // header

vf: 180     // number of visible features
tf: 180     // number of trackable features
frames: 120 // number of frames
vsu: 0.001  // visible structure uncertainty
tsu: 0.001  // trackable structure uncertainty
rtu: 0.0001 // motion uncertainty
f: 10       // focal length
isu: 0.001  // initial state uncertainty
mu: 1e-4    // observation uncertainty
fd: 7       // minimum feature distance
ww: 5       // feature window width
wh: 5       // feature window height

```

Table A.1: *A sample configuration file.*

```

Running SfM:
  Number of visible features (vf): 180
  Number of trackable features (tf >= vf): 180
  Number of frames: 120
  Process uncertainty (structure): 0.001
  Process uncertainty (rotation and translation): 0.0001
Would you like to configure more parameters? (y/n): y
Type 0 to select a default value.
  Camera focal length (default 10): 10
  UKF Initial state uncertainty (default 0.005): 0.001
  UKF Measurement uncertainty (default 1e-4): 0.0001
  KLT Minimum distance among features (default 15): 7
  KLT Window width (default 7): 5
  KLT Window height (default 7): 5

```

Table A.2: *A sample configuration prompt.*

and should be set equal to `vf`. A distinction was made between visible features and occluded features during the course of this thesis when implementing a method to predict occluded features. Although this approach did not work, the framework was left intact. The same holds for the `vsu` and `tsu` fields.

The `KLT` feature tracker outputs a number of files to `obj.i386/data`. The files named `feat001.ppm` to `feat999.ppm` are the original images overlaid with the tracked features. The files named `feat001.mat` to `feat999.mat` are text files which contain the pixel coordinates of each tracked feature—refer to the documentation of the `KLT` feature tracker for more information. The normalised input and output of the `UKF` are respectively written to `input.mat` and `output.mat`. The estimated state at each time-step is written to `state.mat`. A file named `norm.mat` is also produced and contains the norm of the state covariance matrix. The results for a specific sequence are viewed with the `sfmshowcpp.m` script located in `scripts/`.

A.5 Facial feature reconstruction

We will now show how to run the facial feature reconstruction experiment. The face sequence is located under `obj.i386/data/doll/`. Execute the application as described in Section A.4 as `./bin/sfmnew ./data/doll.txt`. Wait an hour or two. The radial lens distortion is corrected with the script `run_face_rdc.m`. The `2D` control points between the scanned face and the reconstructed face must be selected by hand. The scanned data is available in a text format under `resource/data/` as `face_ud_ref.asc`. The following function can be used to correct radial lens distortion, where `T` denotes the input image, `cen` the centre of distortion and `k` the k -coefficients.

```
function Tc = rdc_image(T, cen, k, m)
```

Finally, to view the `3D` reconstructed model, run the script `sfmfit2.m`. A number of clips are available under `demo/`, but the most significant one probably is `face_rec_demo.avi`.

Appendix B

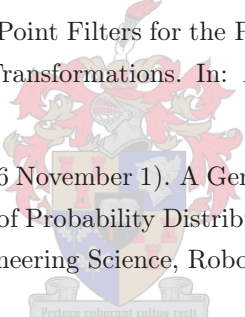
Supplementary CD

Refer to Appendix A for an exposition and explanation of the contents of the CD. Feel free to contact the author.



Bibliography

- Atick, J.J., Griffin, P.A. and Redlich, A.N. (1996). Statistical Approach to Shape from Shading: Reconstruction of Three-Dimensional Face Surfaces from Single Two-Dimensional Images. *Neural Computation*, vol. 8, No. 7, pp. 1321–1340.
Available at: citeseer.nj.nec.com/atick97statistical.html
- Azarbayejani, A. and Pentland, A. (1995 July). Recursive Estimation of Motion, Structure and Focal Length. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 562–575.
Available at: citeseer.nj.nec.com
- Birchfield, S. (2003). KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. Documentation.
Available at: <http://www.ces.clemson.edu/~stb/klt/>
- Bizup, D. and Brown, D.E. (2003). The Overextended Kalman Filter. Tech. Rep., University of Virginia, Systems and Information Engineering.
- Broida, T.J., Chandrashekar, S. and Chelappa, R. (1990). Recursive 3-D Motion Estimation from a Monocular Image Sequence. *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 4, pp. 639–656.
Available at: citeseer.ist.psu.edu/494751.html
- Chen, Y., Huang, T. and Rui, Y. (2002 September). Parametric Contour Tracking using Unscented Kalman Filter. *International Conference on Image Processing (ICIP02 Oral)*.
- Coutsias, E.A. and Romero, L. (1999 February 12). The Quaternions with an Application to Rigid Body Dynamics. Tech. Rep., University of New Mexico, Department of Mathematics and Statistics, Albuquerque, NM 87131.
- Dockstader, S.L. and Tekalp, A.M. (2001 October). On the Tracking of Articulated and Occluded Video Object Motion. *Real Time Imaging*, vol. 7, No. 5, pp. 415–432.
Available at: <http://www.ece.rochester.edu/~dockstad>

- Eberly, D. (2002 September 27). Quaternion Algebra and Calculus. Magic Software, Inc.
Available at: <http://www.magic-software.com>
- Hartley, R. and Zisserman, A. (2001). *Multiple View Geometry in Computer Vision*. Cambridge University Press.
- Isard, M. and Blake, A. (1998). Condensation – Conditional Density Propagation for Visual Tracking. *Computer Vision*.
- Jebara, T., Azarbayejani, A. and Pentland, A. (1999 May). 3D Structure from 2D Motion. *IEEE Signal Processing Magazine*, vol. 16, no. 3.
- Jebara, T. and Pentland, A. (1996 November 28). Parameterized Structure from Motion for 3D Adaptive Feedback Tracking of Faces. Tech. Rep., MIT Media Laboratory, Perceptual Computing Technical Report 401, Cambridge MA, 02139.
- Julier, S. (1999). The Scaled Unscented Transformation. Tech. Rep., IDAK Industries, 901 Missouri Blvd., 179 Jefferson City, MO 65109.
- Julier, S. (2002). Reduced Sigma Point Filters for the Propagation of Means and Covariances Through Nonlinear Transformations. In: *Proc. American Control Conference*.
- Julier, S. and Uhlmann, J.K. (1996 November 1). A General Method for Approximating Nonlinear Transformations of Probability Distributions. Tech. Rep., University of Oxford, Department of Engineering Science, Robotics Research Group, Oxford, OX1 3PJ United Kingdom.
- 
- Kalman, R.E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45.
- LaViola, Jr, J.J. (2003 June). A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion. In: *American Control Conference*, pp. 2435–2440. IEEE press, PO Box 1910, Providence, RI, 02912, USA.
- Lerios, A. (1995 December 13). Rotations and Quaternions. Tech. Rep., Stanford University.
- Li, P. and Zhang, T. (2002). Unscented Kalman Filter for Visual Curve Tracking. Tech. Rep., Harbin Institute of Technology, Department of Computer Science and Engineering 360, Harbin, Hei Long, Jiang Province, China, 150001. Statistical Methods in Video Processing in conjunction with ECCV2002.

- Lucas, B.D. and Kanade, T. (1981 April). An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI). In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 674–679. A more complete version is available as Proceedings DARPA Image Understanding Workshop, April 1981, pp.121–130. When you refer to this work, please refer to the IJCAI paper.
- Morrell, D. (2003). Extended Kalman Filter Lecture Notes. Class notes.
Available at: <http://www.fulton.asu.edu/~morrell/581/>
- Nickels, K. and Hutchinson, S. (2001 February). Model-Based Tracking of Complex Articulated Objects. *IEEE Trans. on Robotics and Automation*, vol. 17, pp. 28–36.
Available at: citeseer.ist.psu.edu/nickels01modelbased.html
- Qian, G. and Chellappa, R. (2001). Structure from Motion using Sequential Monte Carlo Methods. Tech. Rep., University of Maryland, Center for Automation Research and Department of Electrical and Computer Engineering, College Park, MD 20742.
- Qian, G., Chellappa, R. and Zheng, Q. (2001 December). Robust Structure from Motion Estimation using Intertial Data. *Journal of the Optical Society of America A*, vol. 18, no. 12.
- Rui, Y. and Chen, Y. (2001 December 11-13). Better Proposal Distributions: Object Tracking Using Unscented Particle Filter. *International Conference on Computer Vision and Pattern Recognition*, pp. 786–793.
- Schwab, A.L. (2002 May 16). Quaternions, Finite Rotation and Euler Parameters. Unpublished.
- Schwardt, L. (2003). DSP813 Lecture 10 – The Kalman Filter. Class notes.
- Sherlock, B. and Herbst, B. (2003). Introduction to the Kalman Filter and applications. Unpublished.
- Ström, J., Jebara, T., Basu, S. and Pentland, A. (1999). Real Time Tracking and Modelling of Faces. Tech. Rep., MIT Media Laboratory, Cambridge MA, 02139.
Available at: citeseer.nj.nec.com/strom99real.html
- Sturm, P. (1997 June). Critical Motion Sequences for Monocular Self-Calibration and Uncalibrated Euclidean Reconstruction. *International Conference on Computer Vision and Pattern Recognition*, pp. 1100–1105.

- Szeleski, R. and Kang, S.B. (1996 February). Shape Ambiguities in Structure from Motion. Tech. Rep., Cambridge Research Laboratory, Digital Equipment Corporation.
- Tomasi, C. and Kanade, T. (1991 April). Detection and Tracking of Point Features. Tech. Rep. CMU-CS-91-132, Carnegie Mellon University.
Available at: citeseer.ist.psu.edu/tomasi91detection.html
- van der Merwe, R., Doucet, A., de Freitas, N. and Wan, E. (2000 June 9). The Unscented Particle Filter. Tech. Rep., Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, England.
- van der Merwe, R. and Wan, E. (2001 April). Efficient Derivative-Free Kalman Filters for Online Learning. In: *Proceedings of European Symposium on Artificial Neural Networks (ESANN)*. Bruges, Belgium.
Available at: citeseer.nj.nec.com/vandermerwe01efficient.html
- van der Merwe, R. and Wan, E. (2003 May 23). The Square-Root Unscented Kalman Filter for State and Parameter Estimation. Tech. Rep., Oregon Graduate Institute of Science and Technology, 20000 NW Walker Road, Beaverton, Oregon 97006, USA.
- Venter, C. (2002). *Structure from Motion estimation using a Nonlinear Kalman Filter*. Master's thesis, University of Stellenbosch.
- Wagener, D.W. and Herbst, B. (2002). Face Tracking: An implementation of the Kanade-Lucas-Tomasi Tracking algorithm. Unpublished.
- Ward, D.B., Lehmann, E.A. and Williamson, R.C. (2003 November). Particle Filtering Algorithms for Tracking an Acoustic Source in a Reverberant Environment. *IEEE Transactions on speech and audio processing*, vol. 11, no. 6, pp. 826–836.
- Wertz, J.R. (1986). *Spacecraft Attitude Determination and Control*. D Reidel Publishing Company.
- Wheeler, M.D. and Ikeuchi, K. (1995 December 10). Iterative Estimation of Rotation and Translation. Tech. Rep., Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213.
- Zikic, D. and Wein, W. (2004 May 4). 3D Rotations and Quaternions. Slides.

We must not cease from exploration and the end of all our exploring will be to arrive where we began and to know the place for the first time.

TS Elliot

