UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot · your knowledge partner

A VIDEO-BASED TRAFFIC MONITORING SYSTEM

by

Lourenço *Lázaro* Magaia

Dissertation presented for the Degree

of

Doctor of Philosophy

at the University of Stellenbosch

Supervisors:
Prof. BM HERBST
Dr. NL MULLER

Stellenbosch, South Africa

December 2006

Declaration

I, the undersigned, hereby declare that the work contained in this dissertation 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

This thesis addresses the problem of building a *video-based traffic monitoring system*. We employ clustering, tracking and three-dimensional reconstruction of moving objects over a long image sequence. We present an algorithm that robustly recovers the motion and reconstructs three-dimensional shapes from a sequence of video images, Magaia et al [91].

The problem of recovering three-dimensional information from two-dimensional projections is an important consideration in many fields, such as computer vision, automation, traffic and the entertainment industry. Structure-from-Motion aims to recover the three-dimensional structure of a moving object from a series of two-dimensional projections. The solution to the problem is obtained by employing non-linear approaches/models, for example an extension or generalization of the Kalman filter. Over the past twenty years, the *Extended Kalman Filter* (*EKF*) has become established as the best algorithm to solve non-linear estimation problems, Bar-Shalom et al [7], van der Merwe [146]. As a result, most recent tracking and estimation algorithms are based on non-linear Kalman filters.

The increasing demand for performance, robustness and low cost of surveillance systems is one of the main reasons why researchers need to recover motion robustly. Therefore, in this study we have applied a robust algorithm based on *Robust Optical Flow* (*ROF*) and the *Unscented Kalman Filter* (*UKF*) to track and estimate the structure and motion of moving objects respectively. In order to recover the structure of an object we need to provide our system with as many features as possible. There are many tracking algorithms available, such as *Kanade-Lucas-Tomasi* algorithm (*KLT*), *Corner Tracker* and *Robust Optical Flow*. Comparing *KLT* and *ROF* we found that the *ROF* algorithm gave more accurate results, thanks to its ability to track features robustly. Hence *ROF* does not loose feature during tracking process. In our experiments we use the Harris corner detector, Harris and Stephens [57], to provide *ROF* with as many features as possible.

Given a sequence of video frames, we first extract the background and then track some features including those inside and outside of the boundaries of the moving

object. Next we apply the *UKF* to recover the three-dimensional structure of the object. We assume that we are reconstructing a rigid object under perspective projection. Since we only know the projection, we initialize the system with zero, but this results in the algorithm converging slowly or diverging. Therefore, we have accelerated the convergence of the system by considering the initialization of the system with the two-dimensional projection for x and y coordinates while z is maintained as zero or using the minimal information about a flat object. The *Unscented Kalman Filter* converges about eighty frames.

The separation of objects was made possible by using the k-means clustering algorithm. First we tested the algorithm using synthetic data and then we applied it to real data. The algorithm was tested on twelve sequences taken in four different places under different weather conditions. We included a case where the camera was subject to small vibration. The system behaved perfectly and it was able to recover the shapes of objects even when the clustering algorithm failed. Full three-dimensional reconstruction could be achieved using less than eighty frames only.

Opsomming

Hierdie proefskrif word daarop toegespits om 'n video-gebasseerde verkeer moniteringsisteem daar te stel deur gebruik te maak van bondeling, opsporing en drie dimensionele rekonstruksie van bewegende voorwerpe oor 'n groot aantal beelde. Ons ontwikkel 'n algoritme wat die beweging en rekonstruksie van drie dimensionele vorms uit 'n reeks videobeelde onttrek, Magaia et al [91].

Die verkryging van drie dimensionele inligting vanuit twee dimensionele projeksies is 'n probleem wat in verskeie velde voorkom, soos byvoorbeeld rekenaar visie, outomatisasie, die verkeer, sowel as die vermaaklikheidsindustrie. Die basis agter Stuktuur-uit-Beweging is die verkryging van 'n drie dimensionele struktuur van 'n bewegende voorwerp vanuit die voorwerp se twee dimensionele projeksies. 'n Oplossing tot die probleem word verkry deur gebruik te maak van nie-lineêre benaderings/modelle, soos byvoorbeeld 'n uitbreiding of veralgemening van die *Kalman Filter* (*KF*). Die *Extended Kalman Filters* (*EKF*) het oor die laaste twintig jaar bekendheid verwerf as die beste algoritme om nie-lineêre benaderingsprobleme op te los, Bar-Shalom et al [7], van der Merwe [146]. As gevolg hiervan word meeste van die onlangse opsporing- en benaderingsalgoritmes gebasseer op nie-lineêre *Kalman Filters*.

Die toenemende aanvraag in vertoning, robuustheid en lae onkoste van toesig sisteme is van die hoof redes waarom navorsers beweging so robuust as moontlik wil verkry. Daarom het ons in hierdie studie 'n robuuste algoritme, gebasseer op *Robust Optical Flow* (*ROF*) en die *Unscented Kalman Filter* (*UKF*), toegepas om respektiewelik die opsporing en benadering van die struktuur en beweging van voorwerpe te bepaal. Vir die verkryging van 'n struktuur van 'n voorwerp moet ons sisteem met soveel as moontlik eienskappe voorsien word. Daar bestaan baie opsporingsalgoritmes soos byvoorbeeld *Kanade-Lucas-Tomasi (KLT)*, *Corner-Tracker* en *Robust Optical Flow*. Deur *KLT* en *ROF* te vergelyk, is daar gevind dat die *ROF* algoritme meer akkurate resultate lewer, danksy die algoritme se robuuste opsporingseienskappe. *ROF* verloor dus nie eienskappe gedurende die opsporingsproses nie. In ons eksperimente het ons gebruik gemaak van die Harris hoek waarneming, Harris en Stephens [57], om *ROF* met so veel as moontlik eienskappe te voorsien.

Die eerste taak, gegee 'n reeks van opeenvolgende videobeelde, is om die agtergrond te onttrek. Dit word gevolg deur die opsporing van sekere eienskappe, insluitend eienskappe binne en buite die grense van die bewegende voorwerp. Volgende word die *UKF* toegepas om die drie dimensionele struktuur van die voorwerp te verkry. Ons aanvaar dat ons die rekonstruksie van 'n rigiede voorwerp onder perspektiewe projeksie doen. Aangesien slegs die projeksie bekend is, inisialiseer ons die sisteem met die waarde nul, maar dit kan óf lei tot 'n stadige konvergensie, óf divergensie van die algoritme. Gevolglik word die sisteem se konvergensie versnel deur die sisteem te inisialiseer met twee dimensionele projeksies van x en y koördinate, terwyl z as nul gevestig word, of deur die minimum inligting van 'n nie-starre voorwerp te gebruik. Die *Unscented Kalman Filter* konvergeer na ongeveer tagtig beelde.

Die verdeling van die voorwerpe was moontlik gemaak deur die *k-means clustering* algoritme te gebruik. Die algoritmes is eers getoets op kunsmatige data, waarna dit toegepas is op werklike, realistiese data. Die algoritme was getoets op twaalf reekse van beelde wat in vier verskillende plekke onder verskillende weersomstandighede geneem is. 'n Geval waar die kamera aan vibrasie blootgestel was, was ook ingesluit. Die sisteem het perfek gewerk en kon vorms van voorwerpe verkry, selfs al het die bondelingsalgoritme gevaal. 'n Volledige drie dimensionele rekonstruksie kon verkry word deur minder as tagtig beelde te gebruik.

Contents

List of Figures

List of Tables

Acknowledgments

In the course of projecting, preparing and realizing this thesis I had the collaboration, help and support of prestigious people, foremost among them Ben Herbst, Neil Muller, the Magaia family and the Stephens family. I am deeply grateful for their close collaboration and help.

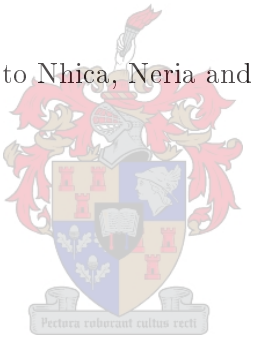
My thanks go to my project leaders, Professor Ben M Herbst, whose energy and enthusiasm is matched only by his understanding of the subtle art of Applied Mathematics. Without him, the project would have been impossible to accomplish. He guided me in my research path and never ceased to astonish me with fresh view points and ideas. My family whose inspiration guided me all the time and whose support was a key for my long stay away from them. Dr. Neil Muller, who served as mentor and research collaborator listened to all the complaints I had when things didn't go as planned, which happened quite often. His creativity and joy at maths and programming tricks, system, hardware and software configuration were very helpful. Neil is a person whose enthusiasm makes working with him fun and exciting. Abreu, Barbara Stephens, Debby and kids, Dique and Pam and kids: you are on this page because you deserve my deep respect.

I am very grateful to the Department of Transport of Stellenbosch particularly Bennie Fourie for all things I learned from him; it was truly thrilling to work with him. I had wonderful training in this Department.

To the people of the Department of Mathematical Sciences at the University of Stellenbosch, you all deserve my special thanks. The people of Eduardo Mondlane University, particularly from the Department of Mathematics and Informatics, glad to be a part of you, my thanks.

Last, but not least, my thanks goes to all the people from the International Office at the University of Stellenbosch, particularly Linda, and other friends I made in Stellenbosch. I am very thankful.

Dedicated to Nhica, Neria and the Magaia family



TO MY FATHER WHO IS ASTONISHINGLY INSPIRING!

Notation

Note: Exceptions to the following notations will be specified where they occur.

Notation	*Description*
a	Scalar value
$\mathbf{a}$	Column vector
$\mathbf{a}^T$	The transpose of $\mathbf{a}$.
A	Matrix A
$F(.)$ or $H(.)$	Function $F(.)$ or $H(.)$
$a(\tau_k)$	The value of a at time τ_k.
a_n	Refers to the n^th component of $\mathbf{a}$.
$\overline{\mathbf{x}}_{i\|i-1}$	The predicted state mean of expected value of $\mathbf{x}_i$, given the observations $\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}$
$\{a\}_n$	a sequence of objects
$\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$	n-dimensional basis vectors or index
$\Re^{n\times n}$	n-dimensional space
$a(\tau_{k+1}\|\tau_k)$	The *a priori* value of $\mathbf{a}$ at time τ_{k+1}, based on its value up to time τ_k.
$x(\tau_k)$	The *a priori* value of x at time τ_k
$\hat{a}$	an *estimate* instead of the true value of a
$a + bi$	A complex number. a and b are real scalars, with i the imaginary unit.
$a + bi + cj + dk$	A quaternion representation of numbers. a, b, c, d are real scalars, and i, j, k are generalisations of $(-1)^{\frac{1}{2}}$ or $i^2 = j^2 = k^2 = -1$.
(Z_x, Z_y)	Image plane projection
S_{ro}	Shape of rigid object

Glossary

CCS - Camera Coordinate System - A system used to describe an object using the camera location.

CCTV - Closed Circuit Cameras - A camera system used to monitor objects in a determined area, such as *road monitoring*.

DOF - Dense Optical Flow - estimation of the flow at each pixel in the image.

EKF - Extended Kalman Filter - The first version of Kalman Filter (ordinary Kalman Filter) was initially applied to linear problems only but with development of ITC technologies, non-linear and complex problems have challenged the researchers. In the Extended Kalman Filter, the state distribution is approximated analytically by *Gaussian Random Variables* using a first order linearization of non-linear systems. New efforts such as *Dual EKF* have found better ways to approximate this transformation. The *Dual EKF* forms the basis of a second-order approximation with superior performance to that of the ordinary *EKF* (see section 3.3.1 or Bar-Shalom et al [7] for more detail).

ITC - Information Technology and Communication - Information Technology (IT) or Information Technology and Communication (ITC) is the technology required for information processing. This includes the use of computers and software to convert, store, protect, process, transmit, and retrieve information from any source.

ITS - Intelligent Transport Systems - Vehicle Safety and Control Systems which include advanced information processing using computers, technologies and management strategies in an integrated manner to improve the safety, capacity and efficiency of transportation systems.

KF - Kalman Filter - A probabilistic state estimation/control recursive feedback filter, which uses Gaussian Distributions to describe the estimate of some hidden state.

KLT - Kanade-Lucas-Tomasi Algorithm - This feature tracker is a texture based algorithm first proposed by Lucas and Kanade and later on generalized by Tomasi and Kanade [140], Birchfield [14].

MMSE - Minimum Mean Square Error - A method used to calculate the *minimum mean square error* which is the smallest value of *mean square error*.

MSE - Mean Square Error - A method used to calculate the *mean square error*. This is a criterion for an estimator: the choice is the one that minimizes the sum of squared errors due to bias and due to variance.

OCS - Object Coordinate System - A system used to describe an object in reference to its location in space.

RMS - Root Mean Square - A method used to calculate the *root mean square*. *RMS* is a statistical measure of the magnitude of a varying quantity. It can be calculated for a series of discrete values or for a continuously varying function. *RMS* is the square root of the mean of the squares of the values.

ROF - Robust Optical Flow - Computer Vision and Image Understanding Problem of finding robustly a *2D* velocity field which describes the apparent motion in the image sequence. The *ROF* was intensively studied by Black [17].

SfM - Structure from Motion - A problem in Computer Vision. Input data consists of two-dimensional features of an object, which are tracked in a sequence of images. The desired output data is the scaled three-dimensional locations of the features, as well as the scaled three-dimensional translation and rotation of the object.

SRUKF - Square Root Unscented Kalman Filter - A Kalman Filter developed by van der Merwe and Wan in [146] which implements the *UKF* to calculate the Cholesky Factorization at once during the initialization process.

UKF - Unscented Kalman Filter - A Kalman Filter which implements the *UT* to transform the covariance matrix during the propagation and observation stages. This makes it a better approximation technique compared to *EKF* and ordinary *KF*. The *UKF* and *EKF* differs mainly in the way in which *Gaussian Random Variables* are calculated. *GRV* are represented for propagating through system dynamics, Haykin [60], Wan and van der Merwe [154], Nelson [104] and Wan et al [156] or Section 3.3.2 of this Thesis.

UT - Unscented Transform - Developed by Julier et al [72], which attempts to apply a non-linear function to a Gaussian Distribution in a more accurate way by computing the Jacobian Matrices explicitly than the *Extended Kalman Filter* (see Subsection 3.3.2.1 of this Thesis for more information).

Chapter 1

Introduction

The world is characterized by dynamic three-dimensional processes, Shaogang et al [128], Maybeck [96], Bogdanov et al [23] and Wan and Bogdanov [152]. The ultimate aim of computer vision is to make it mimic the human brain's processing capacity, Ward and Heeman [157], Schalkwyk et al [124], Heeman and Damnati [61], Schmid and Barnard [125], Barfoof and D'Eleuterio [8] and Hosom and Cole [64]. A biological system uses various ways of recognizing *3D* information and the structure of the objects. For example, the process of reconstructing *3D* information by the human vision system is mainly based on stereo vision, Jebara et al [68], Samaras et al [121] and Lucas and Kanade [86]. Stereo vision is the process of constructing a *3D* model of a scene through processing two simultaneous *2D* images of this scene. The *3D* structure is obtained using the disparity between the two images produced by the two eyes, Erikson [46] and Jahne [67]. This technique can be applied in computer vision using multiple cameras.

Let us now imagine the use of a single camera. For example, given a video sequence obtained from a single camera, the goal is to recover *3D* structure of the object and its motion. Note that recovering *3D* structure usually requires motion, Wei and Hirzinger [158], Stewart and Langer [135], Samaras et al [121]. In this thesis we explain how one can use a single video camera (image sequence) to obtain *3D* information. This problem is known as the Structure-from-Motion (*SfM*) problem.

1.1 Vision for South Africa's Traffic System

The increase in traffic density and urban growth demands new ways of monitoring, management, control and optimization of traffic flow and acquisition of enough information for long-term planning of transport infrastructure, SADT [120], Wetteland and Lundebye [161], Papenfus [107] and Yiu et al [165].

The National Department of Transport is a government institution responsible

for defining road safety policies. Its 20-year strategy, defined in the White Paper of 1996, provides the guidelines on what role the different transport sectors should play in order to deliver a seamless transport system, Wetteland and Lundebye [161]. South Africa's road transport system is composed of three major parts: the **roads**, the **road users** and the **vehicles**. The behavior of road users is influenced by or linked to social, economic and technological issues. Road accidents continue to take the lives of travelers worldwide, particularly in South Africa. Traffic congestion is another serious and increasing problem in South Africa. The strategies for congestion relief generally fall in two classes: *supply and demand strategies*. Supply strategies are those that include the development of new infrastructures or the expansion of existing ones, so that the demand is better satisfied and/or the existing traffic system is improved. Demand strategies aim to reduce or minimize the travel demand.

The Third African Road Safety Congress (with the theme *Financing of Road Safety Actions*) was held in Pretoria, South Africa, April 14-17, 1997. The congress raised major problems concerning public safety, particularly on South Africa's roads, Wetteland and Lundebye [161]. The South African government took the responsibility of defining the correct safety policies. As a result, in 1999, South Africa published its **Action Agenda** : *"By 2020, transport in South Africa will meet the needs of freight and passenger customers for accessible, affordable, safe, frequent, high quality, reliable, efficient and seamless transport operations and infrastructure. It will do so in a constantly upgrading, innovative, flexible and economically and environmentally sustainable manner"*. The South African Department of Transport is presently engaged in activities geared towards the achievement of this Action Agenda goal by importing and developing transport technologies, SADT [120].

Intelligent Transport Systems *(ITS)* are designed to make more efficient use of existing road capacity by managing and controlling traffic flow with real-time traffic information. *ITS* include Advanced Traffic Management Systems *(ATMS)*, Advanced Traveler Information Systems (*ATIS*), and Automated Vehicle Control Systems (*AVCS*). The successful implementation of *ITS* will depend not only on high-quality real-time information (transport networks), but also on *qualified manpower*, Wetteland and Lundebye [161], and Papenfus [107]. Therefore, there is a need to develop national traffic surveillance systems. The National Department of Transport's White Paper of 1996 makes it clear that the use of cutting-edge technology is to be promoted: *"As part of the overall long-term vision for the South African transport system, transport infrastructure will $\cdots$ incorporate technological advances which promote and enhance the role of transport in the economy and development"*, SADT [120]. South Africa is therefore interested in developing a traffic monitoring system.

The South African Traffic Counting Network began in 1984 with a pilot study on the 600 km route N3 between Johannesburg and Durban, SANRAL [123]. In 1985, the National Transport Commission expanded the Traffic Counting Network to other counting stations. According to SANRAL [123] *"The traffic information is generally gathered by means of the Traffic Event Logger (*TEL*). The* TEL *receives signals from inductive loops installed beneath the road surface. The TEL records ⋯, speed (to the nearest km/h), length (to the nearest tenth of meter), the chassis height (low, medium, high) and the lane number. The speed of each vehicle ⋯ is calculated of time the second loop is occupied. Where traffic counting station are equipped with loops only, the vehicle classification is deducted from the vehicle length and the chassis height. This allows a distinction to be made among five types of vehicles, ⋯ long trucks (usually a truck-tractor plus a semi-trailer and full trailer combination)."*. These systems provide statistical information about traffic, i.e. vehicle counting, volume and density. In 2004, the Counting Network consisted of 796 counting stations where 381 are permanent and 415 secondary.

1.2 Problem Statement

The traffic surveillance in South Africa is mainly based on *TEL*, which falls when there is a higher traffic volume due to congestion.

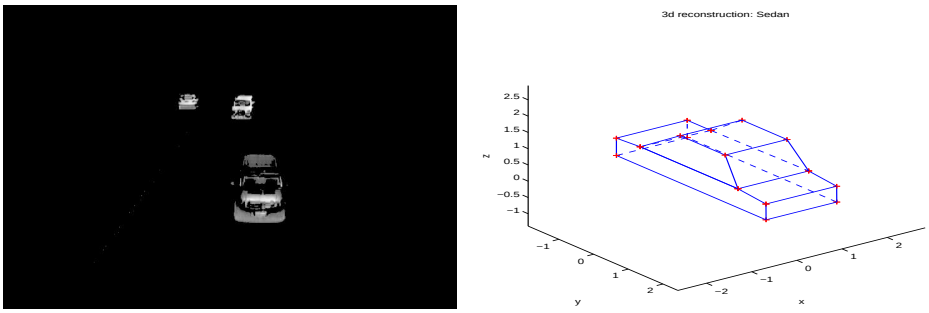
This study aims to develop a video-based traffic monitoring system which (i) can be used in higher traffic volume and (ii) potentially detects incidents by verifying the congestion or stopped objects on the lane.

1.3 Practical Problems in Intelligent Transportation Systems (ITS)

Intelligent Transportation Systems integrate hardware and software technologies to address and alleviate transportation congestion problems. As a result, traffic can be directed to other routes, thereby providing faster and more efficient routes for travelers. Optimizing the existing highway system is done through implementation of the Automated Traffic Surveillance and Control System *(ATSCS)*, which requires a lot of effort and creates several challenges. The robustness of *AVCS* also depends on outdoor weather conditions, vehicles' shadows, partial or full occlusion and system calibration.

1.4 Review of Road Monitoring Systems

In general, traffic surveillance provides the information needed for all of the other applications, including road monitoring systems, De Papenfus [107], van Niekerk [148], De Moor [44], Coifman and Beymer [35]. Most metropolitan areas use loop detectors and closed circuit television ($CCTV$) for traffic surveillance. The remainder use other methods of surveillance, such as radar, lasers, or video image processing.



(a) (b)

Figure 1.1: *(a) A feature-based approach detects and tracks prominent features. (b) A model-based approach matches features to the vehicle model.*

Road traffic monitoring research can be divided in two main groups:

- Model-based road traffic monitoring and
- Feature-based road traffic monitoring.

In the model-based approach, wireframe models are used to represent the geometry of the traffic scene, Lou and Tan [85]. The feature-based approach relies on motion detection to segment moving regions, Setchell [127] and Inigo [66]. In general, the idea is to identify a number of features of the object that can be tracked over a long sequence of video frames. These features should be selected in such a way that the object can be reconstructed from them. There are different ways of selecting and tracking features; a particularly useful one is referred to in Kanade et al [14], as well as Lucas and Kanade [86], Toshihiko and Kanade [142] and Truco and Verri [144].

1.4.1 Model-based Road Traffic Monitoring

The aim of the model-based approach is to derive an object's *3D* pose by mapping the image data to corresponding model descriptions, Lou and Tan [85]. It uses vehicle geometry to locate vehicles in an image frame and track them over consecutive frames. The process is mainly based on updating the background, which is

constantly updated to accommodate changing weather conditions and lighting. The model-based models encounter difficulties when there is congestion and occlusion, Badenas and Filiberto [6], Rittscher et al [117].

3D-based approaches uses knowledge of *3D* geometry to segment image regions corresponding to a variety of image region types, i.e. vehicle surfaces, Ferryman et al [49] and Lou and Tan [85]. For instance, Ferryman et al [49] uses features for both the background and vehicles, as a global eigenspace representation. The general disadvantage of *3D* models lies in the detailed geometric vehicle models, and the fact that they can be computationally complex.

An important example is the Koller System developed at the University of Karlsruhe, Koller et al [81]. In the first step, the Koller's System identifies clusters of coherently moving image features. Then the axis orientation of each cluster is defined as the initial values for the vehicle's position. The next step is to use a *3D* polyhedral model (see Figure 1.1b) of the vehicle projected into the image. Koller has proposed the *3D* generic vehicle models which are parameterized by twelve length parameters. The motion of the vehicle is associated with the model motion. Koller et al [81] and Koller et al [80] give more information about this.

It is important to point out that in this thesis model-based systems are not investigated.

1.4.2 Feature-based Road Traffic Monitoring

Feature-based motion detection finds and tracks prominent features from one frame to other, Birchfield [15]. The features are identified in the first few frames. The second step is to find which features correspond to each other in subsequent frames. This procedure is known as matching features in consecutive frames. The detection and tracking of prominent features is crucial in order to recover the *3D* information, as explained in the next chapters.

The feature-based systems which track only object features that are correctly identified are not computationally expensive by modern standards, but their cost depends on the number of features to be tracked. In addition, the features may disappear in long video sequences due to occlusion or lighting transitions.

The motion detection can be achieved either by frame differencing, Milan et al [98], or feature-based motion detection, Robert et al [119] and Harris and Stephens [57].

Next some road traffic monitoring systems developed or under development are reviewed.

1.4.3 Some Research on Road Traffic Monitoring Systems

Because of its complexity, road traffic monitoring has received a great deal of attention from several researchers, Coifman and Dhoorjaty [36], Coifman and Ergueta, [37], Coifman and Beymer [35] and Coifman and Yang [38]. *3D* tracking models, ASV [3], region-based tracking models, Paragio and Diriche [108] and Hager and Belhmeur [54], and contour-based tracking models, Yilmar and Shah [164], are some examples where researchers apply different strategies to solve traffic monitoring problems.

1.4.3.1 University of Tokyo

In 1973, the University of Tokyo built a system, called *sample point*, to detect the presence of a vehicle in the scene, Takaba [136]. By positioning a sample point in each lane of traffic it is possible to perform a vehicle count. When a vehicle passes, the sample point gets occluded and the system detects the presence of a vehicle. The system can determine the speed of the vehicle by placing two sample points a known distance apart in each lane and measuring the time delay between the triggering of the first and the second sample point in the lane. More contributions in this area come from other researchers including Matsubara et al [95] and Dailey et al [43].

1.4.3.2 University of Manchester

Early in the 1980's, researchers at the University of Manchester, the University of Sheffield and the Institute of Science and Technology (UK) developed a joint project designed to use computer vision for road traffic monitoring. The aim of the system was to count the vehicles traveling along a bi-directional highway. The system uses a frame differencing method to extract moving objects. The background image is computed by taking a photo of the site without moving objects. The difference between the incoming image and the background results in moving objects which are counted manually. The first experiments with this system were not impressive, Indigo [66]. Yet, later experimental results have shown the system to be usable.

1.4.3.3 University of Bristol

The Advanced Computing Research Center uses motion detection based on frame differencing. In the 1990s researchers at the University of Bristol developed a system capable of tracking vehicles through complex sites, multiple lane vehicle counting and speed calculation, Ali and Dagless [1], Cruz et al [41] and Yu et al [165]. The segmented regions of motion are tracked from one frame to another, enabling the vehicle path and speed to be monitored. The background is updated using programmable

hardware. The prototype vision system uses tracking based on correspondence between visual features in the scene. The camera motion is computed using either the Kalman filter or particle filter in conjunction with a Monte Carlo sequential method. The system is flexible and computationally efficient to solve nonlinear problems, Pupilli and Calway [115].

Presently, research is focused on further extensions of the approach, particularly in the areas of feature initialization, robust simultaneous localization and mapping (SLAM), guided tracking using *3D* vehicle models, and the generation of super-resolution imagery from a roving camera, Calway and Mayol-Cuevas [28].

1.4.3.4 University College London

Hoose [63] has developed a traffic monitoring system called *IMPACTS* which operates at the macroscopic level. It provides a quantitative description of the spatial distribution of moving and stationary objects in the scene. The initial results using this system were presented in 1992 by Kelly [76]. The system splits up the image into a number of cells approximately the width of a lane and the length of a car. The cell can then be classified as either no object, a stationary object or a moving object. The clustering is done by looking at cells with a similar characteristic. Then the object detection is achieved by looking for a large number of strong edges within the cell. The system was tested using data obtained during summer and winter from a camera mounted ten meters above the road. The resulting cell classification correctness were between 88.2 to 99.5 percent depending on the weather, lighting and other traffic conditions, Kelly [76].

1.4.3.5 University of California, Berkeley

The researchers at the University of California developed a system which is able to both detect and track multiple vehicles using shape information for vehicle classification, Koller et al [82], [79] and Malik et al [92]. The system, whose results were first presented in 1994, employs both feature and shape trackers, Beymer et al [12]. The background is removed by frame differencing and updated using the Kalman filter. The initial features are obtained by using contour- or snake-based algorithms. Again, the Kalman filter is used to track the shape and the position of the contours of an object over consecutive frames. Another innovation is that occlusion is detected by depth ordering the regions of each contour, Koller et al [78].

1.4.3.6 California Institute of Technology

The researchers at Jet Propulsion Laboratory (*JPL*) of California Institute of Technology developed the Wide Area Detection System (WADS) which is able to detect,

count and measure the speed of moving vehicles. The vehicle speed is computed using three lines placed at known intervals along the road (see MPC-Combi in section 1.5 for a similar system setup). The system's development began in the 1980s and now has an error rate of less than four percent compared to human observation on vehicle counting, but it has an error rate of less than two percent compared to radar units under clear sunny conditions. This system (WADS) is not video-based as the system described in our study.

1.4.3.7 Urban Transit Institute - North Carolina Agricultural and Technical State University

The Urban Transit Institute developed *A Fiber Optic Intersection Traffic Control System Based on a Competitive Cumulative Momentum Model*, Yu et al [165]. The study focused on the feasibility of replacing the existing in-ground induction loops with above-ground traffic detection systems.

The in-ground loops at intersections are mainly based on induction loops embedded in pavements at intersection approaches. The system is activated by the magnetic field created by a vehicle passing over them. The systems have a short range and mainly perform vehicle counting.

This system is designed as a stand-alone, zone-traffic control system. The system consists of two fiber optic strips, one at a designated distance from the traffic light or intersection, and the other very close to the traffic light. The former is used to monitor vehicles entering the detection zone, and the latter those exiting the zone. Thus the total number of vehicles in the detection zone can be used as an indication of traffic congestion.

1.4.4 Commercial Systems

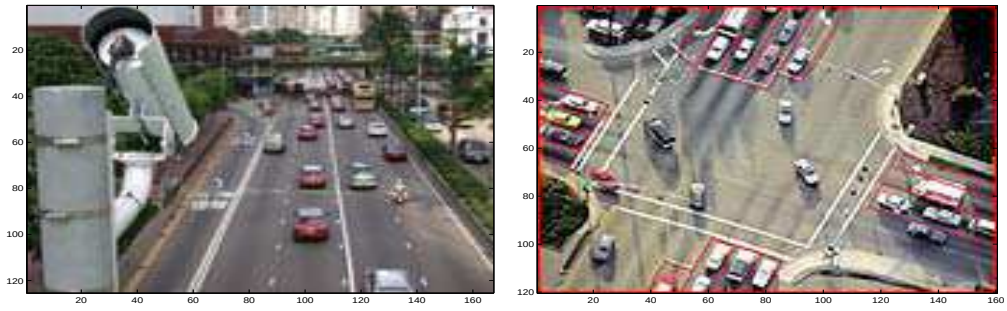
There are several commercial surveillance systems available. Surveillance traffic systems are used for both commercial purposes and scientific research, Samuel and Popoli [122], Cilliers [32], Pitman and Tenne [112], Coifman [34], Koller et al [78], [82] and Malik et al [92].

Nihan et al [106], Tierney et al [138] and De Moor [44] reported that commercial Video Image Processing Systems (*VIPS*) have problems with traffic congestion, high flow, occlusion and camera vibration.

Most commercial traffic monitoring systems are tripwire systems, Lorio [84] and Nihan et al [106] (see also Section 1.5 in this Thesis for detail). Some of these systems

use laser to detect the presence of a moving object and consequently compute its speed (see Section 1.5 for more information). The Combi-based system measures the speed of vehicles using three cables placed across the road giving some disadvantages compared to the laser system (see Section 1.5 for more details).

1.4.4.1 Autoscope



(a) (b)

Figure 1.2: *(a) Autoscope Solo Pro - a video traffic detection system and (b) MeadiaRoad System - allowing fully automatic surveillance.*

Autoscope is a commercial system developed at the University of Minnesota in 1984 and supported by both the Minnesota Department of Transport and the Federal Highway Administration of the United States of America. Presently, the system is used in cities worldwide, Michalopoulos [97] . This system consists of one or more *Autoscope* connected to cameras. The *Autoscope* output is fed to a supercomputer, enabling the control of the traffic, Michalopoulos [97]. *Autoscope* can be used in traffic management of highways, vehicle detection, incident detection for tunnel safety and video surveillance for safety and security. The autoscope technology provides traffic decision makers and managers with the tools to reduce roadway congestion and improve roadway planning. It is a real-time detection application improving highway speed data for traffic control centers and Internet information systems. The details of algorithms are not available but they appear to be based on either background subtraction or frame differencing.

1.4.4.2 The Camera and Computer Aided Traffic Sensor

The Camera and Computer Aided Traffic Sensor (*CCATS*) is another commercial system developed through co-operation between the Belgium Government, Delvonics Control Company and the University of Leuven, Hoose [62]. The Image Sensing System (*ISS*) provides vehicle detection for a variety of traffic applications such as urban control, highways, tunnels and bridges using the following guiding principles:

higher performance, reliability, user-friendly system and cost effectiveness. Vehicles are detected by frame differencing techniques using a threshold to cope with variation in lighting, shadows and noise. This system is mainly video-based.

1.4.4.3 Citilog

Citilog is another commercial system that uses video detection, automatic incident detection and traffic data collection, for which the details of algorithms are not available. The video detection uses image processing algorithms to extract important information from a standard sequence of video images. The automatic incident detection and traffic data collection are integrated in a single system called MeadiaRoad which aims to do the following, Citilog [33]:

- automatic flow detection,
- speed calculation,
- vehicle classification,
- traffic congestion detection,
- travel time calculation,
- detection of stopped vehicles,
- detection of vehicles going the wrong way,
- vehicle headway,
- detection of pedestrians, and
- detection of other objects than vehicles on the road

1.5 Current South African Road Monitoring and Law Enforcement System

There are places in South Africa, such as airports, where monitoring systems are clearly applied. Road monitoring systems are major factor in decision making [107]. The Traffic Event Logger is the most used system for traffic monitoring in South Africa [123]. There are four systems currently being used by South African traffic authorities for law enforcement, namely MPC-Combi, Digi-Cam, Pro-Laser and Multi-Combi. All of these systems have advantages and disadvantages, mostly related to cost and operation.

The MPC-Combi is a mobile combination system. It uses wet film, and has to be set up in a different location each time. The system is very heavy to carry and the

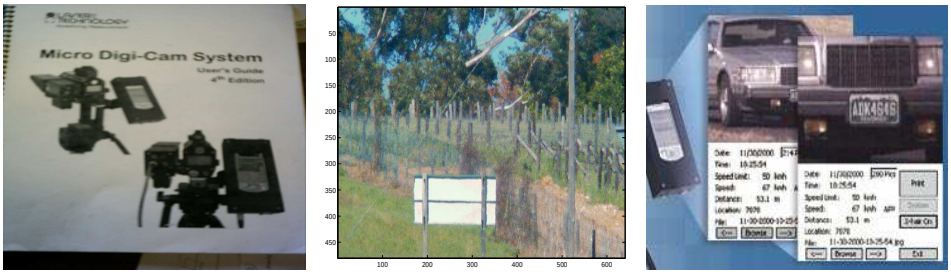
setup takes about thirty minutes. This system is potentially hazardous for officials during the setup period, since it involves laying cables across the road according to specific measurements. There must be three parallel cables across the road, 1.55 meters apart. The measurements have to be checked regularly. The system's operation also depends on weather conditions – it cannot be used in wet conditions. The cables are also affected by drivers' behavior. For instance, braking over the cables can disturb the position of the cables making it impossible to calculate the speed accurately. In order to identify the car, the system takes photos (mostly of the back of the car) and after the session the film has to be developed and printed. This system can only be used for monitoring one lane. If the official wants to monitor the opposite direction the system must be set up again.

The second type of system is the Multi-Combi (fixed-site system), which works in the same way as an MPC-Combi but it can monitor cars in more than one lane. This system is operated by a private company which downloads the data on a daily basis. The system is erected in an infrastructure (shown in Figure 1.3) and municipal traffic officers have to re-calibrate it every day. Another advantage of this system is that it is protected by a metal box which is water and bullet proof. The images are stored on a hardrive installed inside the system box.

Figure 1.3: *The fixed-site System is erected in an infrastructure - the metal box is water and bullet proof.*

The third type is the Pro-Laser system, which works with laser beams and a video camera. This system uses normal time-over-distance measurements to calculate speed. It is easy to operate compared to the ones mentioned above. The laser system is used to calculate the speed of the vehicle while the camera takes photos. The photos are used to recover the vehicle ID and they are processed by a

private company. After processing they are stored on CD and returned to the Traffic Department authorities. At this stage they are read-only (protected against editing).



(a) (b) (c)

Figure 1.4: *Laser System (a) Digi-Cam System, (b) Calibration zero velocity and (c) Photo and statistical information including car ID*

The Digi-Cam is the easiest system to set up (about five minutes) and operate. It uses a laser beam, a video camera and a PDA, as shown in Figure 1.4. This system is water proof, so it can be used in inclement weather conditions. Its biggest disadvantage is that the system must be re-calibrated every hour. This process is no automatic, but only takes 2 to 3 minutes. Another disadvantage of this system is that it has no automatic focus. So the officer has to choose the focus and it is set up to capture a vehicle at no less than 100 meters. Most of the above systems are imported and cost between $R350\,000$ and $R600\,000$, Magaia and Fourie [89]. The maintenance of the hardware and software is done by a South African company.

1.6 Our Approach

In this thesis we use algorithms for object clustering, tracking and *3D* reconstruction to develop a practical *video-based traffic monitoring system*. The system should provide information about the traffic, such as how fast cars are moving (speed) and car density (counting). Our approach allows for

- noise modeling,
- correct feature matching between different frames, and
- the use of a single camera to obtain correct *3D* information for system calibration and speed calculation.

The thesis discusses problems related to

- background extraction,

- feature selection,
- feature tracking,
- *3D* reconstruction, and
- system calibration and speed computation.

The solution of a *3D* reconstruction problem poses several questions, including:

1. What information does one need from the sensor in order to recover the *3D* structure and motion?
2. How to extract that information (tracking problem)?
3. Which is the best model to represent this information?
4. Which is the best algorithm to recover the structure?

The above questions are hard to answer. The difficulty of the first question lies in the number of output images needed to obtain the *3D* reconstruction. For example, in order to recover the *3D* information using data obtained from *Robust Optical Flow*, eighty frames are needed. The second question is difficult because features must be identified, selected and tracked over the entire sequence. But feature selection can be a difficult procedure. We need to balance the accuracy of the model with the cost of working with the model. Given any mathematical model, the computer approximates the solution, which can be computationally expensive. Obviously, the answer to the last question depends heavily on the precision of the reconstruction. The worse the state estimate, the less information is available (see Jahne [67] and Sclaroff and Pentland [126] for more detail). The computational complexity of the problem depends on the specific mathematical models and statistical methods used, Forsyth et al [50], Bouthemy and Lalande [24], Alireza and David [2], Penio [109], Kambhatla [74], Hampel et al [55], Huber [65], Barnett and Lewis [9] and Vempela and Blum [149].

The reasonably realistic perspective projection model leads to a nonlinear problem with all the associated complications (see for example, Azarbayejani and Pentland [5], Azarbayejani [4], Soatto and Perona [131] or Toshihiko and Kanade [142]). The simpler orthographic model assumes an infinite focal length and is therefore only appropriate for situations where the object is far from the image plane (thereby approximating an infinite focal length). On the other hand it leads to a linear problem that is reasonably straightforward to solve using the singular value decomposition (SVD) (cf. Muller et al [103], Costeira [40], Branco and Costeira [25] and Tomasi and Kanade [140], [141]). In our experiments, the orthographic model was not suitable for real data computation because it does not model observation and process noise

and assumes objects far away from the camera.

We have developed and tested an algorithm which uses the Kanade-Lucas-Tomasi feature tracker, Harris Corner Detector and various optical flow tracking methods (Black [17], Zhao [167], Barron et al [10], Yu and Dyer [166]) and a nonlinear Kalman filter extension, called the *Unscented Kalman Filter* (*UKF*), to solve the *3D* reconstruction problem. We examine different clustering algorithms to separate the objects. Specifically, we find that the k-means algorithm performs well (see Muller [101] for more information).

1.7 Comparison of Our System with Others

The main differences between our system and those discussed earlier are (i) that ours is not a model-based approach and (ii) ours uses of *Structure-from-Motion* (SfM) approach, Magaia et al [91]. We employ the *Unscented Kalman Filter* for *3D* reconstruction. The applied model recovers both the shape and speed of the object. The system calibration consists of determining the scale factor which is applied to get the real dimensions of the shape and speed of the object. Any error in the reconstruction will lead to incorrect system output, particularly for speed and vehicle dimension.

Another important difference between our system and others described above is that we use dense optical flow as a feature tracker. The only difficulty in using dense optical flow is that it is computationally expensive. The more features we have to track the more computer memory is needed.

1.8 Thesis Outline

Chapter 2 is an introduction to *Shape-from-Motion* (*SfM*)in a simple setting. It explains in detail the *3D* reconstruction of rigid objects using the singular value decomposition and discusses why it fails to solve some real-world problems. The numerical results are presented in Section 2.3.

Chapter 3 discusses the Kalman filter. In this thesis, the Kalman filter is used as the *3D* reconstruction algorithm. Section 3.3.2 discusses in detail the nonlinear Kalman filter which we use for nonlinear approximations, the *Unscented Kalman Filter* (*UKF*). We also present the *UKF* algorithm. Some experimental results are presented in this chapter as well.

Chapter 4 explains in detail the dense optical flow algorithms. It explores the robustness of the method and explains why it can be used to track the features. We give optical flow algorithms.

Chapter 5 discusses the *SfM* problem using a Kalman filter approach. The first section introduces the problem of extracting *3D* camera coordinate system given only a sequence of *2D* projections. In the second section, we discuss the structure and motion models as Kalman filter problems. In the last section of this chapter we give the initialization model.

Chapter 6 discusses the other aspects needed to build a traffic monitoring system. We give some requirements, and discuss two methods for background removal, namely histogram modeling of background and frame-by-frame difference. We discuss the clustering techniques used for an object's separation following the *3D* reconstruction of the shape and motion of rigid objects and the speed computation. We then describe tracking rigid objects using robust optical flow. System calibration and some experimental results are also discussed in this chapter.

Chapter 7 discusses the major results obtained in our experiments, starting with synthetic data and progressing to real data.

Conclusions and recommendations are given in **Chapter 8**.

Appendix A is dedicated to general mathematical background such as singular value decomposition, important definitions used in this thesis and a brief description of the quaternion.

Appendix B provides detailed descriptions of the equipment used and software directories.

Appendix C gives details of how the user can run the demonstration programs either to reproduce some of the results contained in this thesis or to process his/her own data. It also explains how to use most of the programs, including where to make some crucial changes to the settings. This chapter is helpful to anyone who would like to test this system or process other data.

We also provide a **CD** and a **DVD** containing the code and some data used in this thesis.

1.9 Our Contribution

Our contributions to the field of surveillance systems lie in (i) the reduction of the number of frames required to obtain the *3D* information (This system converges after 80 frames.), (ii) the fact that even if the clustering algorithm fails to correctly classify an object it is still able to recover its shape, (iii) the use of dense optical flow enabling us to track a larger number of features, (iv) the employment of the *Unscented Kalman Filter* enabling us to accurately recover the speed of that object, (v) the system calibration is achieved by determining the *3D* scale factor and (vi) the fact that our system infers the object structure directly from the video sequence, and does not fit a model to the data. Model fitting can be added as a post-processing stage using the recovered *3D* data.

Chapter 2

The Singular Value Decomposition and 3D Reconstruction

In Computer Vision, the *SfM* problem has been used to determine the motion of the camera or object independently from the structure of the scene, Peter et al [111], Sclaroff and Pentland [126], Kingsbury and Margarey [77] and Shaogang et al [128]. In this thesis we investigate the problem of recovering the *3D* structure of an object from a video sequence of a rigid object.

It is well known that *3D* objects can be reconstructed from stereo pairs of *2D* images. For example, the human *3D* vision system relies heavily on the presence of *two* eyes. *3D* information is also present in a *sequence* of images, such as a video film or, indeed, human vision with one eye closed. Imagine that a person moves about and that we capture views of objects from different angles. These views certainly contain *3D* information. Again, if we only have access to a single *2D* image projection, we can rarely recover its structure.

Researchers were able to solve some problems of missing or occluded information using other sources, such as the light source interacting with object surfaces. The reader is advised to consult Wei and Hirzinger [158], Samaras et al [121], Brooks et al [26] and Stewart and Langer [135] for more information.

In this chapter, the factorization method is discussed. By assuming an orthographic camera model, the *SfM* problem can be reduced to a matrix factorization problem. This can be solved using the singular value decomposition (*SVD*), one of the most important matrix factorizations in linear algebra.

We describe and implement the method first introduced by Kanade and Tomasi [75] for a single rigid object and then modified by Poelman and Kanade [113]. In 1994, Kanade and Morita extended this algorithm to the so-called sequential fac-

torization method, Toshihito and Kanade [142]. Finally, in 1995 the factorization method was generalized to multiple objects, Costeira [40]. The simplest case in which we can apply the factorization method is where we have a single object (as in Figure 2.4). In the case of multiple objects the situation is complicated due to the fact that it is not known a *priori* which features belong to what object, Costeira [40]. But this can be resolved as discussed in Subsection 2.2.2.

In this chapter we describe the orthographic camera model. We describe how the *SfM* problem can be solved in Section 2.2. A number of practical experiments are presented in Section 2.3. For a short description of the types of noise used in the experiments see Appendix A.

2.1 Orthographic Camera Model

The *3D* reconstruction of moving objects are complicated because *3D* objects require a large number of feature points for their reconstruction. In addition, they can also contain large areas that can appear and disappear due to occlusion, making it difficult to track. The problem we have to solve is an inverse one—from the projections on the video film we need to reconstruct the original object. For this we need to know how the projection was created in the first place; i.e., we need to have a model of the camera. A reasonable model is the so-called pinhole camera with its perspective projection; see, for example, Hartley and Zisserman [58]. Although it is possible to do a reconstruction using this camera model, the problem becomes nonlinear, requiring advanced techniques such as the Kalman filter or one of its nonlinear extensions discussed later in this Thesis (also see Azarbayejani and Pentland [5], Strom et al [134]).

Kanade and coworkers [139], [140], [141], [142] developed a useful simplification algorithm. Consider objects far from the camera or a focal length that approaches infinity. In that case, to a good approximation, the object is projected onto the film parallel to the Z-axis, which is the optical axis, as shown in Figure 2.1. This is known as an orthographic projection.

2.2 The 3D Reconstruction of Objects

The above simple camera model allows one to solve the problem using linear methods, in essence, by using a matrix factorization, Tomasi and Kanade [141], Magaia and Herbst [90]. Assume that only one object is observed and, importantly, that the object is a rigid body. The importance of the rigid body assumption lies in the fact that the only movements that rigid bodies can perform are translations and rotations.

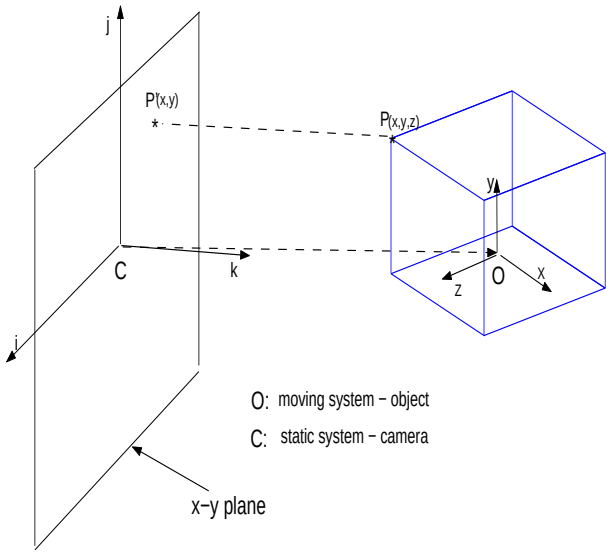


Figure 2.1: *Camera and object in different coordinate systems.*

Translations are easily described by means of a *3D* translation vector. Rotations, on the other hand, can be described by rotation matrices. Noting that an arbitrary rotation of an object in *3D* has three degrees of freedom—it rotates through an *angle* of rotation (one degree of freedom) around its normalized *axis* of rotation (another two degrees of freedom)—one looks for 3×3 matrices with three degrees of freedom. Since 3×3 real, orthonormal matrices R, defined by $RR^T = I = R^T R$, have exactly three degrees of freedom, they are perfect candidates. Since $\det(R) = \pm 1$ in general, there is some ambiguity. A rotation matrix is one with $\det(R) = 1$—the negative sign involves a reflection as well as a rotation. In general one has to be able to construct the rotation matrix given an axis and angle of rotation or, conversely given a rotation matrix, find the axis and angle of rotation. There is a beautiful result due to Rodrigues (see, for example, Faugeras [48]) that enables one to do this quite easily.

Since we are dealing with a rigid object, a reference frame (x, y, z) fixed to the object is described in terms of n feature points (from here on referred to just as features)

$$\boldsymbol{p}_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix}, \quad s = 1, \ldots, n. \tag{2.1}$$

The *shape* matrix, describing the shape of the object in the object coordinate system,

is therefore written as

$$S = \left[\begin{array}{cccc} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \ldots & \boldsymbol{p}_n \end{array} \right]. \tag{2.2}$$

The features are really all that is known about the object, and for a full reconstruction some kind of interpolation or subdivision is required. In the case of a wire-frame cube as in Figure 2.1, the eight vertexes are natural features, allowing a perfect reconstruction. We now allow the object to rotate; in subsection 2.2.1.2 translation is handled similarly, using homogeneous coordinates. The features of a rigid body do not change relative to a coordinate system fixed to the object. Now, imagine that the object rotates with respect to another coordinate system fixed to the camera. Each feature is projected onto the film plane of the camera, and as the object rotates, the features are projected onto slightly different positions in consecutive frames of the video sequence. The information about the *3D* structure of the object is contained in the offsets of the features in the different frames. Consider one of the features, $\boldsymbol{p}_s$, of the object, represented in the *object coordinates* fixed to the object as given by (2.1). If the rotation of the object coordinate system with respect to that of the camera at the time of the tth video frame is given by the rotation matrix

$$R_t = \left[\begin{array}{c} \boldsymbol{i}_t^T \\ \boldsymbol{j}_t^T \\ \boldsymbol{k}_t^T \end{array} \right], \tag{2.3}$$

where

$$\left[\begin{array}{c} \boldsymbol{i}_t^T \\ \boldsymbol{j}_t^T \\ \boldsymbol{k}_t^T \end{array} \right] = \left[\begin{array}{ccc} i_{xt} & i_{yt} & i_{zt} \\ j_{xt} & j_{yt} & j_{zt} \\ k_{xt} & k_{yt} & k_{zt} \end{array} \right].$$

Consider the first frame. The transformation to the camera coordinate system is given by R_1. So each point $\boldsymbol{p}_s$ is transformed as

$$\boldsymbol{p}_{1s}^c = R_1 \boldsymbol{p}_s. \tag{2.4}$$

Similarly, for each time t we write

$$\boldsymbol{p}_{ts}^c = R_t \boldsymbol{p}_s. \tag{2.5}$$

We will exploit the fact that rotation matrices are orthonormal, i.e., $R_t^T R_t = I$. This means that the rows (and columns) are orthogonal and normalized. In particular,

$$\begin{aligned} \boldsymbol{i}_t^T \boldsymbol{i}_t &= 1 = \boldsymbol{j}_t^T \boldsymbol{j}_t = \boldsymbol{k}_t^T \boldsymbol{k}_t, \\ \boldsymbol{i}_t^T \boldsymbol{j}_t &= 0 = \boldsymbol{i}_t^T \boldsymbol{k}_t = \boldsymbol{j}_t^T \boldsymbol{k}_t. \end{aligned} \tag{2.6}$$

This gives six relationships between the nine elements of the rotation matrix R_t, leaving three degrees of freedom. These three degrees of freedom are exactly what is needed to describe the angle and normalized axes of rotation. Since our simple camera projects objects onto the film by translation parallel to the Z-axis of the camera coordinate system, a point $\boldsymbol{p}_{ts}^c = [x_{ts}, y_{ts}, z_{ts}]^T$ in camera coordinates projects onto $\boldsymbol{p}_{ts} = [x_{ts}, y_{ts}]^T$ on the film. This can be written as

$$\boldsymbol{p}_{ts} = H_Z \boldsymbol{p}_{ts}^c = H_Z R_t \boldsymbol{p}_s, \tag{2.7}$$

where H_Z is the orthographic projection matrix

$$H_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.8}$$

If we do this for all n features, we measure the following features in frame t

$$W_t = \begin{bmatrix} x_{t1} & x_{t2} & \cdots & x_{tn} \\ y_{t1} & y_{t2} & \cdots & y_{tn} \end{bmatrix} = H_Z R_t S, \tag{2.9}$$

with S given by (2.2). If we write

$$M_t = H_z R_t = \begin{bmatrix} i_{xt} & i_{yt} & i_{zt} \\ j_{xt} & j_{yt} & j_{zt} \end{bmatrix}, \tag{2.10}$$

equation (2.9) becomes

$$W_t = M_t S. \tag{2.11}$$

All that remains to be done is to collect all the observations over f frames into a single observation matrix W:

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_f \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ y_{11} & y_{12} & \cdots & y_{1n} \\ \vdots & & & \vdots \\ x_{f1} & x_{f2} & \cdots & x_{fn} \\ y_{f1} & y_{f2} & \cdots & y_{fn} \end{bmatrix}. \tag{2.12}$$

From the expression for a single frame (2.11) we obtain

$$W = MS, \tag{2.13}$$

where M is the *motion* matrix assembled from (2.10)

$$M = \begin{bmatrix} M_1 \\ \vdots \\ M_f \end{bmatrix} = \begin{bmatrix} i_{x1} & i_{y1} & i_{z1} \\ j_{x1} & j_{y1} & j_{z1} \\ i_{x2} & i_{y2} & i_{z2} \\ j_{x2} & j_{y2} & j_{z2} \\ & \vdots & \\ i_{xf} & i_{yf} & i_{zf} \\ j_{xf} & j_{yf} & j_{zf} \end{bmatrix}. \tag{2.14}$$

Thus, given the observation matrix W, the mathematical problem is to find M and S; i.e., the problem is reduced to factorizing the observation matrix into motion and shape matrices.

Let us take another look at the structures of the motion and shape matrices, M and S. Note that their maximum rank is 3. This means that the maximum rank of the observation matrix W should also be 3. In practice, however, W is obtained by tracking features over a number of frames, introducing all kinds of tracking and measurement errors, with the result that the actual rank of the $2f \times n$ matrix W may be higher than 3. We should therefore try to extract the best possible rank-3 approximation of W. This is where SVD is particularly effective. It is possible for S to have rank 1, 2, or 3, where 3 indicates a full *3D* object. Rank 2 corresponds to a planar object, rank 1 to a line. For instance, for a planar object we can always choose the object coordinate systems such that the plane of the object is given by $z = 0$. The motion matrix M can have rank 2 or 3. Rank 3 indicates a full *3D* rotation, and rank 2 describes a rotation around the Z-axis. Since a rotation around the Z-axis presents only one side of the object to the film, the *3D* object is perceived as a planar object. All this information is encoded in the rank of the observation matrix W. That rank may not be exact, primarily due to measurement errors. One needs a strategy to determine the effective rank of W; and the key element is SVD.

2.2.1 Reconstruction of a Single Object

As mentioned above, we investigate an approach that is based on an *orthographic camera model*. In practice, this means that all information about the depth is lost, i.e. it is not possible to recover the distance of the object from the image plane (cf. Fukunaga [51], Gonzalez and Wood [53], Yiu [29] and Kanade an Tomasi [75]). In the next subsection we describe the reconstruction considering an object subject to rotation only. The combination of translation and rotation is considered in Subsection 2.2.1.2.

2.2.1.1 Pure Rotation

In section 2.2 we showed that we need to factorize the observation matrix W into a $2f \times 3$ motion matrix M and a $3 \times n$ shape matrix S. For this purpose, we calculate the full SVD of the observation (measurement) matrix

$$W = U\Sigma V^T. \tag{2.15}$$

Note that the decomposition of matrix W is sensitive to noise, especially when the motion between subsequent frames is small, Jebara et al [68]. Luong et al [87] and Luong and Faugeras [88] investigate different methods as well as their stability.

As pointed out at the end of the previous section, the rank of W is in general higher than 3. Thus we choose the 3 largest singular values σ_1 through σ_1 and form the rank-3 approximation of W

$$\widetilde{W} = U_+\Sigma_+V_+^T, \tag{2.16}$$

where

$$\Sigma_+ = \mathrm{diag}\,(\sigma_1, \sigma_2, \sigma_3) \tag{2.17}$$

and U_+ and V_+ consist of the first 3 columns of U and V, respectively. The expression (2.16) for $\widetilde{W}$ is therefore the best rank-3 approximation of W in the sense of (A.11). Thus, we can write

$$\widetilde{W} = \widetilde{M}\widetilde{S}, \tag{2.18}$$

where

$$\widetilde{M} = U_+\Sigma_+^{\frac{1}{2}} \quad \text{and} \quad \widetilde{S} = \Sigma_+^{\frac{1}{2}}V_+^T. \tag{2.19}$$

Although $\widetilde{W}$ is written in the correct form, $\widetilde{M}$ and $\widetilde{S}$ are not yet the desired motion and shape matrices. Note that the factorization is not unique. Indeed, let A be any invertible 3×3 matrix; then

$$\widetilde{W} = \left(\widetilde{M}A\right)\left(A^{-1}\widetilde{S}\right) \tag{2.20}$$

is another possible factorization. We now exploit the freedom provided by A to get the motion matrix M in the correct form; i.e., one chooses A in such a way that

$$M = \widetilde{M}A \tag{2.21}$$

has all the properties of the motion matrix. Recall that the motion matrix M consists of orthogonal rows in the sense of (2.6). Setting $Q = AA^T$, the orthonormality equations (2.6) are written with the help of equations (2.10), (2.19) and (2.21) as

$$\begin{aligned} \widetilde{\boldsymbol{m}}_{2t-1}^T Q \widetilde{\boldsymbol{m}}_{2t-1} &= 1 = \widetilde{\boldsymbol{m}}_{2t}^T Q \widetilde{\boldsymbol{m}}_{2t}, \\ \widetilde{\boldsymbol{m}}_{2t-1}^T Q \widetilde{\boldsymbol{m}}_{2t} &= 0, \quad t = 1, \ldots, f, \end{aligned} \tag{2.22}$$

where $\widetilde{\boldsymbol{m}}_k^T$ is the kth row of $\widetilde{M}$. Since Q is a symmetric 3×3 matrix it has 6 unknown entries that can be determined in a least squares sense from the $3f$ equations (2.22). Once Q is calculated, another factorization is required to obtain A. This problem is also undetermined. Indeed, A has 9 unknown entries, and we only know the 6 elements of Q. So, 3 elements of A remain undetermined. The reason for this is that we are free to choose the object's coordinate system—since we do not yet allow translation, we are free to choose its rotational orientation. Thus, through factorization of Q we find A only up to an arbitrary rotation. More precisely, if $Q = \widetilde{A}\widetilde{A}^T$, then $Q = (\widetilde{A}R)(\widetilde{A}R)^T$ is another factorization for any rotation matrix R. Any 3×3 rotation matrix has precisely 3 arbitrary elements; these are the 3 undetermined elements of A. We therefore have a choice, and one possibility is to require the object to be oriented as in the first frame. Since Q is symmetric it can be diagonalized with an orthonormal matrix U_Q (see, e.g., Golub and van Loan [52]),

$$Q = U_Q \Lambda_Q U_Q^T,$$

where one again notes that the eigenvalues of $Q = AA^T$ are all nonnegative. If $\widetilde{A} = U_Q \Lambda_Q^{\frac{1}{2}}$, then $A = \widetilde{A}R^T$ for any 3×3 rotation matrix R (the transpose of R is used for convenience). Choosing A such that the object is oriented as in the first frame, R follows from

$$\boldsymbol{w}_1 = H_Z R \widetilde{A}^{-1} \widetilde{S}, \tag{2.23}$$

where $\boldsymbol{w}_1$ denotes the first two rows of the observation matrix (2.12). Looking more carefully at (2.23) we note that it consists of six linear equations (the left-hand side is a 2×3 matrix) in six unknowns ($H_Z R$ selects the first two rows of the 3×3 matrix R). Since $\widetilde{A}$ and $\widetilde{S}$ are known, we can solve the resulting 6×6 linear system. For a rotation matrix this then determines the last row uniquely. The final step is to calculate the motion and shape matrices from

$$M = \widetilde{M}\widetilde{A}R^T \quad \text{and} \quad S = R\widetilde{A}^{-1}\widetilde{S}. \tag{2.24}$$

2.2.1.2 Rotation and Translation

For simplicity we have assumed up to now that we are dealing with pure rotations. Now we incorporate translations into the model. The ideas as well as much of the formalism are the same as for pure rotations if we use the homogeneous coordinates. If the translation between frame $t-1$ and frame t is $\boldsymbol{c}_t$, then the observation is given by

$$\boldsymbol{w}_t = H_Z \left(R_t S + \boldsymbol{c}_t \right). \tag{2.25}$$

In homogeneous coordinates each feature point is written as

$$\boldsymbol{p}_s = \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix}, \tag{2.26}$$

resulting in a homogeneous shape matrix of the form[1]

$$S = \begin{bmatrix} x_1 & x_2 & & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & & z_n \\ 1 & 1 & & 1 \end{bmatrix}. \tag{2.27}$$

The advantage of homogeneous coordinates is that the translation can now be incorporated into the motion matrix as

$$M = \left[\begin{array}{ccc|c} i_{x1} & i_{y1} & i_{z1} & c_{x1} \\ j_{x1} & j_{y1} & j_{z1} & c_{y1} \\ & \vdots & \vdots & \\ i_{xf} & i_{yf} & i_{zf} & c_{xf} \\ j_{xf} & j_{yf} & j_{zf} & c_{yf} \end{array}\right] := [M_r \;\; \boldsymbol{c}], \tag{2.28}$$

with the result that the observation matrix is again written as a product $W = MS$. We now proceed along the same lines as with pure rotation, and again the SVD of W is calculated. In this case the maximum rank of W in the non-degenerate case is 4, Magaia and Herbst [90]. Accordingly, we extract the best rank-4 approximation of W and factorize W into $\widetilde{M}$ and $\widetilde{S}$. The calculation of A requires a bit more care. For W of rank 4, A is 4×4 and can be partitioned as

[1] Consult Hartley and Zisserman [58] for an excellent account of the applications of projective geometry to computer vision.

$$A = \begin{bmatrix} A_r & \boldsymbol{a}_c \end{bmatrix}, \tag{2.29}$$

where A_r is a 4×3 matrix related to rotation and $\boldsymbol{a}_c$ is a vector related to translation. More specifically, $\widetilde{M}A_r = M_r$, where M_r is the rotational part of the motion matrix in (2.28). Thus A_r is again obtained from the constraints (2.22), but now with $Q = A_r A_r^T$. In order to determine $\boldsymbol{a}_c$ we note that we are free to choose the translational orientation of the object's coordinate system. Since the centroid of the features maps to the centroid of the observations under an orthographic projection, the centroid of the observations is

$$\overline{\boldsymbol{w}} = \begin{bmatrix} \frac{1}{n}\sum_{s=1}^{n} X_{s1} \\ \frac{1}{n}\sum_{s=1}^{n} Y_{s1} \\ \vdots \\ \frac{1}{n}\sum_{s=1}^{n} X_{sf} \\ \frac{1}{n}\sum_{s=1}^{n} Y_{sf} \end{bmatrix}. \tag{2.30}$$

If

$$\overline{\boldsymbol{p}} = \frac{1}{n}\sum_{s=1}^{n} \boldsymbol{p}_s \tag{2.31}$$

is the centroid of the features, then

$$\overline{\boldsymbol{w}} = \widetilde{M}\begin{bmatrix} A_r & \boldsymbol{a}_c \end{bmatrix}\begin{bmatrix} \overline{\boldsymbol{p}} \\ 1 \end{bmatrix}. \tag{2.32}$$

The simplest choice for the origin of the object coordinate system is $\overline{\boldsymbol{p}} = 0$. Consequently,

$$\overline{\boldsymbol{w}} = \widetilde{M}\boldsymbol{a}_c. \tag{2.33}$$

This over-constrained problem for $\boldsymbol{a}_c$ is again solved with the least squares method. We now summarize the algorithm, incorporating both rotation and translation.

2.2.1.3 Algorithm for a Single Object

1. Select n features on the object and track their images on the film over f frames.
2. Assemble the x and y coordinates of the features on the f frames into a single $2f \times n$ observation matrix W.
3. Compute the SVD of $W = U\Sigma V^T$.

4. Determine the effective rank of W from the magnitudes of the singular values. Theoretically, for non degenerate motion $\mathbf{rank}(W) = 4$, thus we use the best rank-4 approximation of W, namely, $\widetilde{W} = U_+\Sigma_+V_+^T$, when we keep only the final 4 singular values.

5. Construct $\widetilde{M} = U_+\Sigma_+^{\frac{1}{2}}$ and $\widetilde{S} = \Sigma_+^{\frac{1}{2}}V_+^T$.

6. Calculate the symmetric matrix Q as the least squares solution of (2.22).

7. Diagonalize Q, i.e., $Q = U_Q\Lambda_Q U_Q^T$.

8. Set $\widetilde{A} = U_Q\Lambda_Q^{\frac{1}{2}}$ as defined in (2.29).

9. Calculate rotation matrix R and translation vector $\boldsymbol{a}_c$ as described in Section 2.2.1.2.

10. Compute $M = \widetilde{M}\widetilde{A}R^T$ and $S = R\widetilde{A}^{-1}\widetilde{S}$.

2.2.2 Reconstruction of Multiple Objects

In previous subsections, we discussed the reconstruction of a single object. Now, we consider the possibility of reconstructing multiple objects, moving independently (see Figure 2.14). The measurement matrix W contains information about all the moving objects and the main problem is to identify the different objects, where we do not even known the number of objects. Once we have grouped the features into their corresponding objects, the structure of W becomes,

$$W = \begin{bmatrix} O_1 & & & & \\ & O_2 & & 0 & \\ & & \ddots & & \\ & & & O_k & \\ & & & & \ddots \\ & & 0 & & O_N \end{bmatrix}, \tag{2.34}$$

where O_k represent the kth object and N is the number of objects. Once W has been sorted into this form, each sub-matrix can be factorized as described above for the case of a single object. The main difficulty is therefore to develop a suitable sorting algorithm.

Costeira developed an efficient, robust procedure and the reader is referred to [40] for the details. He noted that, if the objects experience different rotations and translations, their elements (features) will show a different energy as defined in equation (2.35) below. The energy will be non-zero if the features belong to the same object and zero if they belong to different objects. From the factorization of W, we

get $W = U\Sigma V^T$. Next, we define a new matrix $Q^* = VV^T$ and call it the *shape interaction matrix*. He showed that this matrix has the following properties. Each element of Q^*_{ij} is given by

$$Q^*_{ij} = \begin{cases} \mathbf{m_i}^T \left(A^{-1}\Sigma^{1/2}V^T\right)\mathbf{m_j} & \text{if } \mathbf{m}_i \text{ and } \mathbf{m}_j \text{ belong to the same object k} \\ 0 & \text{if } \mathbf{m}_i \text{ and } \mathbf{m}_j \text{ belong to different objects,} \end{cases} \tag{2.35}$$

where A is a semi positive-definite-symmetric matrix and $\mathbf{m}_i$, $\mathbf{m}_j$ are columns of V. The function $F_{ij} = (Q^*_{ij})^2$, indicating the link (i, j), is called the energy function and it is used to identify different objects.

2.2.2.1 Factorization of Multiple Objects

Consider a simple case in which we have two objects. Then the observation matrix

$$W = [O_1|O_2]\,. \tag{2.36}$$

Applying the idea from the case of a single object we can decompose W in two big matrices (a shape and a motion matrix). Then the observation (measurement) matrix W will consist of two shape blocks (S) and two motion blocks (M) each with different dimensions. Each observation (measurement) matrix of an object can be factorized as

$$W_k = U_k\Sigma_k V_k^T = M_k S_k \tag{2.37}$$

where k indicates the object in reference. Changing notation (2.36) can now be rewritten as

$$W^* = [M_1|M_2]\begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix}. \tag{2.38}$$

The main difficulty at this stage is to recover the shape space V^* obtained by the singular value decomposition of non-canonical W^*.

Recall (2.35), the resulting shape interaction matrix is computationally unique and each feature pair is associated with the appropriate object. This equation is used to sort the features into corresponding objects. We refer this process to a block-diagonalizing of W^*. Consult Costeira [40] for more information about the procedure.

Numerical illustration: We consider two vertices of a cube tracked over 6 frames and we incorporate both rotation and translation components. Recall Section 2.2.1.2, the rank of W^* is 8 corresponding to the two objects moving independently. For the

first object, we consider an angle of rotation φ_0 and a translation vector $\mathbf{t}_0$ and we rotate and translate 6 times. For the last object we change both rotation and translation slightly. The observation (measurement) matrix of the form $2F \times n$ is

$$W^* = \begin{bmatrix} 1.0853 & -1.0853 & 1.1664 & -1.1664 \\ 1.5939 & -0.5939 & 1.6834 & -0.6834 \\ 2.1113 & -0.0939 & 2.2019 & -0.1986 \\ 2.6373 & 0.4149 & 2.7219 & 0.2878 \\ 3.1719 & 0.9323 & 3.2431 & 0.7758 \\ 3.7151 & 1.4584 & 3.7653 & 1.2650 \\ 0.9961 & -0.9961 & 0.9842 & -0.9842 \\ 1.9945 & 0.0055 & 1.0778 & -0.8778 \\ 2.9892 & 1.0037 & 1.1623 & -0.7795 \\ 3.9800 & 1.9983 & 1.2375 & -0.6891 \\ 4.9668 & 2.9892 & 1.3035 & -0.6069 \\ 5.9494 & 3.9762 & 1.3602 & -0.5329 \end{bmatrix}, \tag{2.39}$$

where $F = 6$ is the number of frames and $n = 4$ is the number of features (vertice coordinates).

Recalling (2.16), the decomposition of matrix W^* using the *singular value decomposition* gives

$$V^* = \begin{bmatrix} -0.0781 & 0.2835 & -0.4574 & 0.4752 \\ -0.1364 & 0.2927 & -0.2674 & 0.3460 \\ -0.1955 & 0.3009 & -0.0764 & 0.2169 \\ -0.2554 & 0.3079 & 0.1156 & 0.0878 \\ -0.3160 & 0.3137 & 0.3085 & -0.0412 \\ -0.3775 & 0.3183 & 0.5023 & -0.1701 \\ -0.0686 & 0.2485 & -0.4028 & -0.5455 \\ -0.1570 & 0.1049 & -0.3186 & -0.3970 \\ -0.2448 & -0.0398 & -0.2373 & -0.2483 \\ -0.3321 & -0.1854 & -0.1590 & -0.0993 \\ -0.4187 & -0.3320 & -0.0838 & 0.0500 \\ -0.5046 & -0.4796 & -0.0115 & 0.1996 \end{bmatrix}, \tag{2.40}$$

$$\Sigma^* = \begin{bmatrix} 13.7337 & 0 & 0 & 0 \\ 0 & 4.8026 & 0 & 0 \\ 0 & 0 & 3.1462 & 0 \\ 0 & 0 & 0 & 0.0863 \end{bmatrix} \tag{2.41}$$

and

$$U^* = \begin{bmatrix} -0.8235 & -0.1048 & -0.2537 & -0.4965 \\ -0.3544 & -0.6626 & 0.4062 & 0.5201 \\ -0.4410 & 0.7364 & 0.2279 & 0.4597 \\ 0.0419 & 0.0881 & 0.8478 & -0.5213 \end{bmatrix}. \tag{2.42}$$

The shape interaction matrix Q^*, in which only the first three digits are presented, is given by

$$Q^* = \begin{bmatrix} 0.522 & 0.380 & 0.239 & 0.096 & -0.047 & -0.191 & 0.000 & 0.000 & -0.002 & -0.001 & 0.000 & 0.004 \\ 0.380 & 0.296 & 0.210 & 0.124 & 0.038 & -0.049 & 0.001 & 0.000 & 0.0007 & 0.000 & 0.000 & 0.000 \\ 0.239 & 0.210 & 0.182 & 0.153 & 0.124 & 0.094 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -0.002 \\ 0.096 & 0.124 & 0.153 & 0.181 & 0.209 & 0.238 & 0.000 & 0.000 & 0.001 & 0.000 & 0.000 & -0.003 \\ -0.047 & 0.038 & 0.124 & 0.209 & 0.295 & 0.381 & -0.002 & 0.000 & 0.002 & 0.002 & 0.000 & -0.003 \\ -0.191 & -0.049 & 0.094 & 0.238 & 0.381 & 0.525 & -0.005 & 0.000 & 0.003 & 0.003 & 0.002 & -0.002 \\ 0.000 & 0.001 & 0.000 & 0.000 & -0.002 & -0.005 & 0.526 & 0.382 & 0.238 & 0.095 & -0.047 & -0.189 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.382 & 0.295 & 0.209 & 0.123 & 0.038 & -0.047 \\ -0.002 & 0.000 & 0.000 & 0.001 & 0.002 & 0.003 & 0.238 & 0.209 & 0.180 & 0.151 & 0.123 & 0.096 \\ -0.001 & 0.000 & 0.000 & 0.000 & 0.002 & 0.003 & 0.095 & 0.123 & 0.151 & 0.180 & 0.209 & 0.239 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.002 & -0.047 & 0.038 & 0.123 & 0.209 & 0.295 & 0.381 \\ 0.004 & 0.000 & -0.002 & -0.003 & -0.003 & -0.002 & -0.189 & -0.047 & 0.096 & 0.239 & 0.381 & 0.525 \end{bmatrix}. \tag{2.43}$$

Note that the size of the shape interaction matrix Q^* should follow from the size of V^*. Because of numerical approximation, it is seen that in matrix Q^* there are still some small entries that should be zero, i.e. in the first line and column nine it is observed that the value is -0.002. It is possible to see from (2.43) that there are four blocks of six entries involved. The top left corner and the right bottom have significance entries and others are zeros. Each block with non-zero entries in (2.43) corresponds to one object and each non-zero entry corresponds to a feature in the measurement matrix W^*. The measurement matrix W^* is of the form

$$W^* = \begin{bmatrix} O_1 & 0 \\ 0 & O_2 \end{bmatrix}. \tag{2.44}$$

That means the objects' blocks are located in the diagonal, see (2.45). Using (2.35)

we get the sorted W^* of the form

$$W^* = \begin{bmatrix} 3.1719 & -1.0853 & 0 & 0 \\ 1.0853 & -0.5939 & 0 & 0 \\ 2.1113 & -0.0939 & 0 & 0 \\ 2.6373 & 0.4149 & 0 & 0 \\ 1.5939 & 0.9323 & 0 & 0 \\ 3.7151 & 1.4584 & 0 & 0 \\ 4.9668 & -0.9961 & 0 & 0 \\ 0.9961 & 0.0055 & 0 & 0 \\ 2.9892 & 1.0037 & 0 & 0 \\ 3.9800 & 1.9983 & 0 & 0 \\ 1.9945 & 2.9892 & 0 & 0 \\ 5.9494 & 3.9762 & 0 & 0 \\ 0 & 0 & 3.2431 & -1.1664 \\ 0 & 0 & 1.6834 & -0.6834 \\ 0 & 0 & 2.2019 & -0.1986 \\ 0 & 0 & 2.7219 & 0.2878 \\ 0 & 0 & 1.1664 & 0.7758 \\ 0 & 0 & 3.7653 & 1.2650 \\ 0 & 0 & 0.9842 & -0.9842 \\ 0 & 0 & 1.3035 & -0.8778 \\ 0 & 0 & 1.1623 & -0.7795 \\ 0 & 0 & 1.2375 & -0.6891 \\ 0 & 0 & 1.0778 & -0.6069 \\ 0 & 0 & 1.3602 & -0.5329 \end{bmatrix} \tag{2.45}$$

Then we apply the algorithm for a single object to each block matrix.

2.2.2.2 Algorithm for Multiple Objects

Given the measurement matrix W

1. Determine the number of objects $\Leftrightarrow$ the number of blocks using equation (2.35),
2. Block-diagonalize W by grouping/sorting the features corresponding to the same object,
3. Apply the algorithm for a single object to each block.

The experimental results of this subsection are presented in Section 2.3.

2.3 Experimental Results

In this section, the results of various experiments are discussed. These experiments include:

- Synthetically generated input, ranging from pure translation in all three axes, to pure rotation around all three axes, to various combinations of translation and rotation.
- A synthetically generated video sequence, processed with feature tracking software, and fed into the algorithm.

Unless otherwise indicated, we abbreviate Observation Noise Variance by (ONV). Next we present the results obtained when running the algorithm on pure synthetic data, synthetic data with Gaussian white noise[2] and on quasi-real data[3]. This will reveal whether or not the algorithm is sensitive to noise.

2.3.1 Procedures

First we create some mathematical objects, i.e. a cube, line (parabola), open envelope, vehicle model. Only six types of vehicles were considered: Hatchback, Sedan, Station Wagon, Mini-Van, Pick-Up and Long Truck. In the first experiment, we track eight features of a cube over fifty frames by rotating these features. Since translation was not included, the rank of the measurement matrix is three as was described in Section 2.2.1. In this case, the measurement matrix W has 16×50 entries, i.e. the eight vertices of the cube were tracked over hundred frames. The observation matrix is displayed graphically in Figure 2.2b where the movement of the 8 vertices over 50 frames is summarized. The reconstruction is shown in Figure 2.2c. Clearly, this reconstruction is perfect (compare with Figure 2.2a). Note that the cube is in its original orientation. Next we run the same experiment but this time we include the translation. Figure 2.3 shows the original, the observation and the *3D* reconstruction. Figure 2.4 shows the observed features and the reconstruction of a cube. Next, we test our algorithm using as many features as possible. In Figure 2.12a we plot the initial image (laboratory data) and the reconstruction is shown in Figure 2.12b. The features were mathematically tracked by applying the rotation and translation matrices 50 times over all axes. Since there is no tracking error the *3D* reconstruction of the face is perfect (see Figure 2.12b). Next, we introduce some

[2]Gaussian white noise is a good pdf approximation of many real-world situations and generates mathematically tractable models.

[3]We define the quasi-real data as the data obtained from objects created by POVRAY (Persistence of Vision Ray-tracer) and tracked by a feature tracker. Visit www.povray.org/ for more information.

noise into the observations and run the same experiments. The results shown in Figures 2.5 to 2.7 are testimony of the sensitivity of the algorithm to the noise. The algorithm is applied to different object types, such as objects' shapes. The results are illustrated in Figures 2.8 to 2.11. Again figure 2.10 shows that the algorithm is sensitive to the noise. We also tested the algorithm using quasi-real data. We generate a data cube using *POVRAY* with a realistic camera model. Then we apply the *KLT* tracker over that sequence. The results are shown in figure 5.8.

We have also applied the algorithm to the case of multiple objects moving independently. We consider three different objects moving simultaneously and independently in the viewing area. We create mathematical objects (cube, parabola and open envelope) and then we track their features (vertices) over 100 frames. The measurement matrix W has $n \times 100$ entries. Figure 2.14 shows the reconstruction of the shape of the three objects separately. The sorted measurement matrix of the three objects is graphically displayed in Figure 2.13b. Although the reference features belonging to the three objects are presented in color for convenience, the reader should keep in mind that it is one of the major tasks to classify the reference features into their respective objects, using for instance the algorithm described by Costeira [40]. The reconstruction of the shapes of the three objects is shown in Figure 2.13c. Although not physically feasible, note that the procedure does not encounter any difficulties if the three objects fall on top of each other.

2.3.2 Results

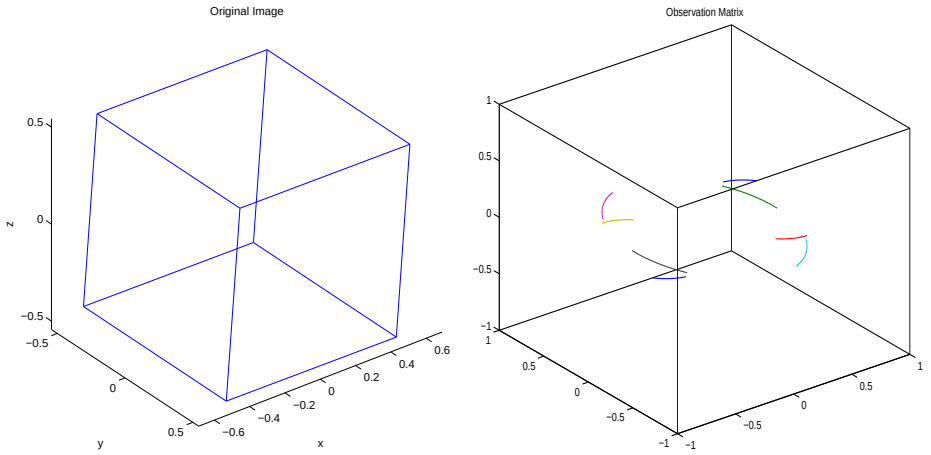
This subsection presents the results obtained in the experiments as described in previous sections. Again we refer to *ONV* as object's noise variance.

As the observation's noise variance (*ONV*) increased, the system becomes less stable. At about 20 percent of the data size, it is not able to recover the shape of the cube correctly (see figures 2.7).

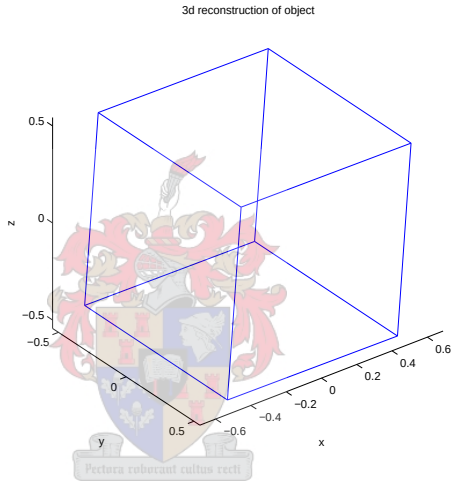
Now the algorithm is applied to the vehicle shapes and some Gaussian white noise is added until no we longer obtain an acceptable reconstruction.

2.4 Discussion

The factorization method works correctly when the features are correctly tracked under orthographic projection. We have tested the ability of the system to recover the three-dimensional information using various synthetic sequences. The experiments consisted of rotating objects over some of or all the axes and building the measurement matrix W. We have also considered a case where the objects were both rotated



(a) (b)



(c)

Figure 2.2: *Structure recovering of the cube under rotation without noise; (a) Original, (b) Observation and (c)* 3D *reconstruction*

and translated. The system was able to recover both shape and motion (see Figures 2.2 and 2.3). The algorithm was tested using different object models (cube, car and truck). Next the experiment is done with some white noise added to the data. Next we found that the algorithm is sensitive to noise, particularly when the observations noise variance is over twenty percent of the object's size (see Figure 2.7).

It is important to note that there is no error correction in this approach. Consequently, small initial errors can compound quickly over long sequences. Also, noise in the measurements of the feature positions can have a significant impact on the final results. Therefore, the use of models with error correction are much more ro-

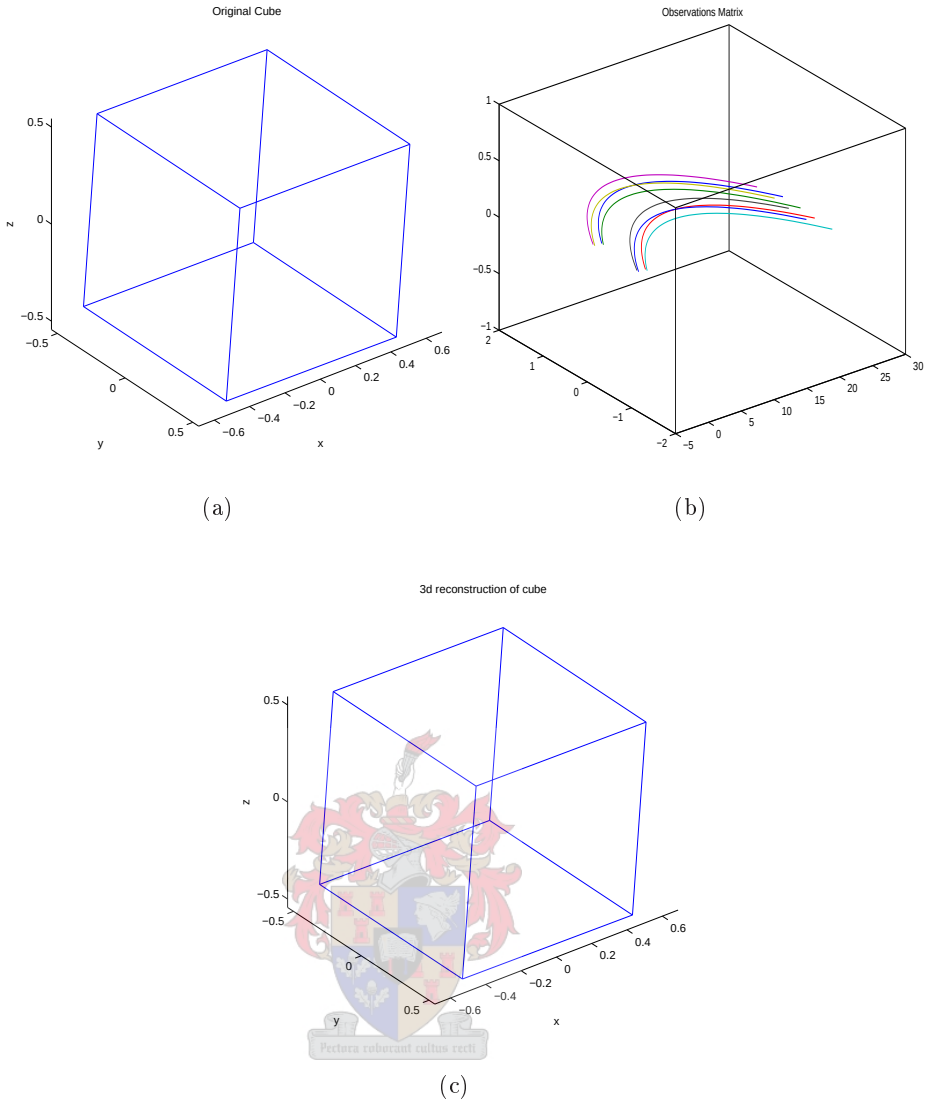


Figure 2.3: *Structure recovering of the cube under rotation and translation without noise; (a) Original, (b) Observation and (c)* 3D *reconstruction*

bust, as is shown in Chapters 3 and 5. So, the *SVD* algorithm is used as a baseline to compare with other systems.

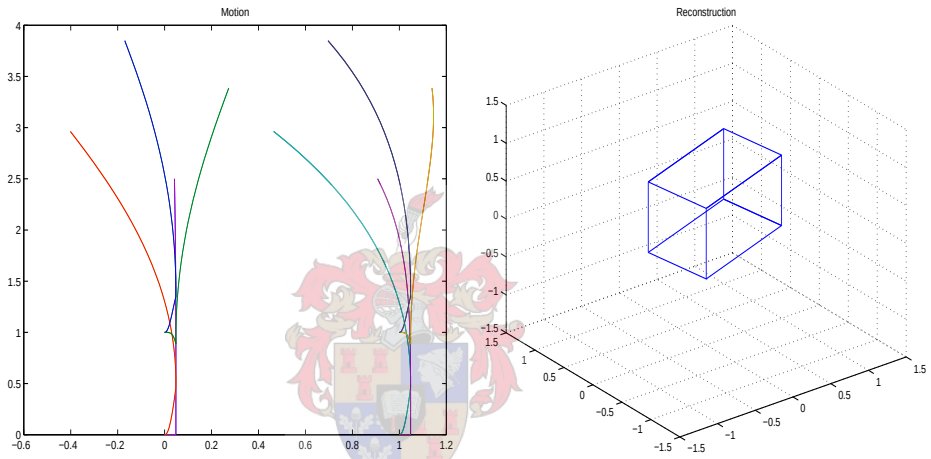
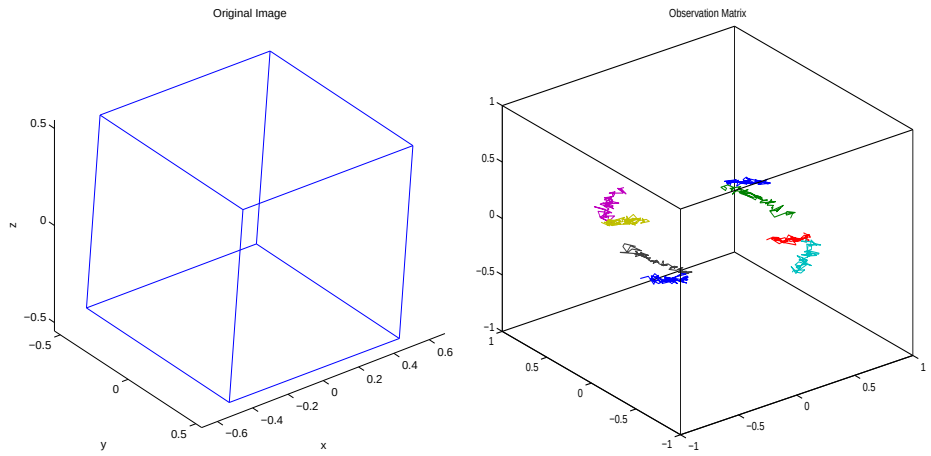
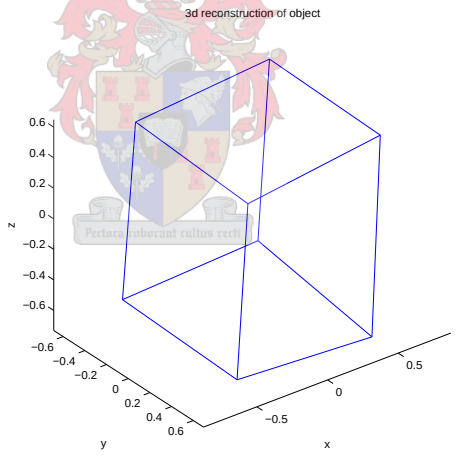


Figure 2.4: *SVD - experiment with synthetic data; (a) Observation and (b) Shape*

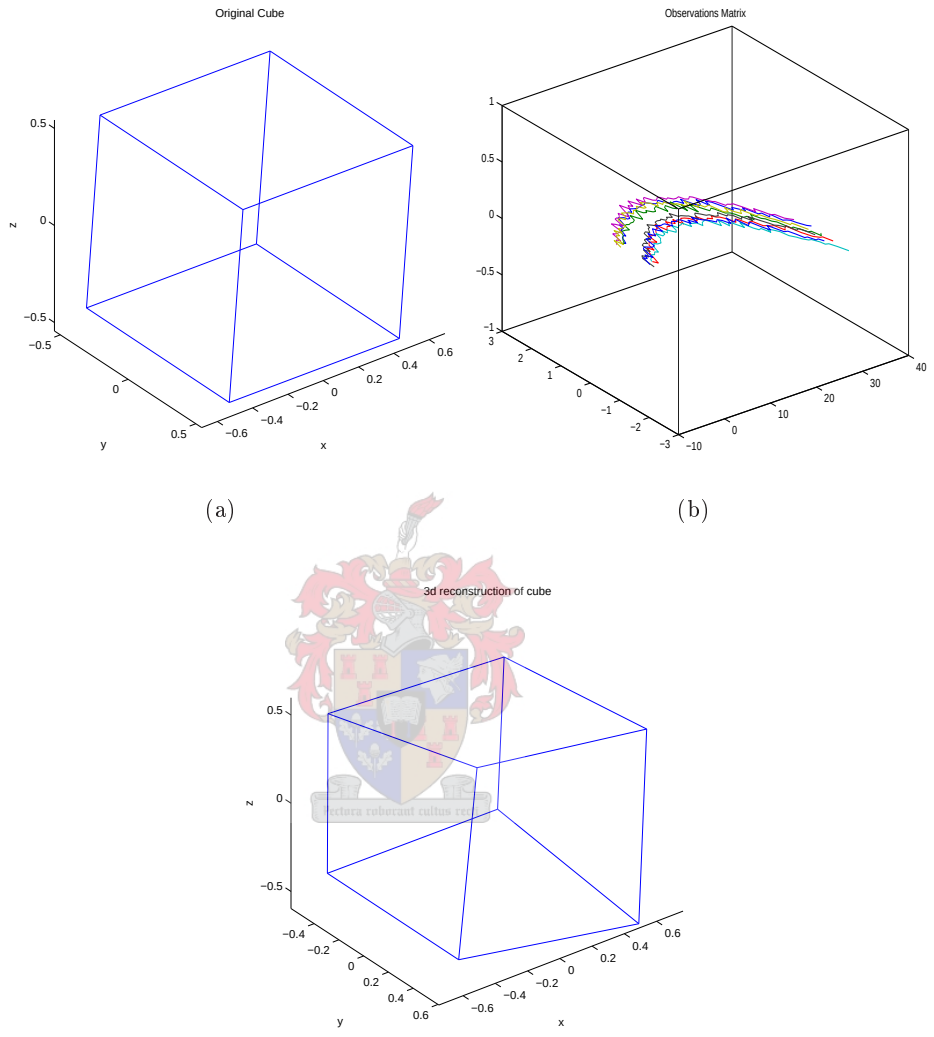


(a) (b)



(c)

Figure 2.5: *Structure recovering of the cube under rotation with ten percent of* ONV*: (a) Original, (b) Observation and (c)* 3D *reconstruction*



(a) (b)

(c)

Figure 2.6: *Structure recovering of the cube under rotation and translation with ten percent of* ONV*; (a) Original, (b) Observation and (c)* 3D *reconstruction*

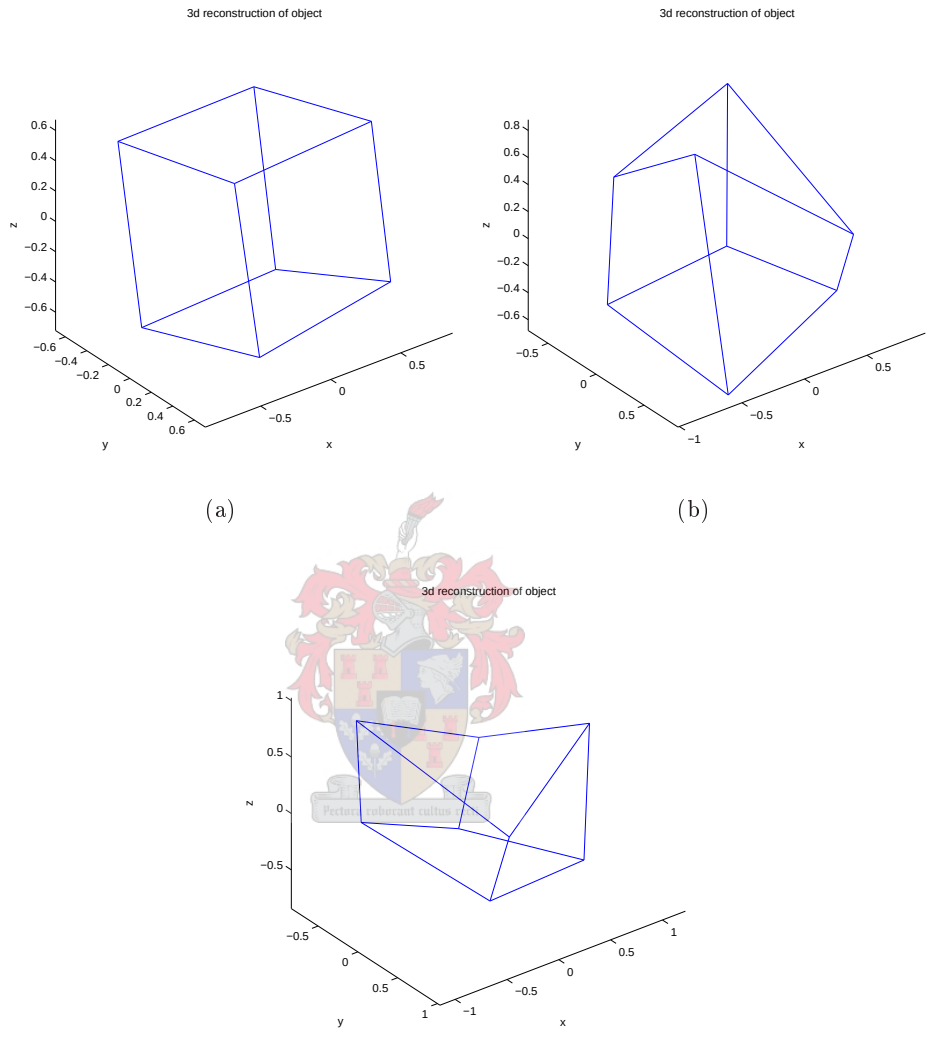
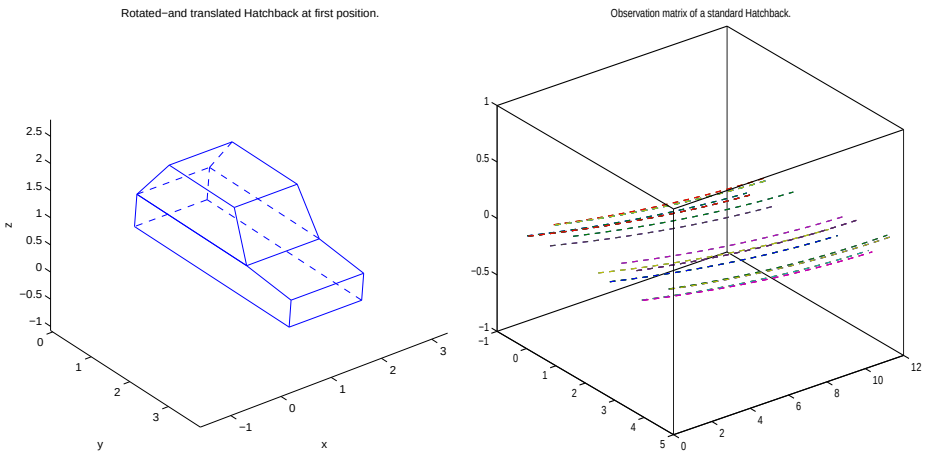
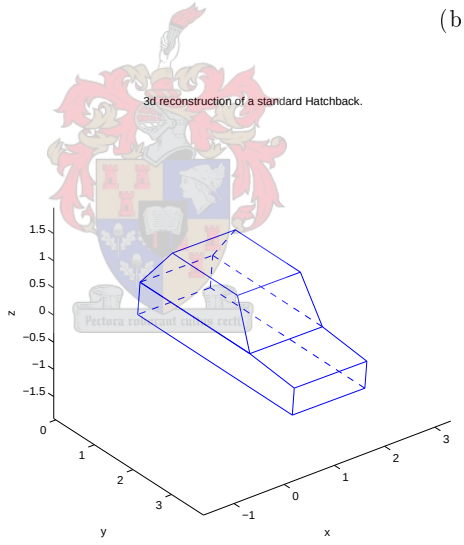


Figure 2.7: *Structure recovering of the cube under rotation with gradual increasing the level of* ONV*; (a) Ten, (b) Fifteen and (c) Twenty percent*

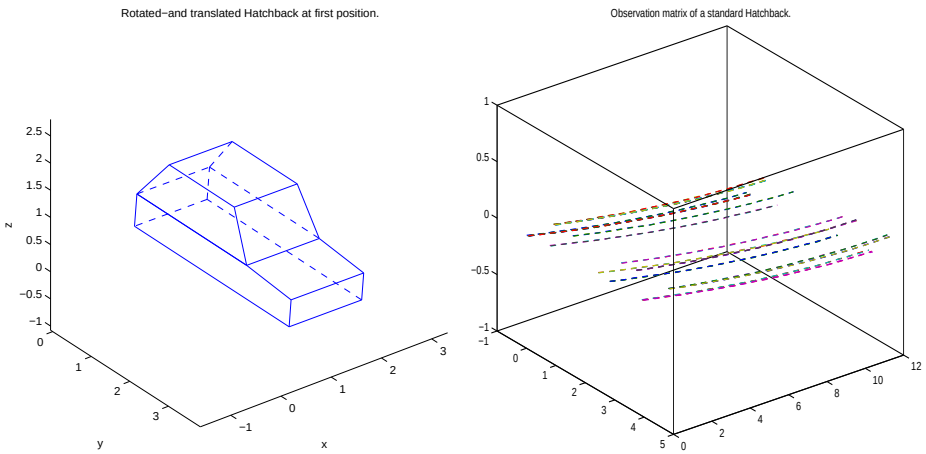


(a) (b)

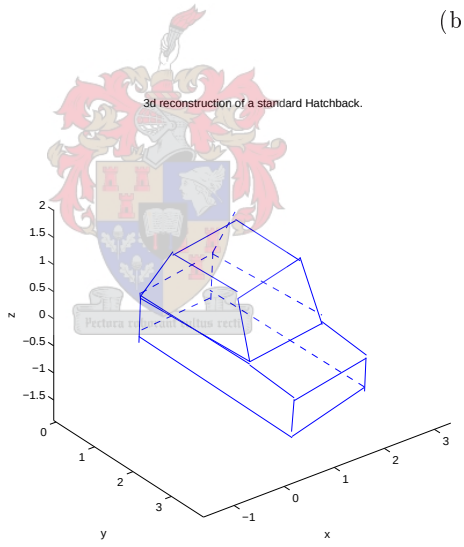


(c)

Figure 2.8: *SVD - Hatchback* $(4.4 \times 1.6 \times 1.4)$ *without noise: (a) Original, (b) Observations and (c)* 3D *reconstruction*

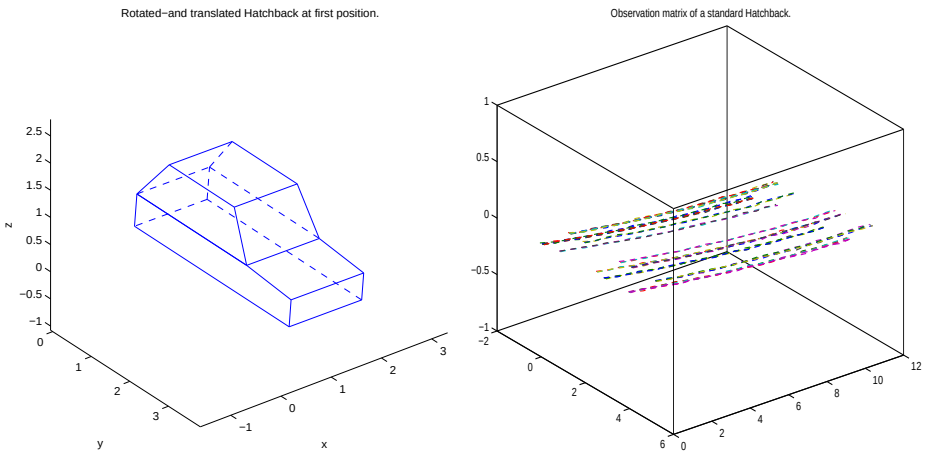


(a) (b)

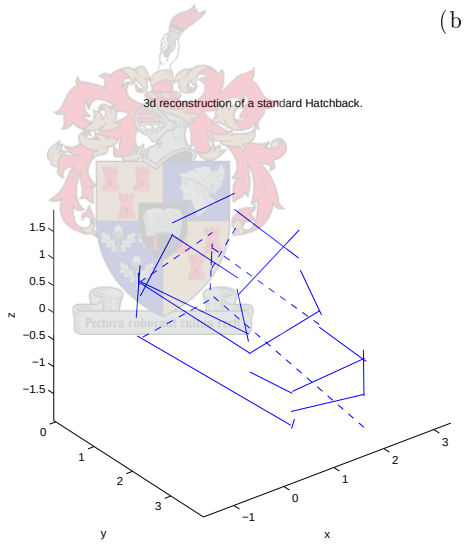


(c)

Figure 2.9: *SVD - Hatchback* ($4.4 \times 1.6 \times 1.4m^3$) *with* ONV$=$ *0.0001: (a) Original, (b) Observations and (c)* 3D *reconstruction*

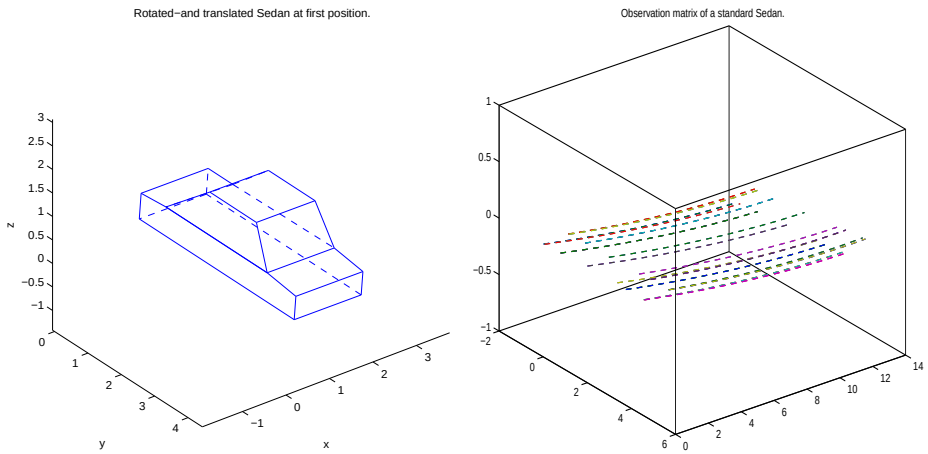


(a) (b)

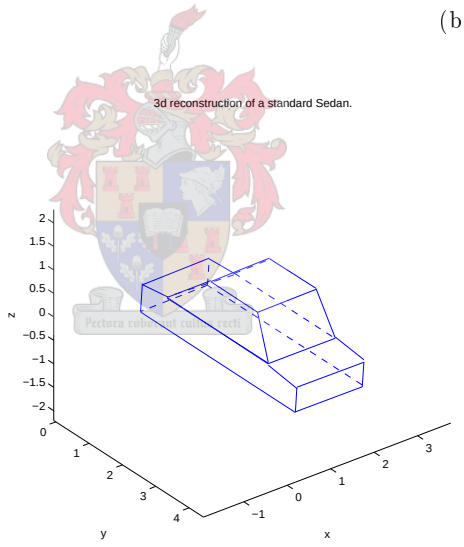


(c)

Figure 2.10: *SVD - Hatchback* ($4.4 \times 1.6 \times 1.4m^3$) *with* ONV$= 0.001$*; (a) Original, (b) Observations and (c)* 3D *reconstruction*



(a) (b)



(c)

Figure 2.11: *SVD - Sedan* ($4.4 \times 1.6 \times 1.4m^3$) *with* ONV$=$ *0.00001; (a) Original, (b) Observations and (c)* 3D *reconstruction*

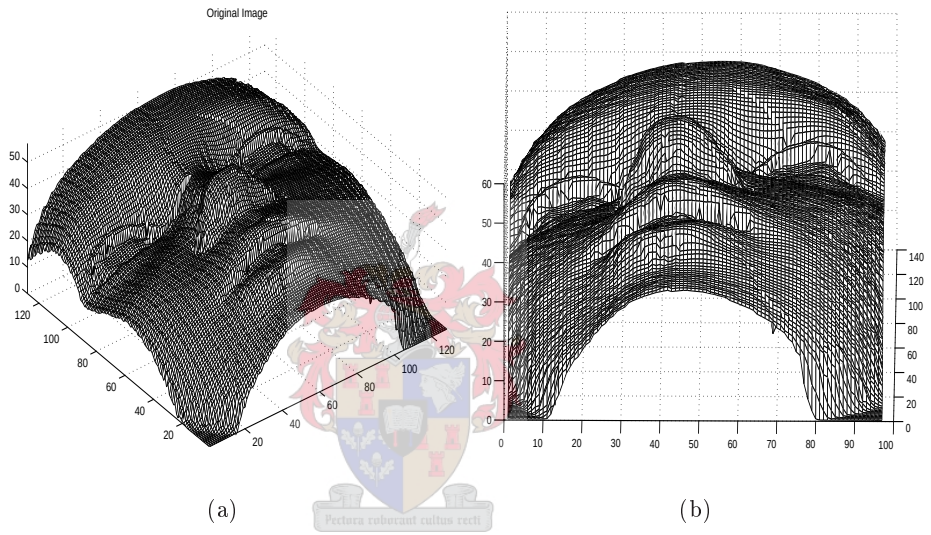
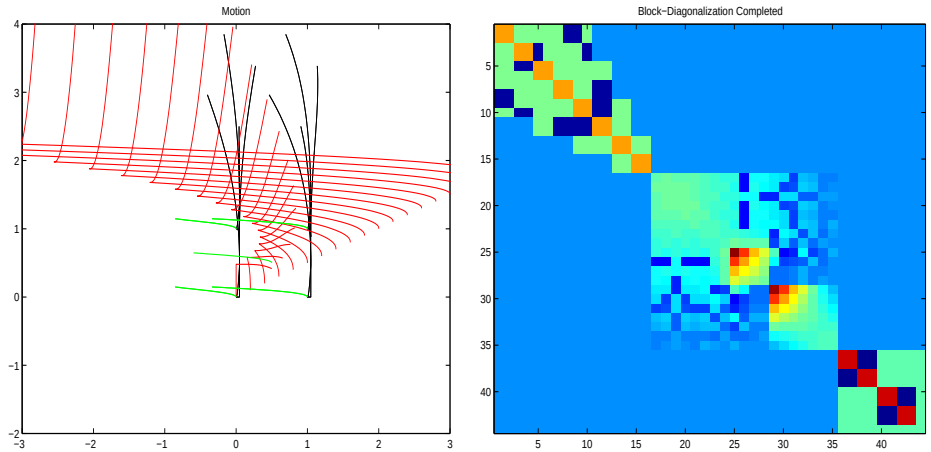
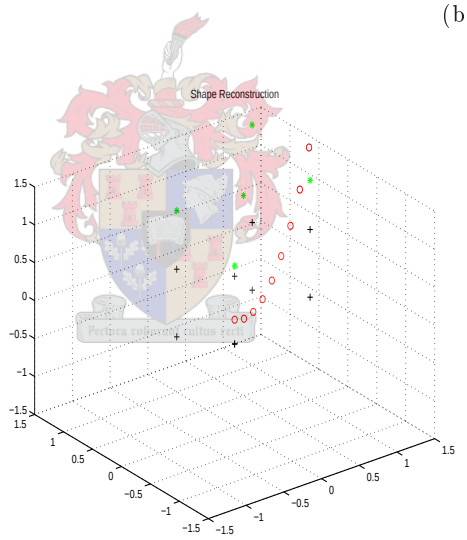


Figure 2.12: *SVD - experiment with synthetic (lab data); (a) Original image and (b)* 3D *reconstruction*

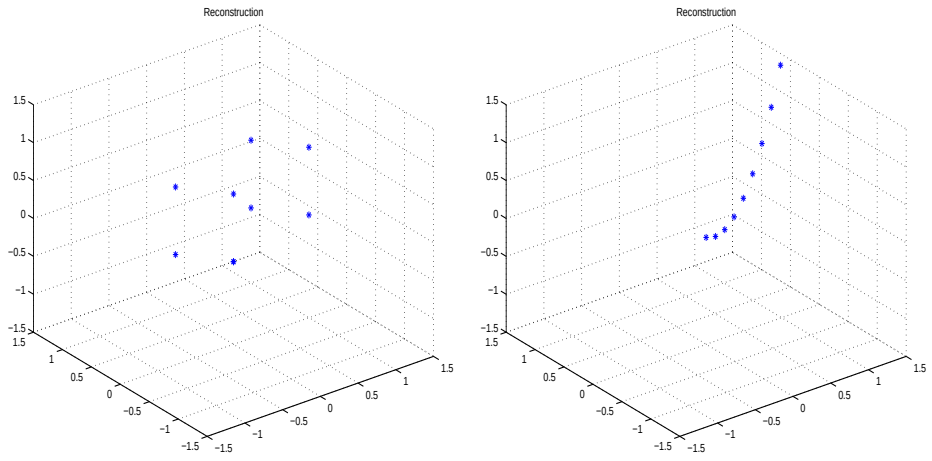


(a) (b)

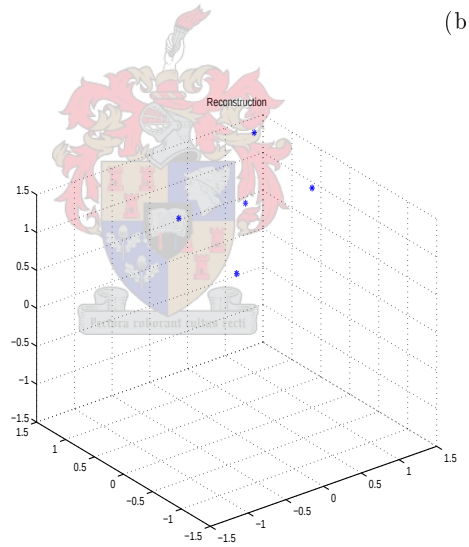


(c)

Figure 2.13: *(a) Observations, (b) Shape Interaction Matrix with three objects detected. Each block corresponds to one object and the block size determines the number of features and (c)* 3D *reconstruction*



(a) (b)



(c)

Figure 2.14: *SVD - Three objects moving independently: (a) Cube, (b) Parabola and (c) Open envelope*

Chapter 3

The Kalman Filter

3.1 Introduction

The Kalman filter (KF) is considered one of the greatest discoveries in the history of statistical estimation theory, Simon and Andrews [99], Maybeck [96]. In 1958, Kalman applied the notion of state variables to the Wiener filter problem, e.g. that of optimal estimation of noisy signal, and the problem of noise-free optimal regulator problem. This led to the discovery of a new filter known as the Kalman filter, Kalman [73]. The first applications of this filter appeared two years later at Ames Researcher Center where it was used to estimate and control the trajectory of the Apollo space craft.

The Kalman filter is a recursive filter which estimates the state of a dynamic system from a series of noisy measurements and dynamic models. Let us first define what we mean by filtering. Filtering is the process of estimating the internal state of a dynamic system, given a set of observations in the presence of noise. Kalman filters are optimal in a mean-square sense, therefore they can also be referred to as Minimum Mean-Square Error (*MMSE*) estimators.

The Kalman filter is computationally efficient, relying only on the current estimate and the new observations during computation. This makes the Kalman filter techniques suitable for real-time implementations in digital computers, as described in Sorenson [132]. In fact, by 1983, Kalman filters revolutionized several areas of applied sciences, such as engineering, navigation and trajectory determination by solving problems related to parameter and state estimation, Bar-Shalom et al [7], Haykin [60]. More discussion on Kalman filters can be found in Welch and Bishop [160], and discussions on tracking and filtering in Samuel and Popoli [122], Haykin [59], van der Merwe and Wan [147], van der Merwe et al [145] and Sourabh and Adelson [133].

In this chapter, we briefly discuss the Kalman filter and some of its extensions. The linear Kalman filter is discussed next.

3.2 The Linear Kalman Filter

The original Kalman filter assumes the linear relationship of previous and update measurement on a linear system with Gaussian, zero-mean noise. Thus the state is given by the following linear system,

$$\boldsymbol{x}_{t+1} = F_t \boldsymbol{x}_t + \boldsymbol{\mu}_t \tag{3.1}$$

where F_t is a matrix, $\boldsymbol{x}_t$ is the state at instant t, and $\boldsymbol{\mu}_t$ is the noise which is assumed to be the zero mean, white and Gaussian with covariance Q_t and statistically independent. The state variables are indirectly observed through the linear system,

$$\boldsymbol{y}_{t+1} = H_{t+1} \boldsymbol{x}_{t+1} + \boldsymbol{\nu}_{t+1} \tag{3.2}$$

where H_{t+1} is a matrix, $\boldsymbol{y}_{t+1}$ is the vector of observations, and $\boldsymbol{\nu}_{t+1}$ is a vector describing measurement noise. It has covariance G_{t+1}, and H_{t+1} is the observation transition model. This above assumption constitutes the linear Gaussian assumption. Equation (3.2) produces the current observation from the current state with some error. This error represents unpredictable deviations or inaccuracies in the measurement of the system.

We assume that the covariances Q_t and G_t are known. At time t, the conditional expectation of the state is written as

$$\hat{\boldsymbol{x}}(j|t) = E[\boldsymbol{x}_j | \boldsymbol{Y}^t] \tag{3.3}$$

where

$$\boldsymbol{Y}^t = \{\boldsymbol{y}_i, \quad i \leq t\}. \tag{3.4}$$

is the sequence of observations available at time t (Bar-Shalom et al [7]).

- If $j = t$ 3.3 preveils an estimate of the state,
- If $j < t$ 3.3 preveils a smoothed value of the state, and
- If $j > t$ 3.3 preveils the predicted value of the state.

The corresponding estimation error is defined as

$$\tilde{\boldsymbol{x}}(j|t) = \boldsymbol{x}_j - \hat{\boldsymbol{x}}(j|t). \tag{3.5}$$

The associated conditional covariance with estimate (3.3) is defined as

$$P(j|t) = E\Big[(\boldsymbol{x}_j - \hat{\boldsymbol{x}}(j|t))(\boldsymbol{x}_j - \hat{\boldsymbol{x}}(j|t))^T|\boldsymbol{Y}^t\Big]. \tag{3.6}$$

Similarly, the conditional measurement estimates are given by

$$\hat{\boldsymbol{y}}(j|t) = E[\boldsymbol{y}_j|\boldsymbol{Y}^t] \tag{3.7}$$

and the associated conditional covariance as

$$S(j|t) = E\Big[(\boldsymbol{y}_j - \hat{\boldsymbol{y}}(j|t))(\boldsymbol{y}_j - \hat{\boldsymbol{y}}(j|t))^T|\boldsymbol{Y}^t\Big]. \tag{3.8}$$

We define the mixed covariance as

$$T(j|t) = E\Big[(\boldsymbol{x}_j - \hat{\boldsymbol{x}}(j|t))(\boldsymbol{y}_j - \hat{\boldsymbol{y}}(j|t))^T|\boldsymbol{Y}^t\Big]. \tag{3.9}$$

Now the problem is to find iteratively the best possible state estimate $\hat{\boldsymbol{x}}(t+1|t)$ together with its covariance $P(t+1|t)$. In other words, we consider the case in which all available measurements at present time are used, that is $j = t$.

We define the state and measurement errors as following:

$$\tilde{\boldsymbol{x}}(t+1|t) = \boldsymbol{x}_{t+1} - \hat{\boldsymbol{x}}(t+1|t) \tag{3.10}$$

and

$$\tilde{\boldsymbol{y}}(t+1|t) = \boldsymbol{y}_{t+1} - \hat{\boldsymbol{y}}(t+1|t). \tag{3.11}$$

The difference between the predicted observation and the true observation is used to update an estimate of the true state. This state can be used to obtain a better estimate of the true state. This update is defined as the Kalman gain (K). The Kalman gain at time $t+1$ is given by, (see e.g. Bar-Shalom et al [7])

$$K(t+1) = E[\tilde{\boldsymbol{x}}(t+1|t)\tilde{\boldsymbol{y}}(t+1|t)^T|\boldsymbol{Y}^t]E[\tilde{\boldsymbol{y}}(t+1|t)\tilde{\boldsymbol{y}}(t+1|t)^T|\boldsymbol{Y}^t]^{-1}. \tag{3.12}$$

Using (3.8) and (3.9), the Kalman gain can be rewritten as

$$K(t+1) = T(t+1|t)S(t+1|t)^{-1} \tag{3.13}$$

and the state update as

$$\hat{\boldsymbol{x}}(t+1|t+1) = \hat{\boldsymbol{x}}(t+1|t) + K(t+1)(\hat{\boldsymbol{y}}_{t+1} - \boldsymbol{y}(t+1|t)). \tag{3.14}$$

The covariance update is

$$P(t+1|t+1) = P(t+1|t) - T(t+1|t)S(t+1|t)^{-1}T(t+1|t)^T. \tag{3.15}$$

Now we can describe the *Linear Kalman Filter* as following: Given $\hat{\boldsymbol{x}}(t|t)$ and its covariance $P(t|t)$ we calculate $\hat{\boldsymbol{x}}(t+1|t)$ using the dynamic system

$$\hat{\boldsymbol{x}}(t+1|t) = F_t\hat{\boldsymbol{x}}(t|t). \tag{3.16}$$

Next we predict the measurement $\hat{\boldsymbol{y}}(t+1)$ using

$$\tilde{\boldsymbol{y}}(t+1|t) = H_t\hat{\boldsymbol{x}}(t+1|t). \tag{3.17}$$

The difference between $\tilde{\boldsymbol{y}}(t+1|t)$ and $\boldsymbol{y}_{t+1}$ is used to update $\hat{\boldsymbol{x}}(t+1|t)$ to $\hat{\boldsymbol{x}}(t+1|t+1)$ using the Kalman gain. At every step we also update the covariance vectors to the next cycle.

An overview of the flow of the Linear Kalman Filter is shown in Table 3.1.

3.3 Kalman Filter Extensions

In this section we consider the problem of solving a nonlinear dynamic process. A general dynamic process of the system, *dynamic model*, is given by

$$\boldsymbol{x}_{t+1} = F(t, \boldsymbol{x}_t) + \boldsymbol{\mu}_t \tag{3.18}$$

where F is the state transition function and $\boldsymbol{\mu}_t$ the system noise. The related state variables are known through the state measurement, *observation model*, and they are given by

$$\boldsymbol{y}_{t+1} = H(t+1, \boldsymbol{x}_{t+1}) + \boldsymbol{\nu}_{t+1} \tag{3.19}$$

where H is the observation function and $\boldsymbol{\nu}_{t+1}$ is the measurement noise.

The problem can be stated as follows: given the measurements $\boldsymbol{y}_j$, $j = 1 \cdots t+1$ and the possible estimate $\hat{\boldsymbol{x}}_t$ of $\boldsymbol{x}_t$, what is the best approximation $\hat{\boldsymbol{x}}_{t+1}$ of $\boldsymbol{x}_{t+1}$?.

The solution of the above problem is mathematically solvable by employing some assumptions. We assume that both process and measurement noise have zero mean, and are white noise and Gaussian noise. That is

$$Q_t := E[\boldsymbol{\mu}_t \boldsymbol{\mu}_t^T] \tag{3.20}$$

We assume that the dynamic evolution of the system, measurements, Q_t and G_t are known and $\hat{\boldsymbol{x}}(1|0) := y_0$. Then given the estimate $\hat{\boldsymbol{x}}(t|t)$ and the correspondent covariance $P(t|t)$, we proceed as follows:

- We calculate predicted state at time $t+1$ prior to the measurement at $t+1$

$$\hat{\boldsymbol{x}}(t+1|t) = F_t\hat{\boldsymbol{x}}(t|t),$$

- We compute the covariance of estimate state at time $t+1$ prior to the measurement at time $t+1$

$$P(t+1|t) = F_tP(t|t)F_t^T + Q_t,$$

- We calculate predicted measurement at time $t+1$

$$\hat{\boldsymbol{y}}(t+1|t) = H_t\hat{\boldsymbol{x}}(t+1|t),$$

- We compute the covariance of the measurement at time $t+1$

$$S(t+1|t) = H_{t+1}P(t+|t)H_{t+1}^T + G_{t+1},$$

- We acquire new true observation

$$\boldsymbol{y}_{t+1},$$

and compare $\hat{\boldsymbol{y}}(t+1|t)$ with $\boldsymbol{y}_{t+1}$,

- We calculate the Kalman gain

$$K(t+1) = P(t+1|t)H_{t+1}^TS(t+1|t)^{-1},$$

- We estimate the optimal state

$$\hat{\boldsymbol{x}}(t+1|t+1) = \hat{\boldsymbol{x}}(t+1|t) + K(t+1)(\boldsymbol{y}_{t+1} - \hat{\boldsymbol{y}}(t+1|t)), \textit{and}$$

- We update the state error covariance

$$P(t+1|t+1) = P(t+1|t) - K(t+1)S(t+1|t)K(t+1)^T.$$

Table 3.1: *Summary of the Linear Kalman Filter.*

and

$$G_t := E[\boldsymbol{\nu}_t{\boldsymbol{\nu}_t}^T] \tag{3.21}$$

with $E[\boldsymbol{\mu}_{t_1}{\boldsymbol{\mu}_{t_2}}^T] = 0 = E[\boldsymbol{\nu}_{t_1}{\boldsymbol{\nu}_{t_2}}^T]$ for $t_2 \neq t_1$. The main idea is that we want the filter to propagate the conditional probability density of the desired quantities, given

the actual data coming from the measuring devices. For example, given a quantity $\boldsymbol{x}$ at time t and all measurements $\boldsymbol{y}_k$, $k = 1, \cdots, t$, then the function which describes the conditional density function is written as $p(\boldsymbol{x}|y_1, \cdots, y_t)$.

3.3.1 The Extended Kalman Filter

The Kalman filter described in Section 3.2 uses linear equations. Since real-world problems are seldom linear we need to extend it. In this section, we present a modified Kalman filter that solves nonlinear problems by using a linearisation approach.

The Extended Kalman Filter (*EKF*) is an adaptation of the Linear Kalman Filter Bar-Shalom et al, Robert and Patrick [118]. It is designed to filter random Gaussian noise from more general, nonlinear systems using simple approximations. Julier [69] observed that *EKF* has two drawbacks in practice

- The linearisation can produce highly unstable filters if the assumption of local linearity is violated.
- The derivation of the Jacobian matrices is nontrivial in most applications and often leads to significant implementation difficulties.

Consider a dynamic process described by the following unknown discrete nonlinear system:

$$\boldsymbol{x}_{t+1} = F_t(t, \boldsymbol{x}_t) + \boldsymbol{\mu}_t \tag{3.22}$$

$$\mathbf{y}_{t+1} = H_{t+1}(t+1, \boldsymbol{x}_{t+1}) + \boldsymbol{\nu}_{t+1}. \tag{3.23}$$

We assume that the noises $\boldsymbol{\mu}_t$ and $\boldsymbol{\nu}_t$ are addictive, zero mean and white noise and that they are normally distributed. That is, $\boldsymbol{\mu}_t \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, G_t)$ and $\boldsymbol{\nu}_t \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, Q_t)$. The noise may be uncorrelated with the initial estimate $\hat{\boldsymbol{x}}(0)$ and associated covariance $P(0)$. Recalling Section 3.2, we also assume we have an appropriate conditional mean given by

$$\hat{\boldsymbol{x}}(t|t) \approx E[\boldsymbol{x}_t|\boldsymbol{Y}^t] \tag{3.24}$$

and the associated covariance matrix $P(t|t)$. Because of nonlinearity we only have approximate conditional means. Therefore, the covariance matrix $P(t|t)$ is the mean square error matrix.

The *EKF* uses a Taylor series expansion to construct a linearized approximation of the nonlinear system. The idea behind *EKF* is to linearize the nonlinear problem and then to proceed as in a linear case, Bar-Shalom et al [7]. We then use the standard Linear Kalman Filter techniques to solve this linearized system. A Taylor

series expansion around $\hat{\boldsymbol{x}}_{t+1}$, yields

$$\hat{\boldsymbol{x}}_{t+1} = F(t, \hat{\boldsymbol{x}}(t|t)) + F_{\boldsymbol{x}}(t, \hat{\boldsymbol{x}}(t|t)(\boldsymbol{x}_t - \hat{\boldsymbol{x}}(t|t)) + HOT + \boldsymbol{\mu}_t, \tag{3.25}$$

where $F_{\boldsymbol{x}}$ is the Jacobian of F and HOT refers to *High Order Terms*. The predicted state can be found by taking the conditional expectation of (3.25) from time t to $t+1$. This yields

$$\hat{\boldsymbol{x}}(t+1|t) = F(t, \hat{\boldsymbol{x}}(t|t)). \tag{3.26}$$

The conditional expectation is given by, Bar-Shalom et al [7]

$$P(t+1|t) = F_{\boldsymbol{x}}(t, \hat{\boldsymbol{x}}(t|t))P(t|t)F_{\boldsymbol{x}}(t, \hat{\boldsymbol{x}}(t|t))^T + Q_t. \tag{3.27}$$

Similarly, the measurement update follows from

$$\hat{\boldsymbol{y}}(t+1|t) = H(t, \hat{\boldsymbol{x}}(t|t)) \tag{3.28}$$

and the measurement prediction error covariance is

$$S(t+1|t) = H_{\boldsymbol{x}}(t, \hat{\boldsymbol{x}}(t+1|t))P(t+1|t)H_{\boldsymbol{x}}(t, \hat{\boldsymbol{x}}(t+1|t))^T + G_{t+1}. \tag{3.29}$$

Similar to Section 3.2, given $\hat{x}(t|t)$ we calculate $\hat{x}(t+1|t)$ using the dynamic system (3.22). We predict the measurement $\boldsymbol{y}(t+1|t)$ using (3.28).

An overview of the flow of the Extended Kalman Filter is shown in Table 3.2. More information about the Extended Kalman Filter can be found in e.g. Bar-Shalom et al [7] and Haykin [60].

The Extended Kalman Filter diverges for many real problems, van der Merwe and Wan [146]. Another nonlinear extension of the Kalman filter, known as the *Unscented Kalman Filter*, has became a popular alternative and is discussed next.

3.3.2 The Unscented Kalman Filter

The *Unscented Kalman Filter* is a recent extension of the Kalman filter to nonlinear systems, Julier [69], Julier and Uhlmann ([70], [71]), Julier et al [72]), Wan and van der Merwe [154]. The *UKF* was first introduced by Julier and Uhlmann [70] to address the shortcomings of the *EKF*. 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 model.

In the *UKF*, the state is calculated from a Gaussian Random Variable (GRV).

- Define a more general dynamic nonlinear system

$$\boldsymbol{x}_{t+1} = F(t, \boldsymbol{x}_t) + \boldsymbol{\mu}_t, \tag{3.30}$$

$$\boldsymbol{y}_{t+1} = H(t+1, \boldsymbol{x}_{t+1}) + \boldsymbol{\nu}_{t+1}. \tag{3.31}$$

- Initialise

$$\boldsymbol{x}_0 := E[\boldsymbol{x}_0], \tag{3.32}$$

$$P(0) := E[P(0)]. \tag{3.33}$$

- Predict state

$$\boldsymbol{x}_{t+1} = F(t, \boldsymbol{x}_t) + F_{\boldsymbol{x}}(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t), \tag{3.34}$$

$$F_{t+1} = F_{\boldsymbol{x}}(t, \boldsymbol{x}_t)(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t), \tag{3.35}$$

$$P(t+1|t) = F_{t+1} P(t|t) F_{t+1}^T + Q_t. \tag{3.36}$$

- Predict observation

$$\hat{\boldsymbol{y}}_{t+1} = H(t+1, \boldsymbol{x}_{t+1}), \tag{3.37}$$

$$H_{t+1} = H_{\boldsymbol{x}}(t, \boldsymbol{x}_t), \tag{3.38}$$

$$S_{t+1} = H_{t+1} P(t+1|t) H_{t+1}^T + G_t. \tag{3.39}$$

- Get new true observation $\boldsymbol{y}_{t+1}$ and then compare it with $\hat{\boldsymbol{y}}_{t+1}$.
- Calculate Kalman gain

$$K_{t+1} = P(t+1|t) H_{t+1}^T S_{t+1}^{-1}. \tag{3.40}$$

- Update optimal state estimate and error covariance

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{x}_t + K_{t+1}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t), \tag{3.41}$$

$$\hat{P}(t+1|t) = (I - K_{t+1} H_{t+1})\, P(t|t). \tag{3.42}$$

Table 3.2: *Summary of the Extended Kalman Filter.*

The *UKF* captures the new mean and covariance close to the accuracy of a $3rd$ order Taylor series expansion, Wan and van der Merwe [154], promising greater accuracy and stability in nonlinear applications. In order to understand how the *UKF* works, the *Unscented Transform* is discussed first.

3.3.2.1 The Unscented Transform

The *Unscented Transform* (UT) is a method for calculating the statistics of a random variable which undergoes a nonlinear transform.

The UT, first presented by Julier and Uhlmann [70] and Julier et al [72], is a method for predicting means and covariances in nonlinear systems. It guarantees a second order accuracy in both mean and covariance without the need to calculate any Jacobian, Julier and Uhlmann [70].

Let $\boldsymbol{x}$ be an n-dimensional random variable with mean $\bar{\boldsymbol{x}}$ and covariance $\boldsymbol{P}_{xx}$. Consider another random variable undergoing a nonlinear transformation

$$\boldsymbol{y} = H(\boldsymbol{x}). \tag{3.43}$$

The goal is to compute the mean $\bar{\boldsymbol{y}}$ and covariance $\boldsymbol{P}_{yy}$ of $\boldsymbol{y}$. The UT specifies the random variable $\boldsymbol{x}$ in terms of a minimal set of sample points, which are propagated through the nonlinear transform and recombined to produce an approximation of the transformed random variable. This approach generally has second order accuracy. Consider the Taylor expansion of (3.43) about $\bar{\boldsymbol{x}}$ where $\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{\delta x}$, $\boldsymbol{\delta} x$ is a zero mean random variable with covariance $\boldsymbol{P}_{xx}$

$$H(\bar{\boldsymbol{x}}+\boldsymbol{\delta x}) = H(\bar{\boldsymbol{x}})+\nabla H\boldsymbol{\delta x}+\frac{1}{2}\nabla^2 H\boldsymbol{\delta x}^2+\frac{1}{6}\nabla^3 H\boldsymbol{\delta x}^3+\cdots,+\frac{1}{j!}\nabla^j H\boldsymbol{\delta x}^j+\cdots, \tag{3.44}$$

where $\frac{1}{j!}\nabla^j H\boldsymbol{\delta x}^j$ indicates jth order term in the multidimensional Taylor series. Taking expectation on the expansion (3.44), we write

$$\bar{\boldsymbol{y}} = \mathrm{E}[\boldsymbol{y}] = \mathrm{H}(\bar{\boldsymbol{x}}) + \frac{1}{2}\nabla^2\mathrm{H}\boldsymbol{P}_{\mathrm{xx}} + \frac{1}{6}\nabla^3\mathrm{HE}[\boldsymbol{\delta x}^3] + \cdots, \tag{3.45}$$

and the corresponding covariance

$$\begin{aligned}
\boldsymbol{P}_{yy} &= \mathrm{E}\big[(\boldsymbol{y}-\bar{\boldsymbol{y}})(\boldsymbol{y}-\bar{\boldsymbol{y}})^{\mathrm{T}}\big] && (3.46)\\
&= \nabla H\boldsymbol{P}_{xx}(\nabla H)^T + \frac{1}{4}\nabla^2 H\mathrm{E}[\boldsymbol{\delta x}^3](\nabla\mathrm{H})^{\mathrm{T}} + \frac{1}{2}\nabla\mathrm{HE}[\boldsymbol{\delta x}^3](\nabla^2\mathrm{H})^{\mathrm{T}}\\
&\quad +\frac{1}{2}\nabla^2 H\big(\mathrm{E}[\boldsymbol{\delta x}^4] - \mathrm{E}[\boldsymbol{\delta x}^2\boldsymbol{P}_{\mathrm{xx}}] - \mathrm{E}[\boldsymbol{P}_{\mathrm{xx}}\boldsymbol{\delta x}^2] + \boldsymbol{P}_{\mathrm{xx}}^2\big)(\nabla^2\mathrm{H})^{\mathrm{T}}\\
&\quad +\frac{1}{6}\nabla^3 H\mathrm{E}[\boldsymbol{\delta x}^4](\nabla\mathrm{H})^{\mathrm{T}} + \cdots. && (3.47)
\end{aligned}$$

The n-dimensional random variable $\boldsymbol{x}$ with mean $\bar{\boldsymbol{x}}$ is approximated by $n+1$ weighted points called sigma points, Julier et al [72]. It can be shown that the use of higher dimension in computing sigma points creates some difficulties for UT, see Table 3.3. Once the set of weighted points ($S = \{W, \boldsymbol{\mathcal{X}}\}$) has been derived, the prediction method is straightforward. W denotes the weights and $\boldsymbol{\mathcal{X}}$ is the set of sigma points. We apply a nonlinear function $\boldsymbol{\mathcal{Y}}_j = H(\boldsymbol{\mathcal{X}}_j)$ to instantiate each point

$\boldsymbol{\mathcal{Y}}_j$. Then the estimated mean and covariance of $\boldsymbol{y}$ are given by

$$\bar{\boldsymbol{y}} = \sum_{j=0}^{n} W_j \boldsymbol{\mathcal{Y}}_j, \tag{3.48}$$

and

$$P_{yy} = \sum_{j=0}^{n} W_j [\boldsymbol{\mathcal{Y}}_j - \bar{\boldsymbol{y}}][\boldsymbol{\mathcal{Y}}_j - \bar{\boldsymbol{y}}]^T, \tag{3.49}$$

where W_j are called weights and they have the following properties:

- the sum of all weights is always one

$$\sum_{j=0}^{n} W_j = 1, \tag{3.50}$$

- the value of the weights lies between zero and one

$$0 \leq W_j \leq 1. \tag{3.51}$$

In Table 3.3 we present the general algorithm for point selection using the *UT*.

A more recent version of the *UT* has been developed, which is called the *Scaled Unscented Transform* (*SUT*). This transform employs extra parameters to enable a scaled set of sample points to match the mean and covariance of a non-Gaussian distribution, while preserving second-order accuracy. Next, we briefly explain this in Section 3.3.2.2, but more details about *SUT* can be found in Julier [69].

3.3.2.2 Scaled Unscented Transform

The aim of the *SUT* is to find an approximation of the resultant random variable after it has undergone a possible nonlinear transformation. In [70], Julier and Uhlmann discuss the trade-off of *UT* when the number of sigma points increases. The first aspect is the increase of the radius of sphere which bounds the sigma points, since the distance from each point to the origin is given by the function of $2^{n-1/2}$. The second is related to the effects of higher order terms as the dimension increases. The best way to overcome both two drawbacks is to apply the sigma point scaling method. This method attempts to overcome dimensional scaling effects by calculating the transformation of scaled set of sigma points of the form

$$\boldsymbol{\mathcal{X}}'_j = \boldsymbol{\mathcal{X}}_0 + \alpha(\boldsymbol{\mathcal{X}}_j - \boldsymbol{\mathcal{X}}_0), \tag{3.57}$$

We present the general algorithm for computation of sigma points using the *Unscented Transform*.

- Initialize by choosing W_0 so that $0 \leq W_0 \leq 1$.
- Build a weight sequence using

$$W_i = \begin{cases} \frac{1-W_0}{2^n} & \text{for } i = 1, \\ W_1 & \text{for } i = 2, \\ 2^{i-1}W_1 & \text{for } i = 3, \cdots, n+1 \end{cases} \tag{3.52}$$

- Initialize vector sequence $\boldsymbol{\mathcal{X}}$ as

$$\boldsymbol{\mathcal{X}'}_{0,1} = [0], \tag{3.53}$$

$$\boldsymbol{\mathcal{X}'}_{1,1} = \left[-\frac{1}{\sqrt{2W_1}}\right], \tag{3.54}$$

and

$$\boldsymbol{\mathcal{X}'}_{2,1} = \left[\frac{1}{\sqrt{2W_1}}\right]. \tag{3.55}$$

- For $j = 2, \cdots, n$ expand above vector sequence following

$$\boldsymbol{\mathcal{X}'}_{i,j+1} = \begin{cases} \begin{bmatrix} \boldsymbol{\mathcal{X}'}_{0,j} \\ 0 \end{bmatrix} & \text{for } i = 1, \\ \begin{bmatrix} \boldsymbol{\mathcal{X}'}_{i,j} \\ -\frac{1}{\sqrt{2W_j}} \end{bmatrix} & \text{for } i = 1, \cdots, j, \\ \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2W_j}} \end{bmatrix} & \text{for } i = j+1 \end{cases} \tag{3.56}$$

Table 3.3: *Sigma Points Computation using* UT *Algorithm.*

where α is a positive number which minimizes the effects of higher order terms under the following restrictions:

- the predicted covariance must be positive semidefinite for all α and
- the second order accuracy in both the mean and covariance must be preserved.

The *SUT* computes the mean and covariance using (3.48) and (3.49). Assume that the scale factor α has been chosen and a set of sigma points S have been constructed with mean $\bar{\boldsymbol{x}}$ and covariance $\boldsymbol{P}_{xx}$. Note that the points in the new set

S' must obey the condition given in (3.57). Then the weights of the transformed sequence is given by

$$W_0^{(m)} = \frac{\lambda}{(n+\lambda)}, \tag{3.58}$$

$$W_0^{(c)} = \frac{\lambda}{(n+\lambda)} + (1-\alpha^2+\beta), \tag{3.59}$$

$$W_j^{(m)} = W_j^{(c)} = \frac{1}{2(n+\lambda)}, \quad j = 1, \cdots, 2n, \tag{3.60}$$

where $\lambda = \alpha^2(n+\kappa) - n$ and κ a real number used to reduce the overall prediction error, Julier and Uhlmann [70].

Given a set of points, procedure for the *Scaled Unscented Transform* can be summarized as:

- Select positive numbers α and β such that $0 \leq \alpha \leq 1$ and $\beta \geq 0$,
- Impose the scaling method given in (3.57),
- Apply the nonlinear transformation over $\boldsymbol{\mathcal{X}}'$

$$\boldsymbol{\mathcal{Y}}'_j = H[\boldsymbol{\mathcal{X}}'_j], \tag{3.61}$$

- Compute the weights using (3.58),
- Compute the mean as

$$\overline{\boldsymbol{y}} = \sum_{j=0}^{n} W^{(m)}{}_j \boldsymbol{\mathcal{Y}}'_j, \tag{3.62}$$

- Compute the covariance

$$\begin{aligned} \hat{\boldsymbol{P}}_{yy} &= \sum_{j=0}^{n} W^{(m)}{}_j [\boldsymbol{\mathcal{Y}}'_j - \overline{\boldsymbol{y}}][\boldsymbol{\mathcal{Y}}'_j - \overline{\boldsymbol{y}}]^T \\ &\quad + (1 + W_0^{(c)} + \beta - \alpha^2)[\boldsymbol{\mathcal{Y}}'_0 - \overline{\boldsymbol{y}}][\boldsymbol{\mathcal{Y}}'_0 - \overline{\boldsymbol{y}}]^T. \end{aligned} \tag{3.63}$$

  Note that β is a constant obtained when incorporating the higher order information from the Taylor series and in general $\beta = 2$ is optimal.

As described in Julier [69], any formulation of sigma point scaling should have the following two properties:

- the covariances should be positive semidefinite for all chosen α, and
- both mean and covariance should preserve the accuracy.

The next subsection describes how we implemented the sigma points calculation for a given random variable.

3.3.2.3 Sigma Points Computation

Given a mean $\bar{\boldsymbol{x}}$ and covariance matrix P_{xx} of the Gaussian random variable $\boldsymbol{x}$, with dimension n, the following matrix of sigma vectors $\boldsymbol{\chi}$ is generated:

$$\boldsymbol{\chi}_0 = \bar{\boldsymbol{x}}, \tag{3.64}$$

$$\boldsymbol{\chi}_i = \bar{\boldsymbol{x}} + \boldsymbol{\Phi}_i, \tag{3.65}$$

$$\boldsymbol{\chi}_{i+n} = \bar{\boldsymbol{x}} - \boldsymbol{\Phi}_{i+n}, \quad i = 1, \ldots, n \tag{3.66}$$

where $\boldsymbol{\Phi}_i$ is the $i'th$ column of Φ given by

$$P_{xx} = \frac{1}{n}\boldsymbol{\Phi}^T\boldsymbol{\Phi}. \tag{3.67}$$

The mapping from mean $\bar{\boldsymbol{x}}$ and covariance P_{xx} to its sigma points is given by

$$\boldsymbol{\chi} = \Upsilon(\bar{\boldsymbol{x}}, P_{xx}). \tag{3.68}$$

The following (Julier and Uhlmann [70]) use $\kappa = 0$, $\alpha = 1$ and $\beta = 2$. So the weights W^m and W^c are generated by

$$W_0^{(m)} = 0, \tag{3.69}$$

$$W_0^c = 2, \tag{3.70}$$

$$W_i^m = W_i^c = \frac{1}{2n}, \quad i = 1, \ldots, 2n. \tag{3.71}$$

3.3.2.4 Unscented Kalman Filter Algorithm

The algorithm relies on the weights given in (3.58) and the filter is initialized using (3.69) − (3.71). The weights are used to calculate the statistics of the state and observation. Tables 3.4 and 3.5 give the general *UKF* algorithm. The sigma points are formed by first calculating a matrix square root of the state covariance, Bar-Shalom et al [7]. The process noise and state sigma vectors are transformed by update equation into the *a priori* state estimate, which is combined with the measurement noise and processed by the measurement estimation to produce the estimated measurement sigma vectors. The *SUT* is applied to the measurement estimate sigma vectors, and the resulting mean (which corresponds to the measurement estimate) is subtracted from the actual measurement to produce the innovation (improvement update). The Kalman gain is also calculated, using the *a priori* state and measurement sigma points. The product of the Kalman gain and innovation is then added to

the *a priori* state estimate. The end result is the *a posteriori* state and state error covariance estimate.

3.4 Discussion

In this chapter three variants of the Kalman filter were discussed. The Kalman filter, in particularly the *UKF*, has proved to be suitable to solve real world problems. The *Extended Kalman Filter* is capable of modelling only the first and second moments of a Gaussian *PDF*, whereas the *Unscented Kalman Filter* is up to third moment modelling. 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 which is numerically stable and efficient, Julier and Uhlmann [70], Golub and van Loan [52]. The Choleski decomposition of an $n \times n$ matrix P must satisfy (3.67).

We present the general *UKF* implementation.

- The three Gaussian random variables are defined as:
  - State estimate $\hat{\boldsymbol{x}}(t|t)$ and covariance $P(t|t)$,
  - System noise with zero mean and covariance Q_t,
  - Measurement noise with zero mean and covariance G_t.
- Concatenate $\hat{\boldsymbol{x}}(t)$, P_t, Q_t and G_t into a super state variable $\hat{\boldsymbol{x}}^a$ and super state covariance matrix P^a.

  For each step, we do the following

$$\hat{\boldsymbol{a}}^a(t|t) = \begin{bmatrix} \hat{\boldsymbol{x}}(t|t) & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.72}$$

  and

$$P^a(t|t) = \begin{bmatrix} \boldsymbol{P}(t|t) & 0 & 0 \\ 0 & Q_t & 0 \\ 0 & 0 & G_t \end{bmatrix}. \tag{3.73}$$

- Calculate sigma points

$$\boldsymbol{\mathcal{X}}^a(t|t) = \Upsilon(\boldsymbol{x}^a(t|t), P^a(t|t)) \tag{3.74}$$

  or

$$\boldsymbol{\mathcal{X}}^a(t|t) = \begin{bmatrix} \boldsymbol{\mathcal{X}}^x(t|t) & \boldsymbol{\mathcal{X}}_t^q & \boldsymbol{\mathcal{X}}_t^g. \end{bmatrix}^T \tag{3.75}$$

- Apply nonlinear equation $F(.)$ to process the state estimate and process noise
  - sigma point propagation

$$\boldsymbol{\mathcal{X}}^x(t+1|t) = F(t, \boldsymbol{x}^x(t|t)) + \boldsymbol{\mathcal{X}}_t^q, \tag{3.76}$$

  - *a priori* estimate

$$\hat{\boldsymbol{x}}(t+1|t) = \sum_{j=0}^{2n} W_j^{(m)} \boldsymbol{\mathcal{X}}_j^x(t+1|t), \tag{3.77}$$

  - error covariance matrix

$$P(t+1|t) = \sum_{j=0}^{2n} w_j^{(c)} [\boldsymbol{\mathcal{X}}_j^x t+1|t - \hat{\boldsymbol{x}}(t+1|t)][\boldsymbol{\mathcal{X}}_j^x(t+1|t) - \hat{\boldsymbol{x}}(t+1|t)]^T. \tag{3.78}$$

Table 3.4: *Unscented Kalman Filter Algorithm - Part I.*

- Apply nonlinear propagation function $H(.)$ state estimate and measurement noise

$$\boldsymbol{\Upsilon}(t+1|t) = H(t+1, \boldsymbol{x}^x(t+1|t)) + \boldsymbol{\mathcal{X}}^g_{t+1}, \tag{3.79}$$

and *a priori* measurement

$$\hat{\boldsymbol{y}}(t+1|t) = \sum_{j=0}^{2n} W_j^{(m)} \boldsymbol{\Upsilon}(t+1|t). \tag{3.80}$$

- Calculate measurement error covariance matrix

$$S(t+1|t) = \sum_{j=0}^{2n} W_j^{(c)} [\boldsymbol{\Upsilon}(t+1|t) - \hat{\boldsymbol{y}}(t+1|t)][\boldsymbol{\Upsilon}(t+1|t) - \hat{\boldsymbol{y}}(t+1|t)]^T, \tag{3.81}$$

and the observation error covariance matrix

$$T(t+1|t) = \sum_{j=0}^{2n} W_j^{(c)} [\boldsymbol{\mathcal{X}}^x_j t+1|t - \hat{\boldsymbol{x}}(t+1|t)][\boldsymbol{\Upsilon}(t+1|t) - \hat{\boldsymbol{y}}(t+1|t)]^T. \tag{3.82}$$

- Compute the Kalman gain

$$K(t+1) = T(t+1|t) S^{-1}(t+1|t), \tag{3.83}$$

update the state estimate and estimated error covariance matrix

$$\hat{\boldsymbol{x}}(t+1|t+1) = \hat{\boldsymbol{x}}(t+1|t) + K(t+1)(\boldsymbol{y}_{t+1} - \boldsymbol{y}(t+1|t)), \tag{3.84}$$

$$P(t+1|t+1) = P(t+1|t) - K(t+1) S(t+1|t) K^T(t+1). \tag{3.85}$$

Table 3.5: *Unscented Kalman Filter Algorithm - Part II.*

Chapter 4

Optical Flow

4.1 Introduction

One of the difficulties encountered in solving *SfM* problems is the ability to track features over a long sequence, i.e. 100 or more frames.

This chapter discusses important aspects about optical flow and its application in feature tracking. Given a sequence of video images our goal is to recover the motion from *2D* observations by using tracking algorithms (see Figures 4.4 and 4.5). As mentioned in Chapter 2, if we can correctly recover the motion then it is possible to get the *3D* information. There are several feature tracking algorithms, such as the *Kanade-Lucas-Tomasi* algorithm, the *Harris Corner-Tracker*, *Optical Flow* and *Kalman filter*, Samuel and Popoli [122], Wan and Nelson [153], Nelson and Wan [105], Wan and van der Merwe [155], Bouthemy and Lalande [24], Yu and Dyer [166].

The main problem of tracking features using *KLT* is its inability to track every detected feature over a long sequence. Features are lost due to many factors such as occlusion and weather condition. On the other hand, our Kalman filter model does not make an assumption for lost features. One of the way to overcome this is to use robust algorithms for tracking such as robust optical flow.

The main idea in optical flow computation is the recovery of a piecewise smooth flow field in the presence of motion boundaries, i.e. the problem of identifying the motion discontinuities, Black and Anadan [18], Black [16]. As mentioned in Chapter 2, if we are able to recover the motion then we can also easily recover the object's shape.

There are two ways of computing optical flow:

- Dense optical flow

- Sparse optical flow

Dense optical flow is the computation of flow at every pixel of an image. Sparse optical flow is the estimation of optical flow at feature points by matching techniques. Sparse flow can be computed in two ways: either by computing dense optical flow and then throwing away regions of high uncertainty, or by doing sparse feature tracking.

Our goal is to develop an algorithm which should recover flow, provide motion boundaries and detect violated assumptions.

We explore the contribution of Black [17] and show that optical flow can be employed to robustly track features. In 1991, Black and Anandan [20] proposed a new algorithm that robustly estimates motion over time. In his thesis [17], Black formulated a new approach of robust and incremental optical flow, that provides useful information about motion and depth difference in a sequence of video images, however it is computationally expensive, Magaia et al [91].

Another way to compute optical flow is by using *Incremental Motion Estimation* exploiting the dynamic nature of motion. The goal of incremental motion estimation is to *incrementally integrate motion information from new images with previous optical flow estimates to obtain more accurate information about the motion in the scene over time*, Black [17], Beauchemin and Barron [11].

The incremental motion estimation approach has the following properties:

- Anytime Access: Motion estimates are always available;
- Temporal Refinement: Optical flow is updated over time while data is acquired;
- Computation Time Reduction: The main goal is to achieve real-time computation;
- Adaptative: As the motion of the observer and scene changes over time, an incremental algorithm must adapt to the changes in motion.

The basic ideas of incremental estimation are:

- The algorithm has a current estimate of the flow field at any time;
- Constraints must be applied and the estimate of the flow field refined for a new image;
- The temporal continuity assumptions are exploited to predict what the flow field will be at the next instant.

- This process is repeated as new images are acquired resulting in refined flow estimate (temporal refinement).

The approach exploits the robust gradient method and has the following goals:

- It should prevent over-smoothing;
- It should facilitate the recovering of motion discontinuities, and
- It must detect violations of model assumptions and reduce their effect on the solution.

4.2 Motion Field of Rigid Objects

The motion field is defined as the *2D* vector field of velocities of the image points induced by the relative velocity between the viewing camera and the observed scene. The goal of this section is to derive the equations which are used to compute the motion field of rigid objects. The system described in this section is a camera coordinate system. We are investigating the problem of a moving object and fixed camera. The relation between a point and its projection is discussed in detail in Chapter 5.

Given a *3D* point $\boldsymbol{P} = [x, y, z]^T$ then the relative velocity ($\boldsymbol{\gamma}$) between $\boldsymbol{P}$ and the camera is given by

$$\boldsymbol{\gamma} = \omega \boldsymbol{P} + \boldsymbol{\vartheta}, \tag{4.1}$$

where ω is the angular velocity matrix given by

$$\omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{4.2}$$

and $\boldsymbol{\vartheta}$ is the translational velocity vector written as,

$$\boldsymbol{\vartheta} = \begin{bmatrix} \vartheta_x \\ \vartheta_y \\ \vartheta_z \end{bmatrix}. \tag{4.3}$$

Let f be camera focal length, then the relation between the projection point $\boldsymbol{p} = [p_x\ p_y]^T$ in camera coordinate system and the $\boldsymbol{P'} = [x\ y]^T$ in object coordinate system is given by

$$\boldsymbol{p} = \frac{f}{z} \boldsymbol{P'}, \tag{4.4}$$

which can be written as

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = f \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix}. \tag{4.5}$$

We assume that there is only one, rigid, relative velocity between the camera and the observed scene, and the illumination condition changes slowly. Then the motion field can be thought of as the projection of the $3D$ velocity field on the image plane.

The result of differentiating both sides of (4.4) as function of time and substituting the velocity components defined in (4.1) and projection components, is the motion field $\boldsymbol{v} = [u\ v]^T$ given by

$$\begin{bmatrix} u \\ v \end{bmatrix} := \begin{bmatrix} \frac{\vartheta_z p_x - f\vartheta_x}{z} - f\omega_y + \omega_z p_y + \frac{\omega_x p_x p_y - \omega_y p_x^2}{f} \\ \frac{\vartheta_z p_y - f\vartheta_y}{z} + f\omega_x - \omega_z p_x - \frac{\omega_y p_x p_y + \omega_x p_y^2}{f} \end{bmatrix}. \tag{4.6}$$

Analyzing the right side of (4.6) we see four terms for each row. The terms $\frac{\vartheta_z p_x - f\vartheta_x}{z}$ and $\frac{\vartheta_z p_y - f\vartheta_y}{z}$ represent the translational components of the motion field while other are related to the rotational components. In fact, the motion field is the sum of two components, one of which depends on rotation only and the other from translation only.

Next, we briefly discuss an interesting motion field case namely, **pure translation**. The result of setting $\omega = 0$ in (4.6) is

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\vartheta_z p_x - f\vartheta_x}{z} \\ \frac{\vartheta_z p_y - f\vartheta_y}{z} \end{bmatrix}. \tag{4.7}$$

Considering the general case in which ($\vartheta_z \neq 0$) then, we can introduce a point $\boldsymbol{p}_0 = [p_{x0}\ p_{y0}]^T$

$$\begin{bmatrix} p_{x0} \\ p_{y0} \end{bmatrix} := \begin{bmatrix} f\frac{\vartheta_x}{\vartheta_z} \\ f\frac{\vartheta_y}{\vartheta_z} \end{bmatrix}. \tag{4.8}$$

Equation (4.7) becomes

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\vartheta_z}{z}(p_x - p_{x0}) \\ \frac{\vartheta_z}{z}(p_y - p_{y0}) \end{bmatrix} \tag{4.9}$$

which states that the motion field of a pure translation is radial radiating from the common origin ($\boldsymbol{p}_0$).

4.3 Optical Flow Segmentation

The idea behind optical segmentation is based on the following assumption: Models which describe visual motion assume that image regions corresponding to the same object are potentially correlated, Beymer et al [13] and Chau et al [30].

One of the purposes of optical flow segmentation is to identify discontinuities of the surface geometry which is due to either the presence of discontinuous contours on the object surface or to discontinuities in depth, Black and Anadan [21]. Since there is no unique interpretation of the results from segmentation, the result do contain the problem of identifying object boundaries, Black and Anadan [18]. There are several common assumptions used in calculating optical flow, and these are often violated in real-world sequences. We next describe some techniques of segmentation based on optical flow ideas. Optical flow is based on a number of assumptions.

The first is related to data conservation. In data conservation, we assume that the image regions undergo translation in the image plane, not taking into account the effects of deformation due to perspective or zooming. For two consecutive frames we can assume that regions are simply translated in the same way as another region in another frame. Data conservation constraints are violated in situations where there are sensor noise, specular reflections, shadows and illumination changes. Let $f(x, y, t)$ be an image intensity function at a point $p(x, y)$ at time t. We assume that the intensity of small regions in an image remains constant over time. Mathematically we can write this as

$$f(x, y, t) = f(x + dx, y + dy, t + dt) \tag{4.10}$$

or

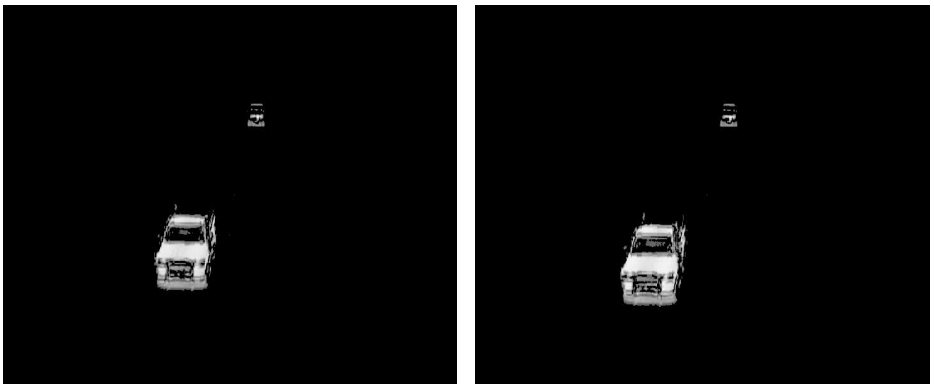
$$f(x, y, t) = f(x + udt, y + vdt, t + dt). \tag{4.11}$$

The second assumption is spatial coherence. The spatial assumptions are formulated by considering that neighbouring pixels in an image typically belong to the same surface and hence have similar velocities. This is true since pixels belonging to the same small region project to neighbouring pixels, we expect optical flow to change slowly and gradually.

The third assumption is temporal continuity in which we assume that a patch in the image has the same motion over time or it has constant velocity over time. Temporal continuity is only valid for short time intervals. It is violated if, for example, the motion of an object (in Figure 4.1) suddenly changes direction, or the camera

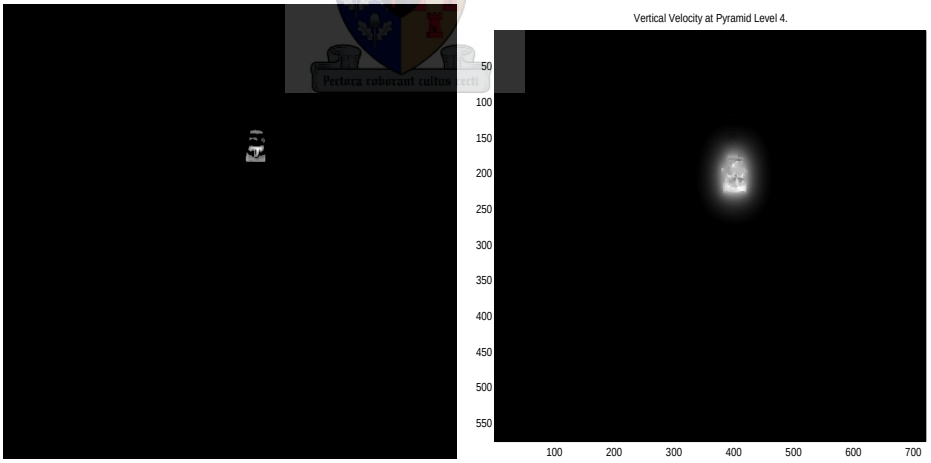
vibrates due to e.g. a truck passing through or high speed wind. Such changes are unpredictable.

Common violations are found on data conservation, spatial coherence, and temporal constraints. The origin of those constraints arises from our assumptions. *Data conservation* and *spatial coherence* are the two main constraints which are exploited by modern techniques to compute optical flow, Black [17]. In Figure 4.2b we plot



(a) (b)

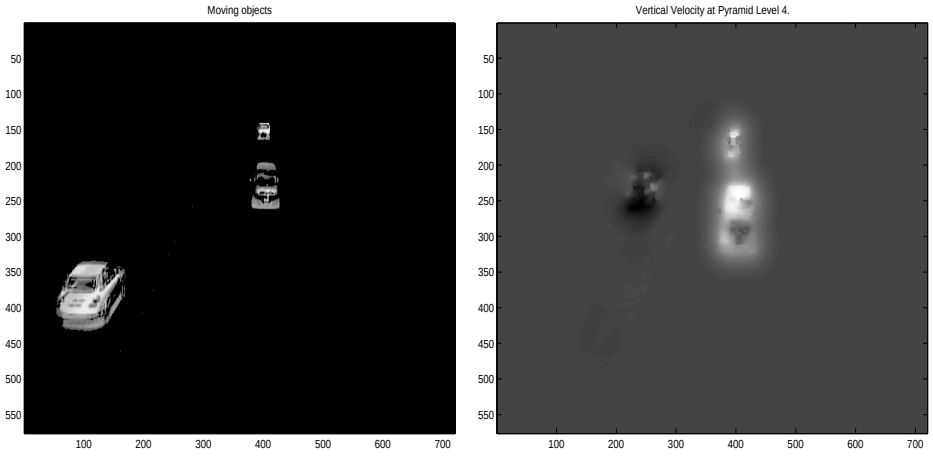
Figure 4.1: *Optical flow assumptions: (a) Frame* 40 *and (b) Frame* 42



(a) (b)

Figure 4.2: *(a) Original image and (b) Plotting of vertical velocity*

the vertical velocity between the observation. The car moves towards the camera. In Figure 4.3 we have three cars moving in different directions. The left side shows



(a) (b)

Figure 4.3: *Objects detection: (a) Original image and (b) Motion of three objects was detected*

three cars, two moving towards the camera and one moving forward. The black part represents the negative vertical velocity, thus the car is moving upwards (i.e. away from the camera) and the white positive vertical velocity (cars moving towards the camera).

The corners are among the features most easily to track from one frame to the next. Since motion boundaries correspond to physically significant structure in the scene, their computation is of great interest, Black and Anandan [19] and [21]. For example, it is clear that the corners of the object in Figure 4.4 correspond to a car. A general and flexible representation of visual motion can be used for many purposes but its effectiveness lies in robust and efficient computation, Tarr and Black [137].

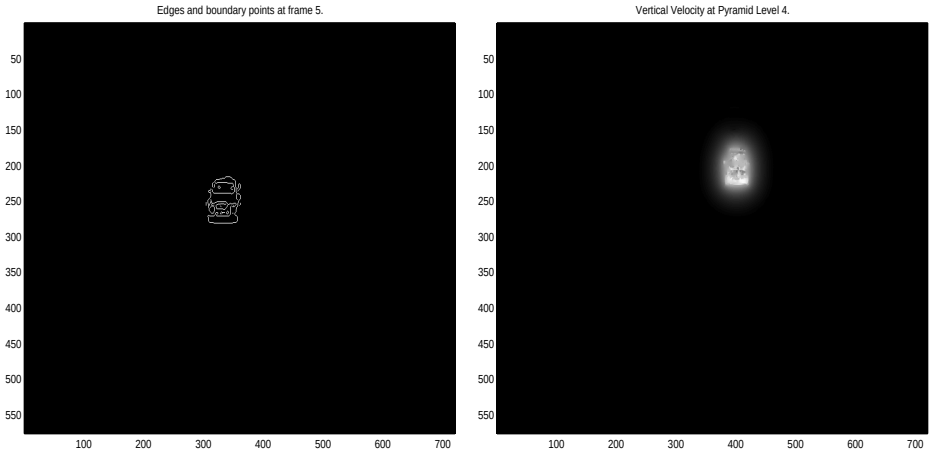
Next we derive the optical flow equations. We expand the RHS of (4.10) as a Taylor series

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + f_x dx + f_y dy + f_t dt + HOT \tag{4.12}$$

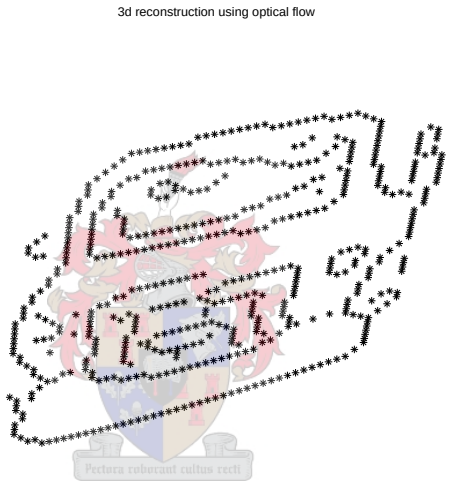
where f_x, f_y and f_t represent the partial derivatives of f with respect to x, y and t respectively. Using (4.10), we can rewrite (4.12) as

$$f(x+dx, y+dy, t+dt) = f(x,y,t) \tag{4.13}$$

If dx and dy are sufficiently small for a given dt, then the higher order in the Taylor series can be ignored.



(a) (b)



(c)

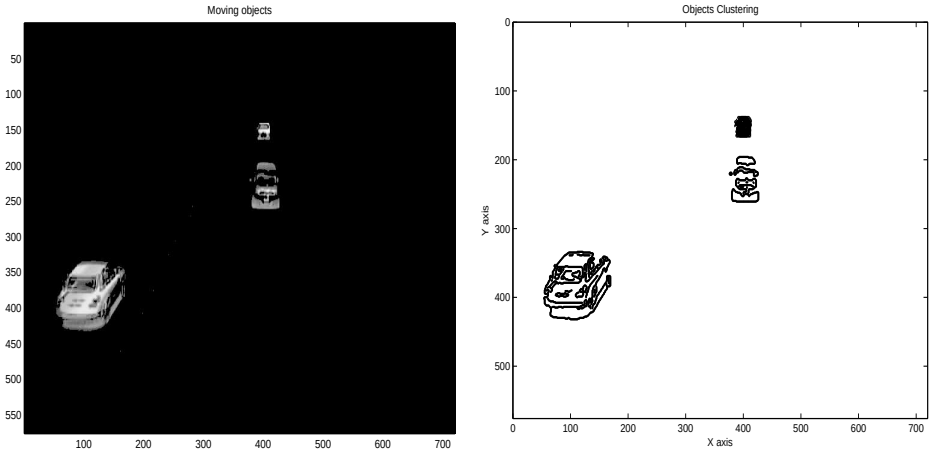
Figure 4.4: *Dense optical flow: (a) Features detection, (b) Motion detection and (c)* 3D *reconstruction*

In practice, the partial derivatives are approximated numerically by a forward difference

$$f_t \approx \frac{f(x, y, t + dt) - f(x, y, t)}{dt}. \tag{4.14}$$

Solving (4.10) for f_t we get

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt}. \tag{4.15}$$



(a) (b)

(c)

Figure 4.5: *Dense optical flow; (a) Original image, (b) Connected corners and (c)* 3D *reconstruction*

Now we can define the image velocity u and v as

$$(u, v) = \left(\frac{dx}{dt}, \frac{dy}{dt}\right). \tag{4.16}$$

Therefore, the *motion velocity* can be written as

$$f_x u + f_y v - f_t = 0. \tag{4.17}$$

The main goal in optical flow is to determine the velocity components (u, v) that

satisfy equation (4.17). Using equation (4.15) we define the overall error as

$$E_i = \sum_R W_i(x,y)(f_x u + f_y v - f_t)^2, \tag{4.18}$$

where R is a region (image) and $W(x,y)$ a *2D-Gaussian* weighting function (circularly symmetric). The overall error is just a minimization problem

$$E = min \sum_{i=1}^{m} E_{r(i)} + P_i \text{ over R}, \tag{4.19}$$

with penalties P_i, m is the number of windows. Because of the large size of the region the penalties are usually chosen close to zero.

E is a function of u and v, and has minimum where its gradient is zero. The resulting system of equations for each pixel is

$$\begin{bmatrix} -\sum W f_x f_t \\ -\sum W f_y f_t \end{bmatrix} = \begin{bmatrix} \sum W f_x^2 & \sum W f_x f_y \\ \sum W f_x f_y & \sum W f_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.20}$$

where $\sum = \sum_R$. The solution of this problem can be determined analytically. For each windows in R the optical flow is computed by (4.20). Define $a = \sum W f_x^2$, $b = \sum W f_x f_y$, $c = \sum W f_y^2$, $d = -\sum W f_x f_t$, $e = -\sum W f_y f_t$ and $f = \sum W f_t^2$, then the solution for each pixel is:

$$(u,v) = \left(\frac{cd - be}{ac - b^2}, \frac{ac - bd}{ac - b^2} \right). \tag{4.21}$$

The error valuation at each pixel is given by: $E_i = f + au^2 + cv^2 + 2(ud + ve + uvb)$. The valuation will end after $NM/2$ steps where N and M are the image size. The reader is advised to consult Xenophon and Peter [163] for more information including the dynamic programing algorithm.

In this section we illustrated the generality of problem posed by motion discontinuities. The next idea is to treat the violation of both constraints in a uniform manner. As in a statistical context, violations models are viewed as outliers, Black [17], Barnett and Lewis [9]. Now the problem can be seen as one of recovering optical flow in the presence of these outliers. Therefore, we use the field of robust statistics to solve the optical flow estimation problem. The robust optical flow is discussed next.

4.4 Robust Optical Flow Estimation

In the previous section we solved the optical flow problem using analytical methods which are not always suitable and stable for complex problems. This section focuses on how to improve the accuracy of the optical flow calculation by robustly estimating the motion. We apply the theory of robust statistics that Hampel et al [55] and Huber [65] have described as *the field developed to address the fact that the parametric models of classic statistics are often approximations of the phenomena being modeled.* The field of robust statistics addresses how to handle outliers, or gross errors which do not conform to the assumptions, Black [17]. The model violations previously discussed can be viewed as outliers, Huber [65]. Therefore, the problem of robustly estimating the motion is correlated to the goals of robust statistics, namely

- finding the structure that best fits the bulk of the data and
- identifying outliers or deviating data points.

Therefore, the problem of robust estimation is a minimization problem in which a set of values $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_m]$ is to be found that fits the model $u(s;\boldsymbol{\alpha})$ to a set of data measurements $\boldsymbol{d} = [d_0, d_1, \cdots, d_S]$ in cases where the data differs statistically from the model assumptions. We wish to find an $\boldsymbol{\alpha}$ that minimizes the residual errors $(d_s - u(s;\boldsymbol{\alpha}))$. So, we need to find

$$\min_{\boldsymbol{\alpha}} \sum_{s \in S} \rho(d_s - u(s;\boldsymbol{\alpha}), \delta_s), \tag{4.22}$$

where δ_s is a scale parameter and ρ is a suitable weight function. In our case, we choose ρ to be quadratic (the reader is advised to consult Black [17] for more details) defined as

$$\rho(x, \delta_s) = \frac{x^2}{2\delta_s^2}. \tag{4.23}$$

We can write the optimal estimator as

$$\rho(d_s - u(s;\boldsymbol{\alpha}), \delta_s) = \frac{(d_s - u(s;\boldsymbol{\alpha}))^2}{2\delta_s^2}. \tag{4.24}$$

Then the least squares method can be applied to the optimal estimator, equation (4.24). If the solution of equation (4.22) is relatively insensitive to small deviations of measurements, i.e. small changes for the bulk of data or large changes for a few points, then the estimator is said to be robust.

The *robust gradient method* is formulated by recasting the least squares minimization technique with the aim of (i) preventing smoothing across motion boundary and

(ii) permitting motion recovery, Black [17]. Therefore, (4.19) can be used as a general framework for assumption violation models (spatial coherence and temporal continuity).

Given an image R of size $n_1 \times n_2$, we define a grid of sites p_k, where $(i(p_k), j(p_k))$ defines the pixel location of the search window, and $0 \leq i(p_k), j(p_k) \leq n_1 n_2 - 1$ is satisfied. Trivially, $k \in [1, 2, \ldots, n_1 n_2]$. We define S as a small neighbour of points near (i, j); in our case

$$S = \{(i-1, j), (i+1, j), (i, j), (i, j-1), (i, j+1)\}. \tag{4.25}$$

We define G as the neighbourhood system that determines the local interaction of sites. G is defined as $G = \{G_s, s \in S\}$ $G = \{G_s, s \in S\}$ where for each $s \in S$, G_s satisfies the following properties, Black [17]:

- $G_s \subseteq S$,
- $s \notin G_s$, and
- $s \in G_t \Leftrightarrow t \in G_s$.

So, in our case G_s is that set of immediate neighbours of s and excludes s itself. We write G_s as

$$G_s = \{(i(s)+1, j(s)), (i(s), j(s)+1), (i(s)-1, j(s)), (i(s), j(s)-1)\}. \tag{4.26}$$

The least-squares form of the optical flow is given by

$$\begin{aligned} E(u,v) &= \sum_S \lambda E_D(u,v) + E_S(u,v) && (4.27) \\ &= \sum_{s \in S} [\lambda \rho(f_x u_s + f_y v_s + f_t, \delta_1) \\ &\quad + \sum_{n \in G_s} \rho(u_s - u_n, \delta_2) + \sum_{n \in G_s} \rho(v_s - v_n, \delta_2)], && (4.28) \end{aligned}$$

when ρ is a quadratic error measure, see (4.23)) and λ is a constant used to balance the two terms (We use $\lambda = 0.5$.), see Appendix A.7 for more information. E_D is the data conservation term and E_S is an explicit smoothness term using a different approach. Equation (4.27) is also known as the general equation of robust optical flow. The object function, $E(u, v)$, for the regularization approach is

$$E(u,v) = \sum_{p \in P} \left[\sum_R \lambda \rho(f_x u + f_y v + f_t, \delta_1) + \sum_{m \in G_p} \left[\rho(u_p - u_m, \delta_2) + \rho(v_p - v_m, \delta_2) \right] \right], \tag{4.29}$$

where G_p represents the set of neighbours of p in the grid, δ_1 and δ_2 are scale parameters, and ρ is the weight function defined in equation (4.23). Our goal is to solve for v_p and u_p. We take the partial derivative of equation (4.29) with respect to u_p, giving

$$\frac{\partial E}{\partial u_p} = \sum_{p \in P} \left[\lambda f_x \frac{\partial \rho}{\partial \delta_x}(f_x u + f_y v + f_t, \delta_1) + \sum_{m \in G_p} \frac{\partial \rho}{\partial \delta_x}(u_p - u_m, \delta_2) \right], \tag{4.30}$$

and set

$$\frac{\partial E}{\partial u_p} = 0. \tag{4.31}$$

Then we compute u_p using a standard iterative scheme. We choose the successive over-relaxation (SOR) method, and the $(k+1)th$ estimate of u_p is given by

$$u_p^{k+1} = u_p^k - \frac{w}{T(u_p)} \frac{\partial E}{\partial u_p}, \tag{4.32}$$

where w is the relaxation parameter and $T(u_p)$ is taken as an upper bound on the second partial derivative of E. Clearly, $T(u_p)$ can be defined as

$$T(u_p) = \frac{\lambda f_x^2}{\delta_1^2} + \frac{4}{\delta_2^2} \geq \frac{\partial^2 E}{\partial {u_p}^2}, \forall p \in P. \tag{4.33}$$

Similarly to solve for v_p, we solve $\frac{\partial E}{\partial v_p} = 0$ using the same technique. The reader is advised to consult Black [17] for more information.

Table 4.1 provides the successive over-relaxation algorithm. In order to make our calculation faster, for every two frame intervals we compute optical flow.

4.5 Discussion

We chose features of interest (corners) using *Harris Corner Detector* and we fed them to optical flow routine. The features are tracked using dense optical flow over a number of pre-defined image sequences. Optical flow is shown to be a better tracker compared to *KLT* due to its ability of tracking robustly. The tracked features are saved in a measurement matrix, say W in x and y order. Finally, the measurement matrix is used by the *Unscented Kalman Filter* to recover the *3D* structure.

Given is a time-varying sequence of n images and $f(x, y, t)$ be a function representing a dynamic change in the image brightness. f_x, f_y and f_t denote partial derivatives with respect to x, y and t, respectively.

- Initialize $(u, v) = (\frac{dx}{dt}, \frac{dy}{dt})$, set the number of iterations $itermax$ and the number of frames to process $nframes$, i.e $(v, u) = (0, 0)$, $itermax = 20$ and $nframes = 100$. $\bar{u}$ and $\bar{v}$ are the means of the velocity in the x and y directions in some neighbourhood of (x, y).

- $m = 1 \cdots nframes$.

  - Compute $P = \boldsymbol{f}_x \bar{u} + \boldsymbol{f}_y \bar{v}$ and $D = \lambda^2 + \boldsymbol{f}_x^2 + \boldsymbol{f}_y^2$. The constant λ is a Lagrange multiplier. See A.7 for more information.
  - Compute optical flow over all pixels in the image for $k = 1 \cdots itermax$ as

    $$u^{k,m} = \bar{u}^{k-1,m} - w\boldsymbol{f}_x \frac{P}{D} \tag{4.34}$$

    and

    $$v^{k,m} = \bar{v}^{k-1,m} - w\boldsymbol{f}_y \frac{P}{D}. \tag{4.35}$$

    The iteration can also stop if $E^2 \leq \xi$, where ξ is a permitted maximum error and

    $$E^2 = (\boldsymbol{f}_x u + \boldsymbol{f}_y v) + \boldsymbol{f}_t)^2 + \lambda(u_x^2 + v_x^2 + u_y^2 + v_y^2), \tag{4.36}$$

    where u_x, u_y, v_x and v_y are partial derivatives.

- Evaluate (u, v) over all image points.

- Increase m until all images in the sequence have been processed.

Table 4.1: *Relaxation Computation of Optical Flow*

Chapter 5

The Structure from Motion as a Kalman Filter Problem

The general problem of Structure from Motion (*SfM*) is the estimation of the *3D* structure and motion given a sequence of video images, Jebara et al [68], Margarey [94], Soato and Perona [129], [130], [131]. The motivation for using the Kalman filter approach lies in the fact that for more realistic camera model, the factorization method based on the simple orthographic can no longer be used fails. In a Kalman filter context, the shape and motion variables correspond to the hidden state. The image video sequence contains the observations.

In this chapter, we present a realistic model. We discuss the framework based on the concepts used by Azarbayejani [4] and Venter [150]. Next, some relevant techniques in the field of *SfM* are discussed, starting with techniques used in observation modeling and ending with structure and motion modeling to recover the *3D* information of the moving objects. Recall that in Chapter 2 we said that it was not possible to apply the factorization approach to real problems because it assumes a simple camera model. In addition it does not model the noise. In this chapter we investigate a realistic camera which models noise.

5.1 Observation Model

The pinhole camera model can be seen as a good approximation for the real-world lens systems found in cameras, see Figure 5.1. This model is characterized by the fact that only rays passing through a single point, the projection center, reach the image plane, and the image will be upside down as seen in the Figure 5.1. This model, a point in space with coordinates $P = [x, y, z]^T$ is mapped to the point on the image plane where a line joining the point P to the centre of projection meets the image plane. The focal plane passes through the focal point and is parallel to

the image plane. There are two coordinate systems, namely the object coordinate system (OCS) and the camera coordinate system (CCS). The relation of these two system is further discussed.

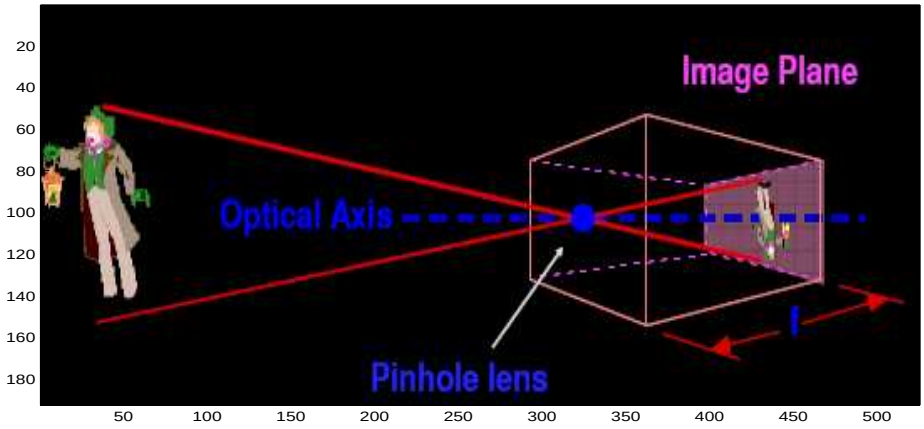


Figure 5.1: *The pinhole camera model as a real-world lens system*

In order to easier understand how this works we give a geometric interpretation of the pinhole camera model in Figure 5.2. Given a scene point in $3D-$space $\boldsymbol{P}$ with coordinates $[x,\ y,\ z]^T$ relative to Euclidean coordinate system is observed by a pinhole camera to point $\boldsymbol{P'}$ on the image plane with $\boldsymbol{O}$ as centre of the projection, see Figure 5.2.

Next a general matrix that describes this type of projection is derived. Consider the centre of the projection to be at point $\boldsymbol{C'}$ known as the *camera centre* or *optical centre*; see Figure 5.2.

Geometrically we represent this as seen in Figure 5.2. Using congruent triangles in Figure 5.2 it is easy to show that the image in the virtual image plane is exactly the same as the real image, only scaled and turned upside down, as can also be seen Figure 5.1. From the similarity we write the projection into the image plane (camera coordinates)

$$(x', y') \approx (f\frac{x}{z}, f\frac{y}{z}). \tag{5.1}$$

In homogeneous representation $[x, y, z, 1]^T$ is mapped to $[f\frac{x}{z}, f\frac{y}{z}, 1]^T$. Using the

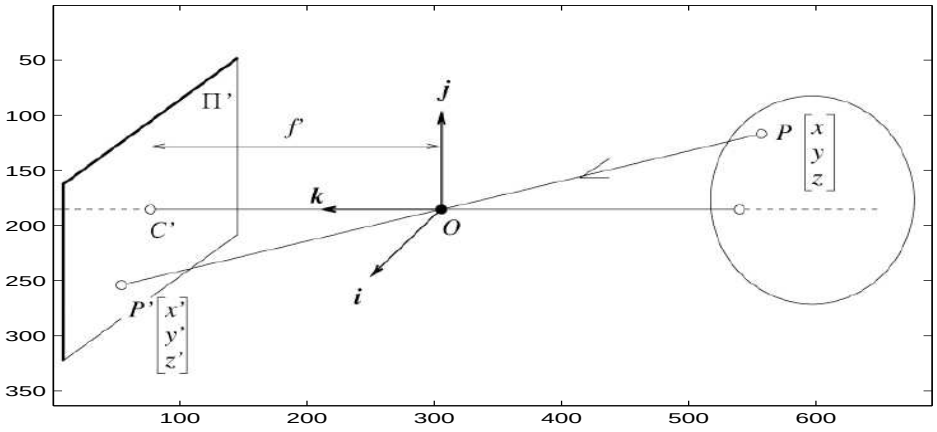


Figure 5.2: *Geometric meaning of pinhole camera projection model*

homogeneous representation we express this mapping as

$$\begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{5.2}$$

This derivation was simple for a homogeneous case but it is not used in practice. There are cases in which the origin of coordinates in the image plane is not at the principal point $(0, 0, f)$.

There is a general relation between object coordinate system and camera coordinate system.We can express a point in terms of a Euclidean coordinate system or in terms of camera coordinate system. These two systems are related to each other by translation and rotation. This is the case to investigate. Let R be a rotation matrix, $\boldsymbol{P}_{obj}$ is a $3D-$space point, $\boldsymbol{X}_{cam}$ the same point in the camera coordinate system and $\boldsymbol{C'}$ the camera centre, we write

$$\boldsymbol{X}_{cam} = R\boldsymbol{X}_{obj} + \boldsymbol{T} = R(\boldsymbol{X}_{obj} - \boldsymbol{C}_{cam}) \tag{5.3}$$

where $\boldsymbol{C}_{cam}$ represents the centre of the camera coordinate in object coordinate system, R is a 3 rotation matrix and $T = -R\boldsymbol{C'}$ translation vector. An orthographic projection is simply the pinhole camera with infinite focal length.

5.1.1 Scale Factor

This Subsection shows that the *3D* reconstruction can be estimated up to a scale value. In practice, the camera discretizes the image into pixels. The scales s_x and s_y translate image coordinates to pixel values where s_x and s_y are scales with respect to x and y. For simplicity, these are ignored for the rest of the discussion. Following Reinhard [116] one can write

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{z+f} \begin{bmatrix} p_x \\ p_y \end{bmatrix} \tag{5.4}$$

where $(p_x,\ p_y)$ is a point in the image coordinate system. By defining

$$\alpha = \frac{z}{f}, \tag{5.5}$$

(5.4) can be rewritten as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{1+\alpha} \begin{bmatrix} p_x \\ p_y \end{bmatrix}. \tag{5.6}$$

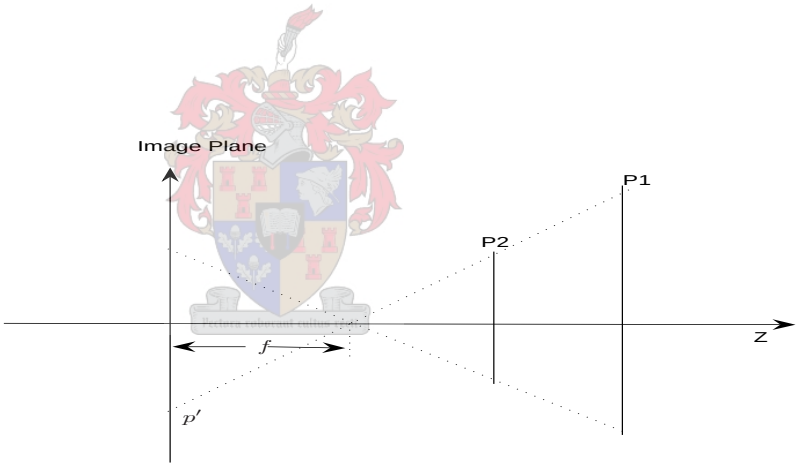


Figure 5.3: *The points p_1 and p_2 yield the same observation.*

The model described in Figure 5.3 tells us that only the ratio α is important. Changing the focal length f in this model will affect the projected coordinate independently of z, Azarbayejani and Pentland [5]. It is impossible to tell whether z has changed or the focal length has changed. Hence, different values yield the same ratio α.

From (5.5) one can see that if the focal length becomes infinite, then $\alpha \to 0$, and

so

$$\lim_{\alpha \to 0} \frac{1}{1+\alpha} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}. \tag{5.7}$$

And then one can write any *3D* point of a rigid object as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_x(1+\alpha) \\ p_y(1+\alpha) \\ z \end{bmatrix}, \tag{5.8}$$

This tells us, that if α and the measurements p_x and p_y are known, the *3D* coordinates x and y can be calculated but z remains undetermined, even in principle.

In fact, Figure 5.3 shows that two different features p_1 and p_2 have produced the same observation. The significance of this is that the structure can only be estimated up to a scale value. For rigid objects, the scale factor can in principle be determined for the whole structure and the object can be scaled accordingly.

The pinhole camera model has eleven degrees of freedom, namely:

- one for the focal length, f
- two for projections, $(x',\ y')$,
- three for rotation matrix R, $(r_x,\ r_y,\ r_z)$,
- three for translation vector $\boldsymbol{T}$, $(t_x,\ t_y,\ t_z)$ and
- two for scale factors, $(s_x,\ s_y)$.

5.2 Dynamic Model

In this chapter a dynamic model is presented where $\boldsymbol{p}$ is considered a *3D* point at a time τ and $\mathbf{z}$ its projection. Assuming the object's structure is specified in some coordinate system attached to the object, called the Object Coordinate System or *OCS*. Thus, the structure of the object is completely defined by a set of features in the Object Coordinate System, given as

$$\boldsymbol{p}^{ocs} = \begin{bmatrix} p_x \\ p_y \\ z \end{bmatrix}. \tag{5.9}$$

The dynamic process model is the transformation from the *OCS* to the camera coordinate system at each time step τ. Thus the motion is described by the set of

transformations $R(\tau)$ and $\boldsymbol{t}(\tau)$ so that, at time τ

$$\boldsymbol{p}^{ccs}(\tau) = R(\tau)\boldsymbol{p}^{ocs} + \boldsymbol{t}(\tau). \tag{5.10}$$

where R is the rotation matrix and $\boldsymbol{t}$ the translation component.

Because we are modeling a real world process, we assume the rate at which we are taking measurements is fast in comparison with changes in the motion. Therefore, the terms $R(\tau)$ and $\boldsymbol{t}(\tau)$ do not change quickly.

5.2.1 Model Initialization

The rotation is initialized by aligning the OCS with the corresponding CCS, that is $R0^{-1}(0) = I$. The translation components t_x and t_y are initialized as the mean of initial projections and t_z is initially equal to zero. That is

$$\mathbf{t}(0) = \begin{bmatrix} \overline{p}_{x,0} \\ \overline{p}_{y,0} \\ 0 \end{bmatrix}. \tag{5.11}$$

Then (5.8) can be rewritten as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_{x,0}(1+\frac{z}{f}) - \overline{p}_{x,0} \\ p_{y,0}(1+\frac{z}{f}) - \overline{p}_{y,0} \\ z \end{bmatrix} \tag{5.12}$$

$$= \begin{bmatrix} p_{x,0}(1+\alpha) - \overline{p}_{x,0} \\ p_{y,0}(1+\alpha) - \overline{p}_{y,0} \\ z \end{bmatrix}. \tag{5.13}$$

Note that any error introduced in the initial observation will affect the final results. We assume that the observations to be corrupted with Gaussian white noise and zero-mean. Thus

$$\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{G}_\tau), \tag{5.14}$$

where $\mathbf{0}$ denotes a zero-mean and $\boldsymbol{G}_\tau$ the observation covariance matrix at time τ.

Our main goal is to robustly recover the motion and structure from a *2D* images sequence. Therefore, structure and motion models are discussed next.

5.2.2 Structure Model

The flavour of the models is described as following:

The structure model is represented by z and the model uncertainties are zero-mean with covariance $\boldsymbol{Q}$. The rotation sub-model is represented by $\boldsymbol{x}$ and it is concatenated by $\boldsymbol{q}$ (rotational quaternion) and $\boldsymbol{w}$ (rotational velocity). The rotation sub-model describes the rotation of the object coordinate system as well as the camera coordinate system. Finally, the translation sub-model is represented by $\boldsymbol{t}$ describing the translation of the object in camera coordinate system over time.

We describe the structure model assuming that the object's features are moving slowly with respect to the frame rate and obey the requirements of the Nyquist theorem, Corporation [39]. This implies that the current state of the structure is similar to the previous one, that is

$$\mathbf{s}_{i|i-1} = \mathbf{s}_{i-1|i-1} + \boldsymbol{\mu}_{i-1}. \tag{5.15}$$

Equation (5.15) is referred as the structure state transition. The modeling uncertainties ($\boldsymbol{\mu}$) will be expressed as

$$\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_\tau), \tag{5.16}$$

where $\mathbf{0}$ denotes a zero-mean and $\boldsymbol{Q}_\tau$ the state covariance matrix. Let l be the number of features of an object. The estimate model of the structure of each feature (state transitions), form the structure vector

$$\boldsymbol{s}_i = [\, s_{i0}, \cdots, s_{ik}, \cdots, s_{in-1} \,]^T, \quad i \in [1, \cdots l]. \tag{5.17}$$

We emphasize that from the projection the missing information is about z. The structure model is initialized by using the initial information about a flat object.

5.2.3 Motion Model

5.2.3.1 The Rotation Sub-model

The estimation of three-dimensional object orientation or rotation and position parameters from image data is an important aspect in *SfM* problems, Mukundan [100]. Our rotation model is represented mathematically by a normalized quaternion (see Section A.4). Quaternions are hyper-complex numbers of rank 4, constituting a four dimensional vector space over the field of real numbers, Faugeras [48], Vicci [151]. The rotation sub-model describes the rotation of the OCS as well as its rotational velocity in relation to its initial orientation in the CCS. The following four-tuple

notation is used to represent a quaternion

$$\mathbf{q} = [\, q_0,\ q_1,\ q_2,\ q_3\,]^T, \tag{5.18}$$

where $\mathbf{q}^T\mathbf{q} = 1$. The most important property of quaternions is that they can characterize rotations in three-dimensional space, using a set of *Euler angles* $\{\psi, \phi, \theta\}$, Mukundan [100]. Following Mukundan [100], the rotation velocity vector ω is concatenated in three quaternion rotations as

$$\omega = [\omega_x\ \omega_y\ \omega_z] \tag{5.19}$$

and ω_x is the velocity around $x - axis$, etc.

There are some advantages in representing rotational parameters using quaternions: (i) the algebraic system of equations has closed-form solutions for quaternion elements; (ii) the estimation of *3D* rotational parameters leads to a nonlinear system of algebraic equations that can be solved using well-known numerical techniques, Mukundan [100], Faugeras [48]. Our model assumes that acceleration changes slowly; so, it is clear that both the quaternion $\mathbf{q}$ and the rotational velocity $\boldsymbol{\omega}$ should form part of the estimation. Therefore, the state variable for rotation is composed as

$$\mathbf{x}_R = [q_0\ q_1\ q_2\ q_3\ \omega_x\ \omega_y\ \omega_z]. \tag{5.20}$$

Using Appendix A.4, the general update equation is given by

$$\mathbf{x}_{R+1} = \mathbf{x}_R + [\hat{s}\ \hat{v_1}\ \hat{v_2}\ \hat{v_3}\ \hat{w}_x\ \hat{w}_y\ \hat{w}_z]\delta\tau + \phi, \tag{5.21}$$

where ϕ is some noise. In the state variable we use the rotation update given by (5.40) and the estimated rotation matrix R is calculated from (A.19). The state-dependence of the model parameters must be incorporated in order to implement it in the Kalman filter, Choukroun et al [31].

5.2.3.2 The Translation Sub-model

In this section we discuss translation ignoring rotation. The goal of the translation sub-model is to describe the translation of the object in the CCS over time. Consider the motion described by τ:

$$\mathbf{p}^{ccs}(\tau) \quad = \quad \Big[\boldsymbol{p}^{ocs} + \boldsymbol{t}(\tau)\Big] = \begin{bmatrix} p_x \\ p_y \\ z \end{bmatrix} + \begin{bmatrix} t_x(\tau) \\ t_y(\tau) \\ t_z(\tau) \end{bmatrix}. \tag{5.22}$$

We wish to rewrite the above equation so that one of the translation parameters will have the most effect on the image plane projections. To do this we define

$$\varphi_x(\tau) = t_x(\tau)/(1 + t_z(\tau)/f), \tag{5.23}$$

$$\varphi_y(\tau) = t_y(\tau)/(1 + t_z(\tau)/f) \tag{5.24}$$

and

$$\varphi_z(\tau) = t_z(\tau)/f. \tag{5.25}$$

From (5.4), the projection is defined as

$$\boldsymbol{z}(\tau) = \frac{f}{(f + z + t_z(\tau))} \begin{bmatrix} p_x + t_x(\tau) \\ p_y + t_y(\tau) \end{bmatrix}, \tag{5.26}$$

and this can be rewritten in terms of transformed variables, from (5.23) to (5.25) as

$$\boldsymbol{z}(\tau) = \frac{f}{(f + z + f\varphi_z)} \begin{bmatrix} p_x + \varphi_x(1 + \varphi_z) \\ p_y + \varphi_y(1 + \varphi_z) \end{bmatrix}. \tag{5.27}$$

The time-propagation of these three variables is now derived in relation to the object's CCS velocity: The time-derivative $\mathbf{d}$ of $\boldsymbol{\varphi}$ is defined as

$$\mathbf{d} = [\varphi'_x \ \varphi'_y \ \varphi'_z]^T. \tag{5.28}$$

Consider the time-derivative of φ_z:

$$\frac{d}{d\tau}\varphi_z = t'_z/f, \tag{5.29}$$

which can be rewritten as a forward difference estimate (see Dahrlquist and Bjorck [42], Burden and Faires [27]).

$$\varphi'_z(\tau) = \frac{\varphi_z(\tau + \Delta\tau) - \varphi_z(\tau)}{\Delta\tau}. \tag{5.30}$$

In the case of $\varphi_x(\tau)$ and $\varphi_y(\tau)$, the computation is more complicated due to the scaling factor φ_z,

$$\frac{d}{d\tau}\varphi_x(\tau) = \frac{t'_x(1 + \frac{t_z}{f}) - \frac{t'_z t_x}{f}}{(1 + \frac{t_z}{f})^2} \tag{5.31}$$

$$= \frac{\varphi'_x}{1 + \varphi_z} - \frac{\varphi'_z t_x}{(1 + \varphi_z)^2}. \tag{5.32}$$

Rewriting this equation as a forward difference and substituting (5.23) produces

$$\varphi_z = \frac{\varphi_z(\tau + \Delta\tau) - \varphi_z(\tau)}{\Delta\tau}, \tag{5.33}$$

which simplifies to

$$\varphi_x(\tau+1) \;=\; \varphi_x(\tau) + \Delta\tau \frac{t_x(\tau+1) - t_x(\tau)}{1+\varphi_z} - \Delta\tau t_x \frac{\varphi_z(\tau+1) - \varphi_z(\tau)}{(1+\varphi_z)^2}. \tag{5.34}$$

Similarly, for $\varphi_y(\tau+1)$,

$$\varphi_y(\tau+1) \;=\; \varphi_y(\tau) + \Delta\tau \frac{t_y(\tau+1) - t_y(\tau)}{1+\varphi_z} - \Delta\tau t_y \frac{\varphi_z(\tau+1) - \varphi_z(\tau)}{(1+\varphi_z)^2}. \tag{5.35}$$

The $\Delta\tau$ value used in the equations above depends on the chosen time scale. When the frame-rate differs from frame to frame over the sequence, the value of $\Delta\tau$ should change in proportion to the frame-rate. The definition of the translation state variable's composition is given

$$\boldsymbol{\varphi} = [\varphi_x \;\; \varphi_y \;\; \varphi_z]^T, \tag{5.36}$$

which uses (5.34), (5.35) and (5.30) as the various time propagation functions from (5.23) to (5.25).

5.2.3.3 Compact Model

The main goal is to build a compact model. The object's rotation relative to the *OCS* is described using quaternions (see Appendix A for more information).

The rotation and angular velocity state transition are modeled as in equations (5.20) and (5.21).

The translation relates the origin of the *OCS* relative to the *CCS*. The translation is given by equation (5.36) and the translation velocity states that the velocity is the time-derivative of the translation

$$\mathbf{d} = [\, d_x, \; d_y, \; d_z \,]^T. \tag{5.37}$$

The respective state column vectors is concatenated to construct the full state column vector with dimension $n + 13$:

$$\mathbf{S} = [\, \mathbf{s}^T, \; \mathbf{q}^T, \; \boldsymbol{\omega}^T, \; \varphi^T, \; \mathbf{d}^T \,]^T. \tag{5.38}$$

5.2.4 Computing the *a priori* Structure and Motion Variables

The Kalman filter calculates the *a priori* state variables during the process update step, which represent the "guess" of the state of the system during the next time-step. In this section the equations for producing these values for each model are discussed. The structure estimation is Gaussian random with zero-mean white noise with a variance $\boldsymbol{P}_s$, which represents structure model noise. The rotational velocity is assumed to change slowly over time it is propagated by

$$\boldsymbol{\omega}(\tau+1) = \boldsymbol{\omega}(\tau) + \hat{\boldsymbol{\omega}}\delta\tau + \eta(\mathbf{0}, \boldsymbol{P}_{\boldsymbol{\omega}}). \tag{5.39}$$

Again $\eta(\mathbf{0}, \boldsymbol{P}_{\boldsymbol{\omega}})$ is the zero-mean white noise which incorporates the modeling approximation error. The rotation estimation given by Appendix A.19 is updated by

$$\begin{bmatrix} \hat{s} \\ \hat{v}_1 \\ \hat{v}_2 \\ \hat{v}_3 \end{bmatrix}_{\tau+1} = \begin{bmatrix} 1 & \frac{\delta\tau}{2}\hat{w}_z & -\frac{\delta\tau}{2}\hat{w}_y & \frac{\delta\tau}{2}\hat{w}_x \\ \frac{\delta\tau}{2}\hat{w}_z & 1 & \frac{\delta\tau}{2}\hat{w}_y & \frac{\delta\tau}{2}\hat{w}_y \\ \frac{\delta\tau}{2}\hat{w}_y & -\frac{\delta\tau}{2}\hat{w}_x & 1 & \frac{\delta\tau}{2}\hat{w}_z \\ \frac{\delta\tau}{2}\hat{w}_x & -\frac{\delta\tau}{2}\hat{w}_y & -\frac{\delta\tau}{2}\hat{w}_z & 1 \end{bmatrix} \begin{bmatrix} \hat{s} \\ \hat{v}_1 \\ \hat{v}_2 \\ \hat{v}_3 \end{bmatrix}_{\tau} + \eta(\mathbf{0}, \boldsymbol{R}). \tag{5.40}$$

The translational velocity estimate is also assumed to change slowly over time, so that

$$\mathbf{d}(\tau+1) = \mathbf{d}(\tau) + \eta(\mathbf{0}, \boldsymbol{P}_{\mathbf{d}}), \tag{5.41}$$

where $\eta(\mathbf{0}, \boldsymbol{P}_{\mathbf{d}})$ represents the estimation error as well as the error in the translation model. This in turn allows us to write the *a priori* scaling factor φ_z as

$$\varphi_z(\tau+1) = \varphi_z(\tau) + \Delta\tau\varphi_z'(\tau+1) + \eta(\mathbf{0}, \boldsymbol{Q}_{\mathbf{t}_z}), \tag{5.42}$$

and the image plane translation estimate as

$$\begin{bmatrix} \varphi_x(\tau+1) \\ \varphi_y(\tau+1) \end{bmatrix} = \begin{bmatrix} \frac{1}{1+\frac{2\Delta\tau d_z(\tau+1)}{f\varphi_z(\tau+1)}} \left[\varphi_x(\tau) + \Delta\tau\frac{d_x(\tau+1)}{\varphi_z(\tau+1)}\right] \\ \frac{1}{1+\frac{2\Delta\tau d_z(\tau+1)}{f\varphi_z(\tau+1)}} \left[\varphi_y(\tau) + \Delta\tau\frac{d_y(\tau+1)}{\varphi_z(\tau+1)}\right] \end{bmatrix} + \eta(\mathbf{0}, \boldsymbol{P}_{\mathbf{t}_{xy}}), \tag{5.43}$$

where $\eta(\mathbf{0}, \boldsymbol{Q}_{t_z})$ and $\eta(\mathbf{0}, \boldsymbol{P}_{t_{xy}})$ represent the estimation error in the object's distance from the camera and its projected center respectively.

5.2.5 The Measurement Estimation Model

The measurement estimate model applies the perspective projection to the current estimate of the *3D* CCS object. This produces the *2D* image plane coordinates of the current object points. Since one can measure the actual current *2D* image plane coordinates of the object, it is possible to compare these two sets of information to produce the Kalman filter's innovation value, see Section 3.2 or Bar-Shalom et al [7]

for more information.

5.2.6 Initialization

The convergence of any model depends on suitable initial conditions. If the initial information is far removed from the solution, either the model converges slowly, or it diverges. When nothing is initially known about the structure of the object, $\boldsymbol{s}_0 = (z_x, z_y, 0)$ is a good choice for each point, since the initial estimated object appears to be a flat plane parallel to the image plane.

Since we do not have any knowledge of rotation, a relative measure of orientation is to align the *OCS* with the *CCS*. We set the initial quaternion as

$$\mathbf{q}_0 = [\,1,\ 0,\ 0,\ 0\,] \tag{5.44}$$

(i.e. the initial rotation is the identity matrix). Accordingly, the initial angular velocity vector is set to $\boldsymbol{\omega}_0 = \mathbf{0}$.

We only have partial information available to initialize the translation vector. The components t_x and t_y are initialized 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

$$\mathbf{t}_0 = \begin{bmatrix} \overline{p}_{x,0} \\ \overline{p}_{y,0} \\ 0 \end{bmatrix}, \tag{5.45}$$

where t_z is set to zero. Thus, the 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. We set the translation velocity to $\mathbf{d}_0 = \mathbf{0}$.

In general it is not easy to estimate the initial error covariance since little *a priori* information is available. The state covariance matrix P is initialized with a spherical covariance matrix, containing the same value for each entry on its main diagonal. The matrices Q and G are also spherical matrices and the entries on the main diagonal are fixed for a particular sequence. Thus, these matrices are each fully described by a variance. There are two factors that influence the observation uncertainty:

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

However it is possible to estimate the observation noise from the resolution of the frames for a particular sequence. Observation noise is never less than the resolution. All images are scaled by the resolution height and width to create a normalized frame with unity width and height. Thus, if one pixel deviation is assumed, then the observation noise variance can be based on the mean of the respective reciprocals of the resolution width and height

$$\sigma_r^2 = \frac{1}{2}\left(\frac{1}{width} + \frac{1}{height}\right). \tag{5.46}$$

The maximum value of 0.0016 (mean error) will be used as the global measurement noise value error,

$$\mathbf{E}_{576\times720} \quad = \tfrac{1}{2} \quad \left[\tfrac{1}{576} + \tfrac{1}{720}\right] = 0.001562500. \tag{5.47}$$

The system proved to be efficient and accurate on tests with both synthetic and real data. We found that *UKF* will need more frames to converge if the data contains some noise over 15 percent of the absolute value of the data. Experimentally we also found that when setting the three covariances as

- $P = I \times 5 \times 10^{-2}$,
- $Q = I \times 10^{-3}$ and
- $G = I \times 10^{-3}$

the system converged after 80 frames where I is identity matrix.

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

$$N = 2 \times (n + 13) + 2 \times n \tag{5.48}$$

$$= 4 \times n + 26. \tag{5.49}$$

5.3 Implementation

Our algorithm implements two estimations as two separate *UKF* Filters. In the Kalman process update step, the estimates of the initial structure (in the OCS) and current motion (in the CCS) are calculated. In the Measurement Estimation step, the motion is applied to the structure. This aligns the estimated structure with the estimated orientation of the object. The image plane projection of each point is now calculated. During the Kalman innovation, these estimated measurements are compared with the actual measurements.

5.3.1 The Unscented Kalman Filter Loop

The Kalman filter is applied to the *SfM* problem as the hidden state a single state variable. In *SfM*, the state to be estimated consists of two distinct sub states:

- The motion information, which may vary dynamically and
- the structural information, which is static (for a rigid object).

It is possible to separate the state into a *motion* and the *structure* components. This allows the use of a dual-kalman filter estimating the two components in conjunction. Thus, the use of two separate Kalman filters is implemented. The first (motion) filter uses the estimate of structure provided by the second filter to estimate the motion. The second (structure) filter uses the estimate of motion produced by the first filter to update the estimate of the structure. The motivations for using dual-estimation are:

- It is natural to remove any coupling between motion and structure, since these states are physically independent.
- The large single state estimate, consisting of motion *and* structure, is divided into two separate states. These states are smaller, and the operations performed on the covariance matrices are faster.

Note that separate process update functions $F_s(\cdot)$ and $F_m(\cdot)$ estimate structure and motion respectively. The measurement estimate function $H(\cdot)$ is used unmodified in both filters.

5.3.2 Implementation of *UKF*

The implementation of the *UKF* is basically for the state, process and measurement. The three Gaussian random variables which are converted into their corresponding sigma point sets are described by

- state: mean $\hat{\mathbf{x}}(k)$ with covariance $P_i(k)$,
- process: zero mean with covariance $Q(k)$ and
- measurement: zero mean with covariance $G_i(k)$.

The matrices P_i, Q and G_i are square matrices with dimensions $(n+13)\times(n+13)$, $(n+13)\times(n+13)$ and $(2\times n)\times(2\times n)$ with an augmented dimension of $n\times n$. These sigma points now take the place of the usual quantities in the Kalman process equations, as described in the Unscented Transform discussion above, where the *a posteriori* distributions are calculated from these transformed sigma points. In more

detail: the *UKF* first concatenates the various quantities $\hat{\mathbf{x}}(k)$, $\boldsymbol{P}(k)$, $\boldsymbol{Q}(k)$ and $G(k)$ into a super state variable $\hat{\mathbf{x}}^a$ and super state covariance matrix P^a,

$$\hat{\mathbf{x}}^a(k) = [\hat{\mathbf{x}}(k)\ 0\ 0], \tag{5.50}$$

$$P^a(k) = \begin{bmatrix} P(k) & 0 & 0 \\ 0 & Q(k) & 0 \\ 0 & 0 & G(k) \end{bmatrix}, \tag{5.51}$$

which is transformed using the forward UT. The algebraic process is as in Chapter 3 and Subsection 3.3.2.1.

5.3.3 Choice of Initial Conditions

The initial conditions of the algorithm include not only the state variables, but also the state error covariance matrices, and the values of the process and measurement noise. The state variables and error covariances are initialized as discussed in Section 5.2.6.

5.4 Comparison: *SVD* Method vs *UKF* Method

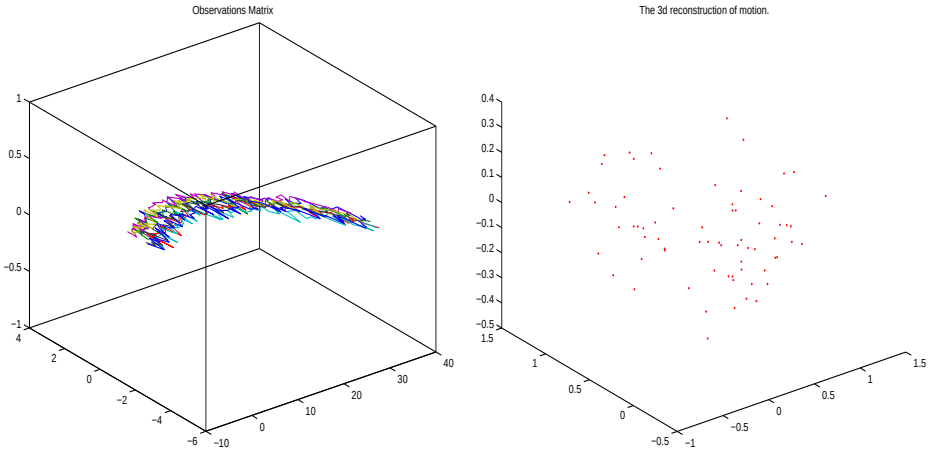
In this section we briefly compare the solution of *SfM* problem using the *Singular Value Decomposition algorithm* and the *Unscented Kalman Filter* approach. We run the experiments using *UKF* algorithm with the same data as used in Chapter 2. The results shown in Figures 5.4 and 5.5 illustrate the advantage of the *Unscented Kalman Filter* against the Factorization Method in solving real problems. The next example does not only show the robustness of the *Unscented Kalman Filter* but also the dependency of this on the number of frames. We run the *UKF* for quasi-real data under perspective projection. First, we track 59 $features$ over 30 $frames$ and the results are shown in Figure 5.6

Again we also track 59 $features$ over 40 $frames$ and the results are shown in Figure 5.7

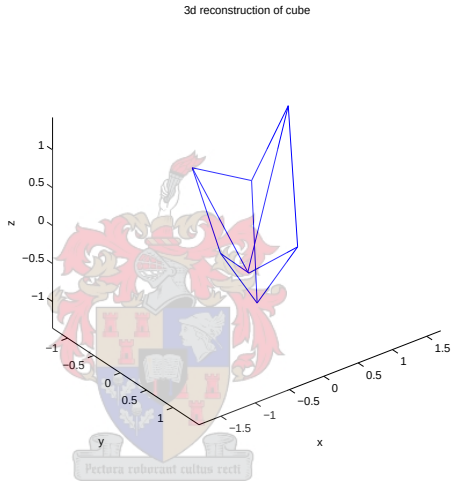
Finally, fifty nine features are tracked over fifty frames using *KLT*, but *SVD* was not able to reconstruct a cube (see Figure 5.8). Comparing Figure 5.8 with Figure 5.6 is clear the advantage of *UKF*.

This is an expected result, as the *SVD* algorithm does not model noise and the model assumes an infinite focal length. The *UKF* models a more realistic camera model and *UKF* explicitly models both process and measurement noise.

Clearly, it takes a while, 80 frames in this case, for the Kalman filter to converge. If more projections are available, then *UKF* converges. The more frames we have the better results we get.



(a) (b)

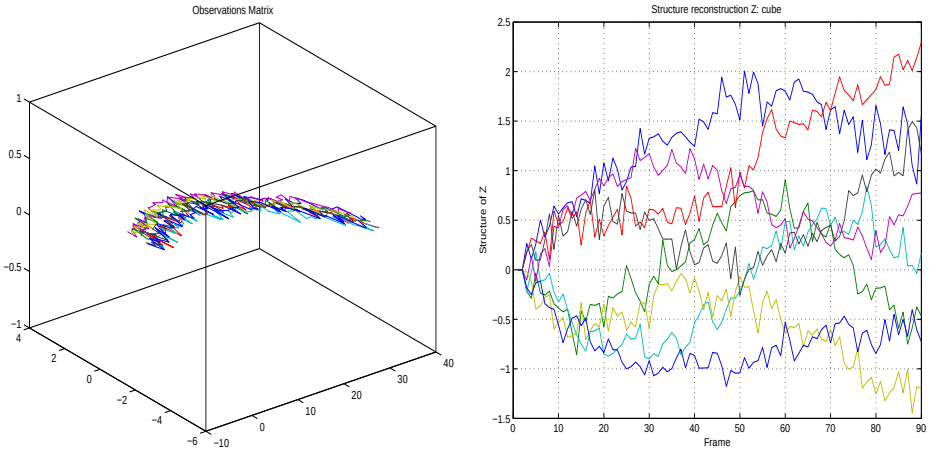


(c)

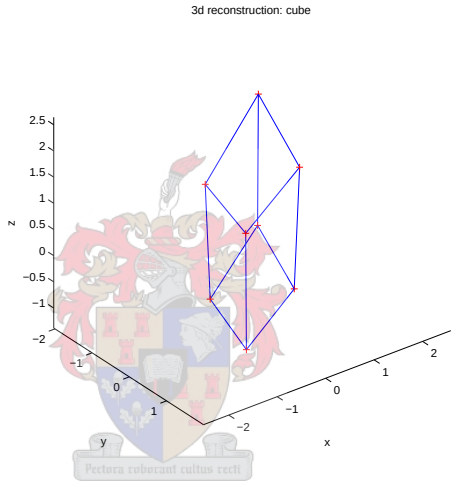
Figure 5.4: *Structure recovering of the cube under rotation and translation with noise of 20 percent using* SVD*: (a) Observations, (b) Motion and (c)* 3D *reconstruction*

5.5 Conclusion

In this chapter we have developed an algorithm which solves real world problems by employing the *Unscented Kalman Filter* algorithm. The algorithm is shown to be robust compared to the *Singular Value Decomposition*. The main advantage of the algorithm developed in this chapter lies in the fact that it models both noise and process, making it suitable to real problems. The algorithm converged after 80 frames but when trying to re-initialize, using the first five frames, the system converges after 75 frames. This is not a very good achievement, therefore another



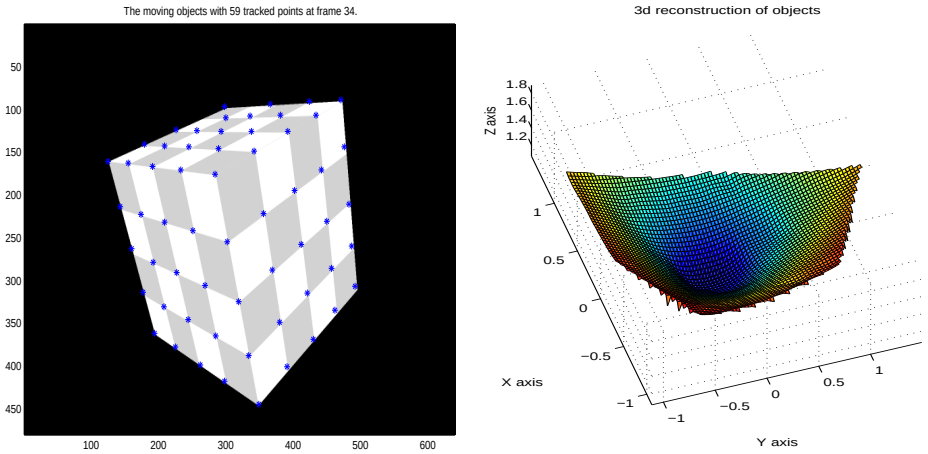
(a) (b)



(c)

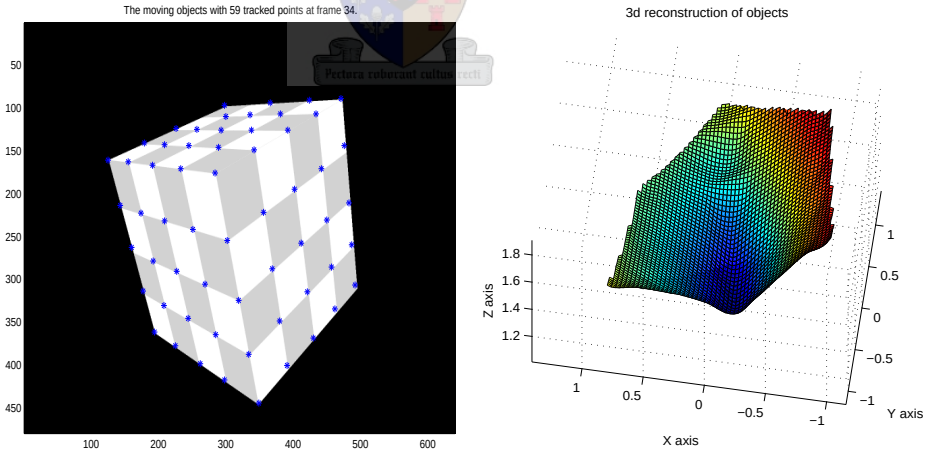
Figure 5.5: *Structure recovering of the cube under rotation and translation with noise of 20 percent using* UKF*: (a) Original image, (b) z-structure and (c)* 3D *reconstruction*

way of improving the *UKF* convergence must be investigated.



(a) (b)

Figure 5.6: 3D *reconstruction and* UKF*: (a) Original image with tracked features and (b)* 3D *reconstruction using* 30 *frames*



(a) (b)

Figure 5.7: 3D *reconstruction and* UKF*: (a) Original image with tracked features and (b)* 3D *reconstruction using* 40 *frames*

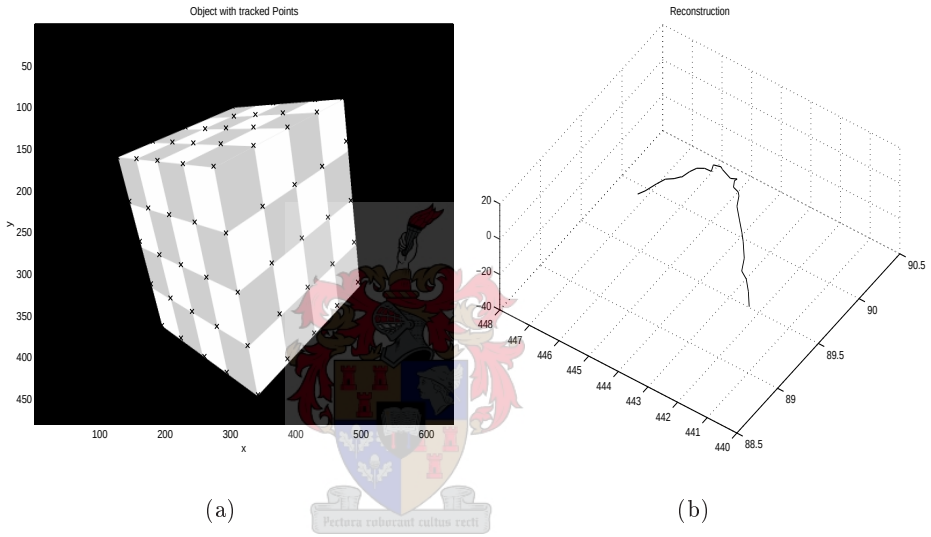


Figure 5.8: *Quasi-real data set: (a) Cube with tracked point using KLT (b)* 3D *reconstruction using data from KLT*

Chapter 6

A Traffic Surveillance System

This chapter, begins with a brief introduction to the traffic surveillance system problem. We describe in detail the techniques we used for image segmentation, particularly background extraction, object clustering and speed computation. We also describe the calibration techniques used in the experiments.

Our main goals are to detect a new object entering in a defined area of interest and then compute its speed. This procedure will also be used to count objects moving past a reference line.

6.1 Requirements

The main requirement of a traffic surveillance system is to identify and track the motion of moving objects. The system must:

- be able to function under a wide range of weather conditions,
- assess traffic conditions accurately,
- operate in real time,
- give a report of variables such as speed, flow and density,
- be able to separate (cluster) objects, and
- be easy to manage or operate automatically.

The implementation or development of a road surveillance system with the above requirements is not an easy task, see e.g. Lorio [84], Papenfus [107], Rittscher et al [117], Coifman and Beymer [35]. If we look at the applicability under a wide variety of weather conditions, the first issue is for example, how to distinguish between a black, green or red car and the background. Other concerns include how to make sure that one is tracking a car and not its shadow. If the cars are very close to each other, how can the system distinguish between them?

6.2 Background Extraction

Our first goal is to extract the background in order to obtain the moving objects. Given a video sequence, the computation of the background consists of determining which features are static. We have applied two ways of computing the background namely, frame differencing and histogram approach. The mathematical representation of an $n \times m$ gray-scale image is specified by nm pixel values arranged as a matrix.

Frame by Frame difference: The frame differencing of two frames at different times reveals regions of motion. Let I be a difference image of two images $I1$ and $I2$. Then we can write I mathematically as

$$I(i,j) = \begin{cases} 0 \text{ if } |I_1(i,j) - I_2(i,j)| \leq T, \text{ where } T \text{ is a suitable threshold} \\ I1(i,j) \text{ otherwise.} \end{cases} \tag{6.1}$$

The value of T was experimentally chosen to satisfy the reduction of noise. The system automatically uses $T = \max|I|/95$.

On histogram we compare the value of a pixel at time τ with another at the same position but at a different time $\tau + i$. In our experiments, we use a camera with a frame rate of 24 frames per second. The video sequences were taken in places where cars were moving with speed above 25 km/h. Since we have a video sequence, and considering that the objects are moving, none would stay in the same place for more than three consecutive frames. Thus, we compute the histogram for every four successive frames. If the pixel value remains the same, then the pixel is considered to be in the background.

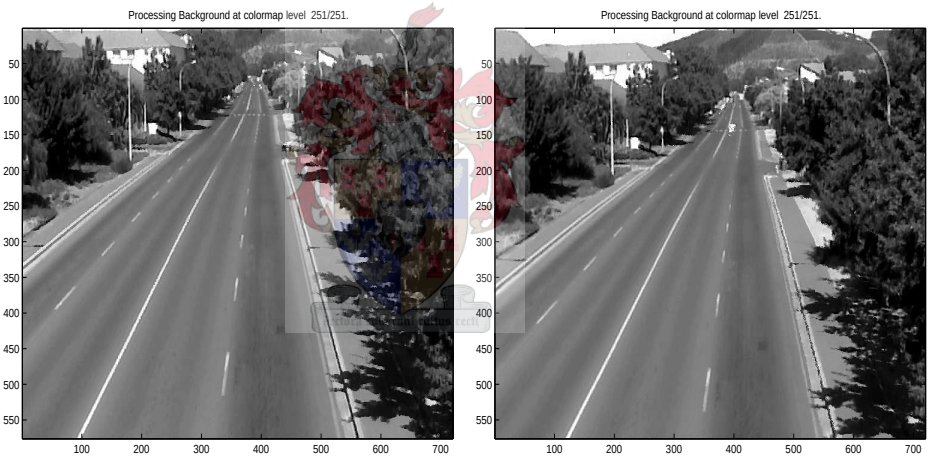
Background computation is sensitive to weather conditions, vibration, etc. For example, the leaves of trees will look like moving objects, Wei et al [159], Lorio [84], Rittscher et al [117]. A difficult problem is when a camera is vibrating, i.e. mounted on a bridge. This requires a statistical approach. It is important to point out that for every video sequence a corresponding background must be extracted. Figure 6.1 shows the general street view taken at the same place using different views in different sequences. The correspondent backgrounds were computed using the histogram approach, Figure 6.2.

Figures 6.3 and 6.4 show the results of subtracting the background from the given frame. The resulting image consists of the moving objects in the sequence.

Next, we show some difficulties that can arise in computing the background. We took some frames on National Freeway (N2) about 40 km from Cape Town. The

(a) (b)

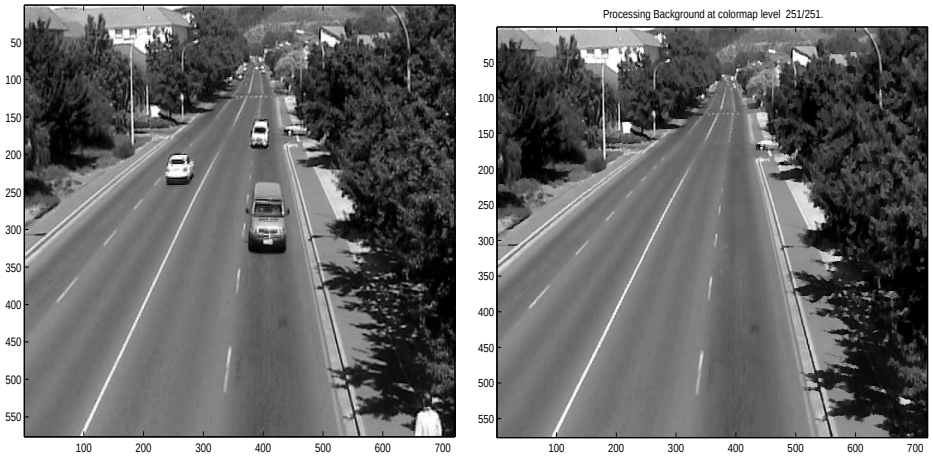
Figure 6.1: *General view of Merriman Avenue: images taken from (a) Video sequence 01 and (b) Video sequence 03 where one car is stopped at the robot*



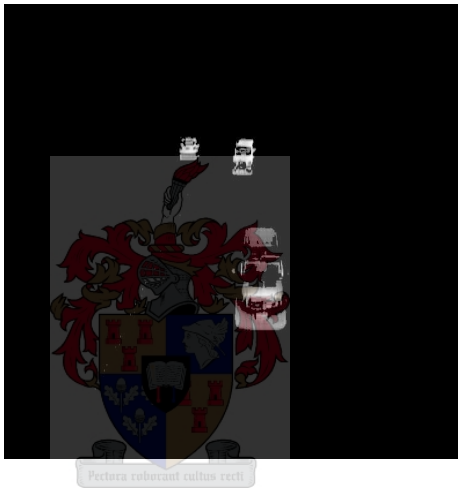
(a) (b)

Figure 6.2: *Two different backgrounds of the same area: Merriman Avenue (a) Video sequence 01 and (b) Video sequence 03*

camera was placed near the bridge and a sequence of 858 image frames was captured. Next we moved the camera to the bridge and we took three more sequences. Sometimes heavily loaded trucks were passing and we could feel the bridge vibrate. We were able to compute the background using histogram techniques (see Figure 6.5), but in the case of the vibrating camera, the histogram fails (see Figure 6.6). Clearly, the difficulty was due to the camera vibration and the objects' movement. As a



(a) (b)

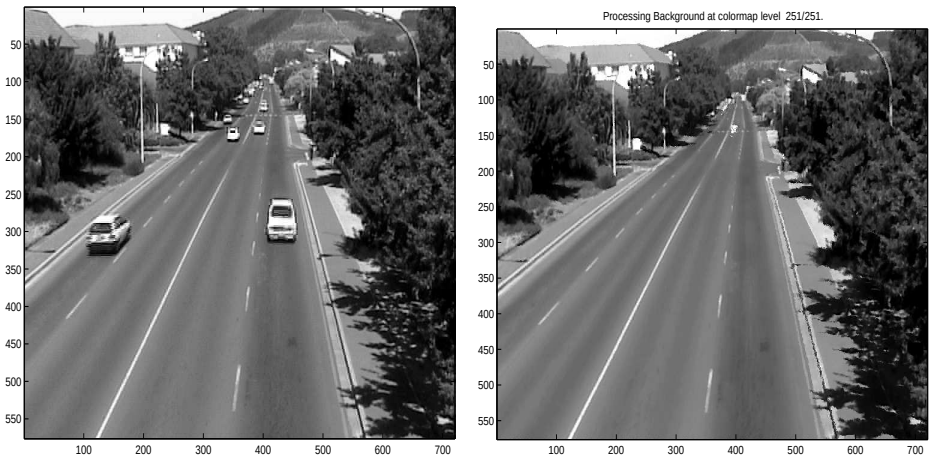


(c)

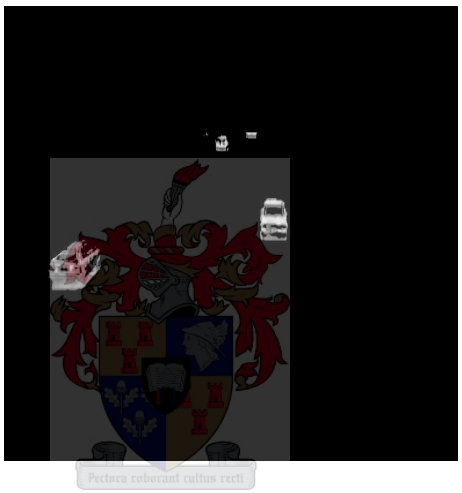
Figure 6.3: *(a) General view of Merriman Avenue, (b) Computed background and c: Moving objects for video sequence 01*

result, everything seems to be moving. In figure 6.7, we present the result obtained using frame by frame difference.

This result shows us that the histogram technique does not work correctly in all situations. Therefore, we suggest to use frame subtraction. The frame subtraction proved to be convenient for determining moving objects. In our implementation the user is free to chose one or the other. Figure 6.8 shows how well the system could detect the moving objects under camera small vibration (sequence 3).



(a) (b)

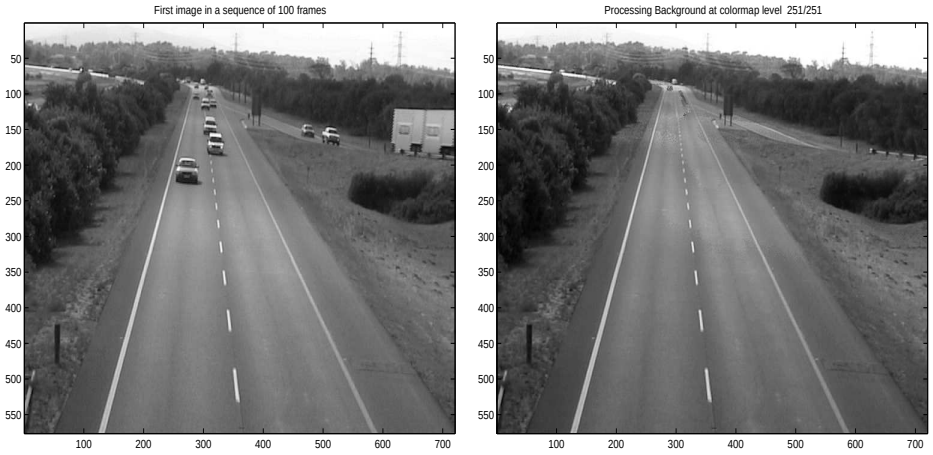


(c)

Figure 6.4: *(a) General view of Merriman Avenue, (b) Computed background and (c) Moving objects for video sequence 03*

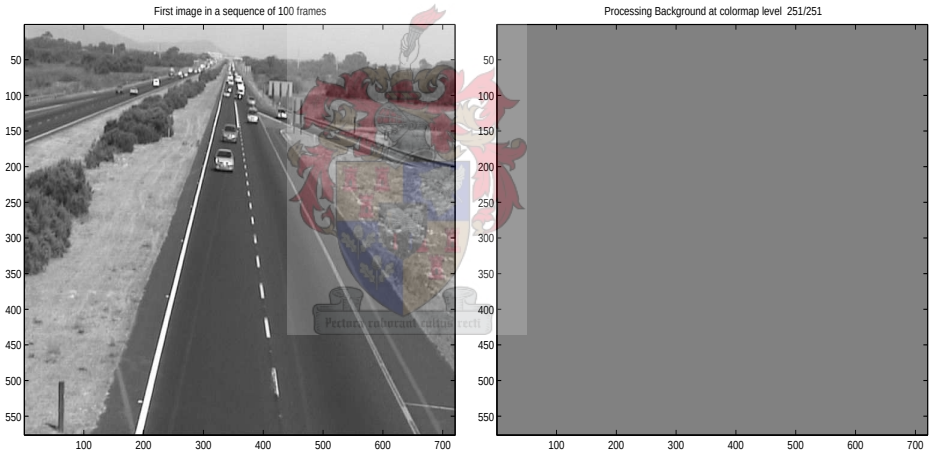
6.3 Objects Clustering

The second goal is to be able to count the number of objects and compute the speed of objects. The features must belong to a single object. In order to achieve this we need to apply the clustering algorithms. Simply speaking, k-means clustering is an algorithm to classify or to group objects based on attributes/features into k groups, where k is a positive integer number. The grouping is done by minimizing the sum of the squares of the distances between the data in each cluster and the corresponding cluster centroid.



(a) (b)

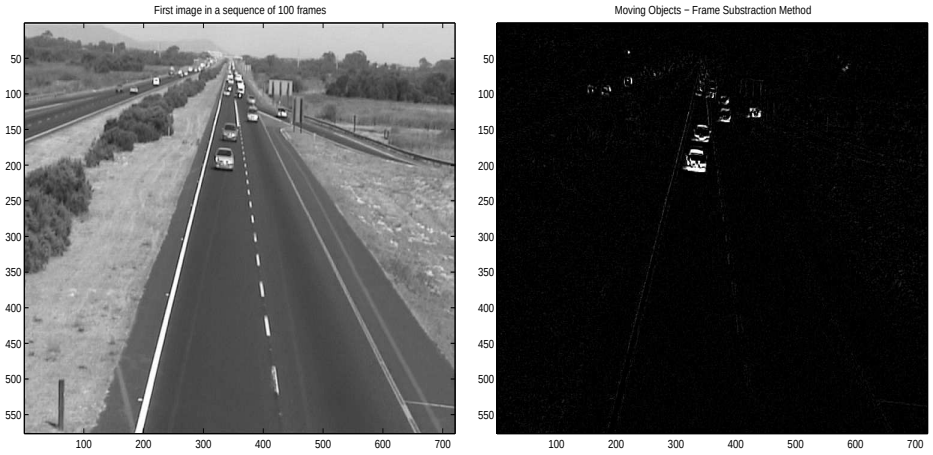
Figure 6.5: *(a) General view of N2 and (b) Computed background using histogram - sequence 01*



(a) (b)

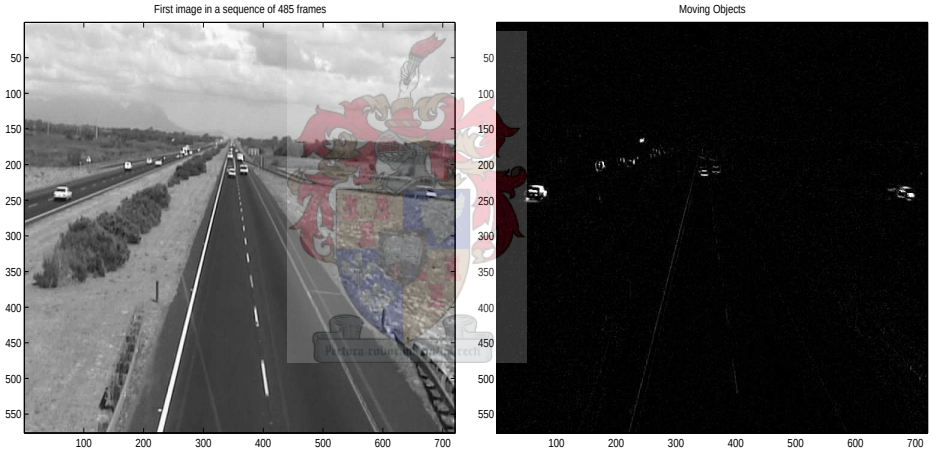
Figure 6.6: *(a) General view of N2 and (b) Histogram fails to compute background - sequence 02*

After removing the background the system detects the corners of objects. The next problem is to determine which feature belongs to what object. Imagine that we have an $n \times m$ image with objects in it. If we know how many objects are in that image, then for every object's center ($\mathbf{C}_{xy}$), for fixed value r, all the features inside of a circle of radius r will be considered to belong to the same object $\|\mathbf{C}_{xy} - \mathbf{p}_i\| \leq r$.



(a) (b)

Figure 6.7: *(a) General view of N2 and (b) Robustness of subtraction method - sequence 02*



(a) (b)

Figure 6.8: *(a) General view of N2 and (b) Detected moving objects for sequence 03*

In our implementation, we use our *Matlab* code ($cluster\_flow.m$) to perform the clustering. Note that r must be chosen not to be very large[1] because this will allow some points that belong to another object to be within the radius. This is the case when two objects are close to each other. The most difficult case is when we don't know the number of objects and we want to determine the number of objects involved

[1] Compared to the maximum size of objects to be separated.

in the scene. Automatic clustering techniques have received great attention and are available, see e.g. Muller [102], Barfoot and D'Eleuterio [8], Manolis and Jeffrey [93]. Since we have image points, clustering can be easily achieved by applying a k-means algorithm over the detected features with the help of (6.4) below. The robustness of the system will depend on how many features are detected; the more we have the better the clustering result. In practice, before a clustering technique can be applied some relation between features must be defined. Consider a set of m features at frame i, that is

$$X_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,m} \\ y_{i,1} & y_{i,2} & \cdots & y_{i,m} \end{bmatrix}. \tag{6.2}$$

The *2D* data has to be used to determine a measure of association between points. So the similarity value r can be calculated from the distance between these m features in the image as

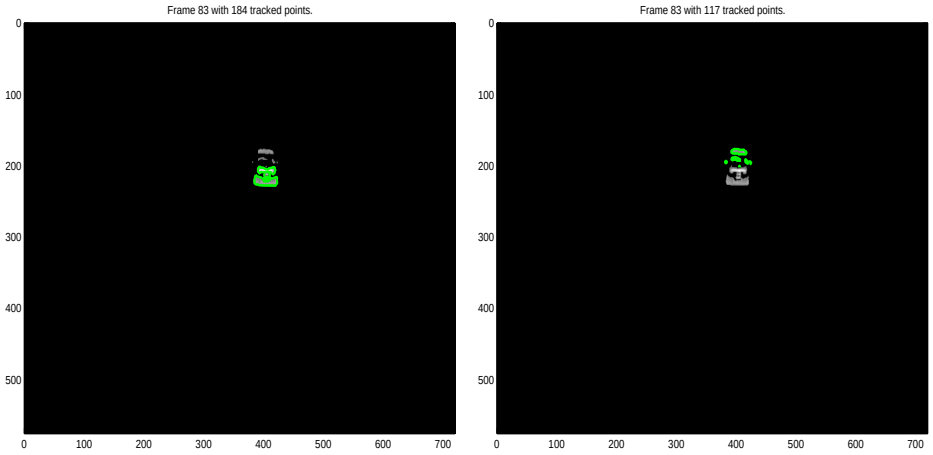
$$r_{k,j} = \sqrt{(x_{i,k} - x_{i,j})^2 + (y_{i,k} - y_{i,j})^2}. \tag{6.3}$$

A set of observations for which the pairwise distance r is smaller than the given threshold is considered a cluster. Based on our experiments, we use a value of 12 as our threshold. Then the clustering algorithm will use the similarity matrix S which is computed using sample standard deviation as

$$s_{k,j} = \sqrt{\frac{\sum_{p=1}^{m}(r_{k,j} - r_{p,j})^2}{m-1}}. \tag{6.4}$$

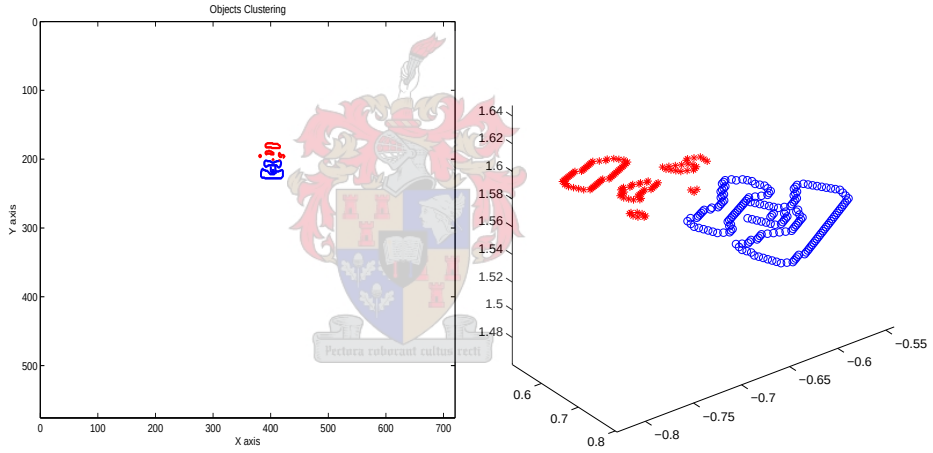
We note that (i) Small values correspond to features inside of a circle with center a reference point indicating that both belong to the same object; (ii) Features far away from each other will be considered not to belong to the same object; (iii) k-means improves the clustering, moving features from one cluster to another and this reduces the total clustering error (sum of individual cluster errors).

Features belonging to the same object can be misclassified or assigned to different clusters (see e.g. Muller [102], Manolis and Jeffrey [93] and Lorio [84] for more information and applications). Misclassified features seldom cause problems during the reconstruction phase, Magaia et al [91]. In Figure 6.9 we have one car but it was classified as two clusters. It shows as two clusters. In Figure 6.10a we display the two clusters together, and in Figure 6.10b the *3D* reconstruction. The two images are displayed in the same figure, enabling us to see the resulting image. The clustering parameters can be determined experimentally but we would rather have



(a) (b)

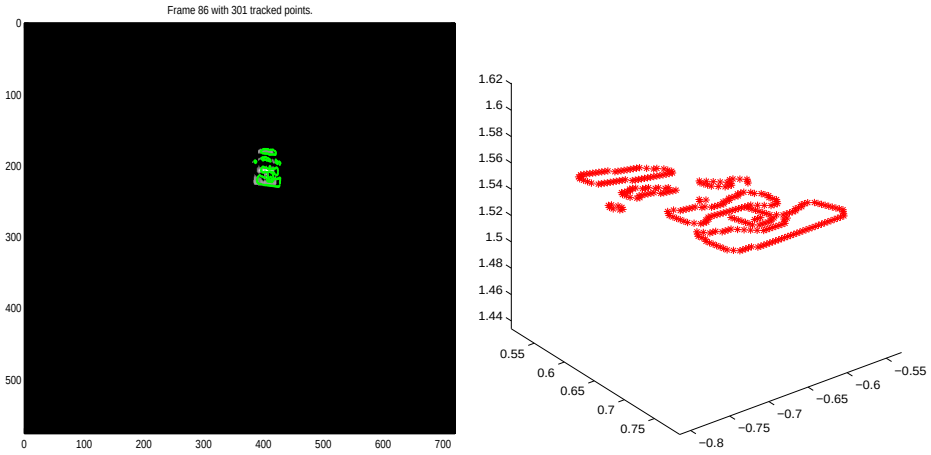
Figure 6.9: *Clustering using a bad parameter: (a) First cluster and (b) Second cluster*



(a) (b)

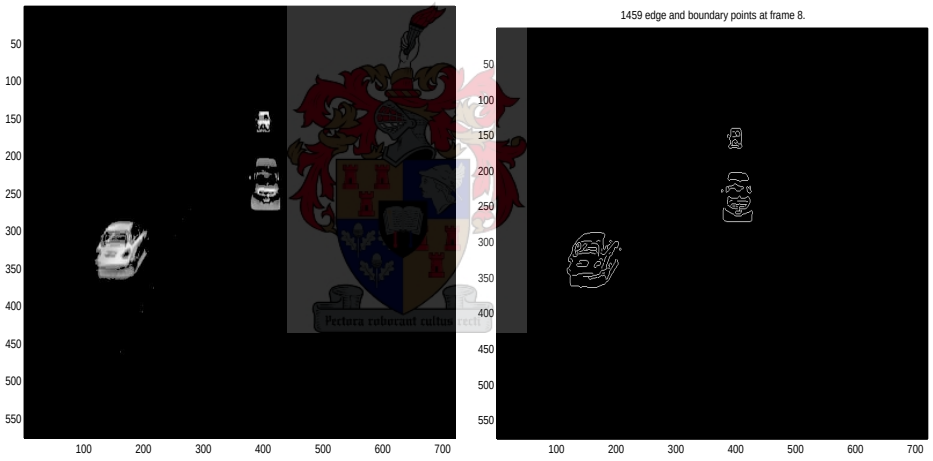
Figure 6.10: *Clustering using a bad parameter: (a) One object detected as two clusters and (b) Reconstruction of the two clusters*

too many clusters than too few. Next we increase the clustering parameter and we were able to detect the whole object as a single object (see Figure 6.11). We have tested our system using images which contain more than one object, and the result can be seen in Figure 6.13. The number of clusters k was automatically computed using available *Matlab Toolboxes*, for instance the Image Processing Toolbox and the Statistical Toolbox. We provide *Matlab* with the Euclidean distance parameter r between 9 and 13 to compute the number of feature groups.



(a) (b)

Figure 6.11: *Clustering using a good parameter: (a) One object classified as one cluster and (b) Reconstruction of the cluster*

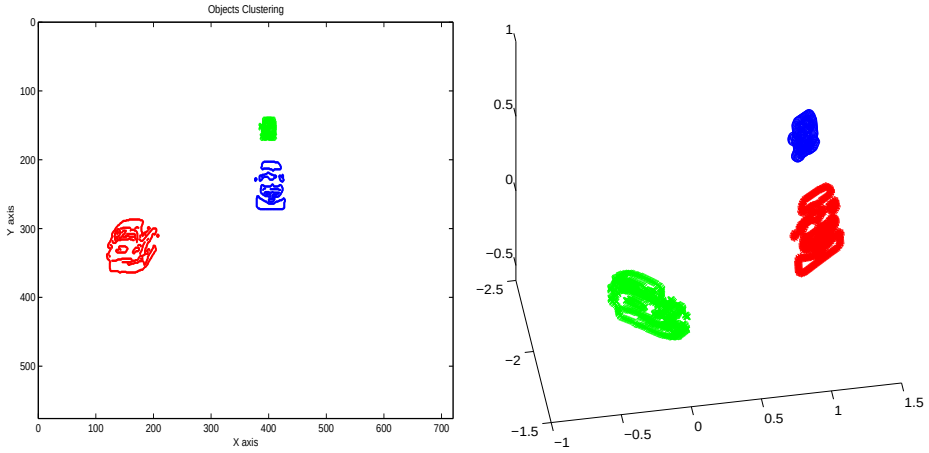


(a) (b)

Figure 6.12: *Clustering using a good parameter: (a) Original image and (b) Corners of objects*

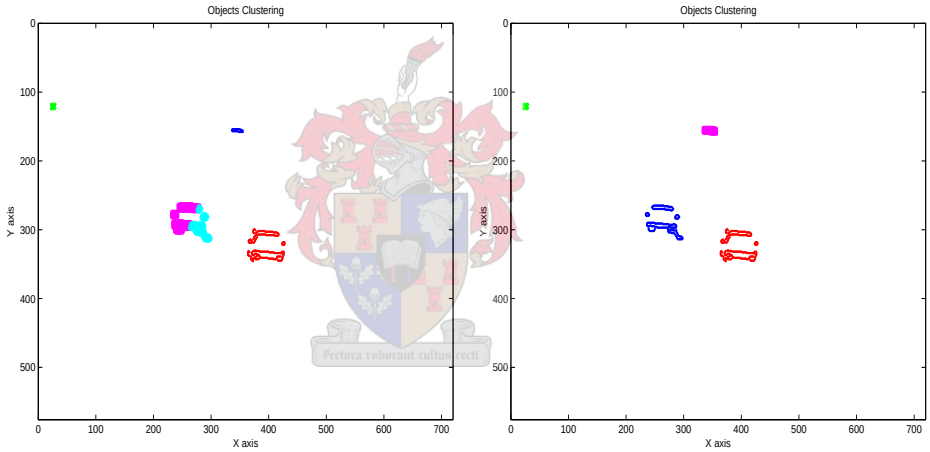
The clustering performance is summarized in the table 6.1. The clustering parameters were taken randomly with the intention of analyzing the clustering robustness[2]. Note that we have applied the same parameters to all the tested images. The general performance was counted manually. We count the number of objects in the image

[2]There is always a parameter that can fit the image best.



(a) (b)

Figure 6.13: *Clustering using a good parameter: (a) clusters (b) the shape of clustered objects*



(a) (b)

Figure 6.14: *Clustering: (a) using a bad parameter and (b) using good parameter*

and then we count the number of correctly classified. The percentage of correctly classified gives us the performance. The *3D* reconstruction does not depend on the cluster parameter. Plotting shapes in the same graph will result in a *3D* object.

The clustering steps can be summarized as:

1. Place k points into the space represented by the objects that are being clustered. These points represent initial group centroids;

2. Assign each feature to the group that has the closest centroid;

Location	*Seq*	*Total images*	*Mis-clustered*	*General Performance*
Merriman	3	26	6	88.46
Highway N2	1	28	9	83.93
Highway N2	2	16	5	84.38
Highway N2	3	16	6	81.25
Highway N2	4	24	6	87.50
Highway N2	5	8	2	87.50
Highway N2	6	28	8	85.72
Road R102	1	16	7	78.13
Road R310	1	1	1	50.00
Road R310	2	8	0	100.0

Table 6.1: *Summary of clustering experiments.*

3. When all features have been assigned, recalculate the positions of the k centroids;

4. Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the features into groups from which the metric to be minimized can be calculated.

6.4 System Calibration

The system calibration is crucial for traffic surveillance systems, particularly for *3D* estimation accuracy, Han and Kanade [56], Luong et al [87], Luong and Faugeras [88]. The *3D* reconstruction only gives us the relative size of the object and speed. We need the calibration to calculate the absolute values. Therefore, in our experiments we use data taken from known moving cars. We use known speed rather than distance because it would be difficult to determine the correct position of the car using its projections.

We take a video sequence of two known cars and we measure their speed and *3D* data. Calibration consists of determining the *3D* scale factor. Note that our *3D* algorithm recovers both shape and linear velocities. Therefore, using the recovered *3D* structure, we are able to compute the car's velocity at a given time and the correspondent driven distance d. We observe where the *3D* reconstruction of the motion or shape begins to stabilize and we assume the *3D* data to be correct only from that point forwards (see Figures 6.16 and 6.18), Magaia et al [91]. Calibration should also be possible using known measurements (size of the car, etc.) as is shown next. Note that when the system has converged the shape remains constant over

(a) (b)

Figure 6.15: *System calibration; (a) Car with known data; (b) Car in the same sequence whose data we estimate*

time.

Recalling the *SfM* problem, in Figure 6.22a we see that the *3D* reconstruction of the motion was correct, and so is the shape in Figure 6.22b. Now this part of the system is used to determine the speed, and from *3D* data we can also approximately compute the car's dimensions, such as the width and height. Figure 6.15a shows one of the cars used to calibrate our system.

We take a sequence containing two cars with known data (speed), since our system recovers both shape and speed. We run the test over one of the known sets of data (car) and we save the computed distance and speed. In order to test our code, we run the same experiment using saved data for the other car in the same sequence. The results confirmed that our code was correct. Next we run another experiment using data of the car in Figure 6.15a over other cars in the same sequence, i.e. Figure 6.15b. Then we apply the system to any other moving objects. The computation of speed can be done using the following equation $V_2 = \frac{d_2 \times V_1}{d_1}$, where V_1, d_1 and d_2 are known speed and distances and V_2 is the unknown speed. Note that the distances d_1 and d_2 are determined by calibrating the distance in time interval where the speed is calculated. It is important to point out that if the camera moves or the camera's focal length changes (zooming) the system must be re-calibrated.

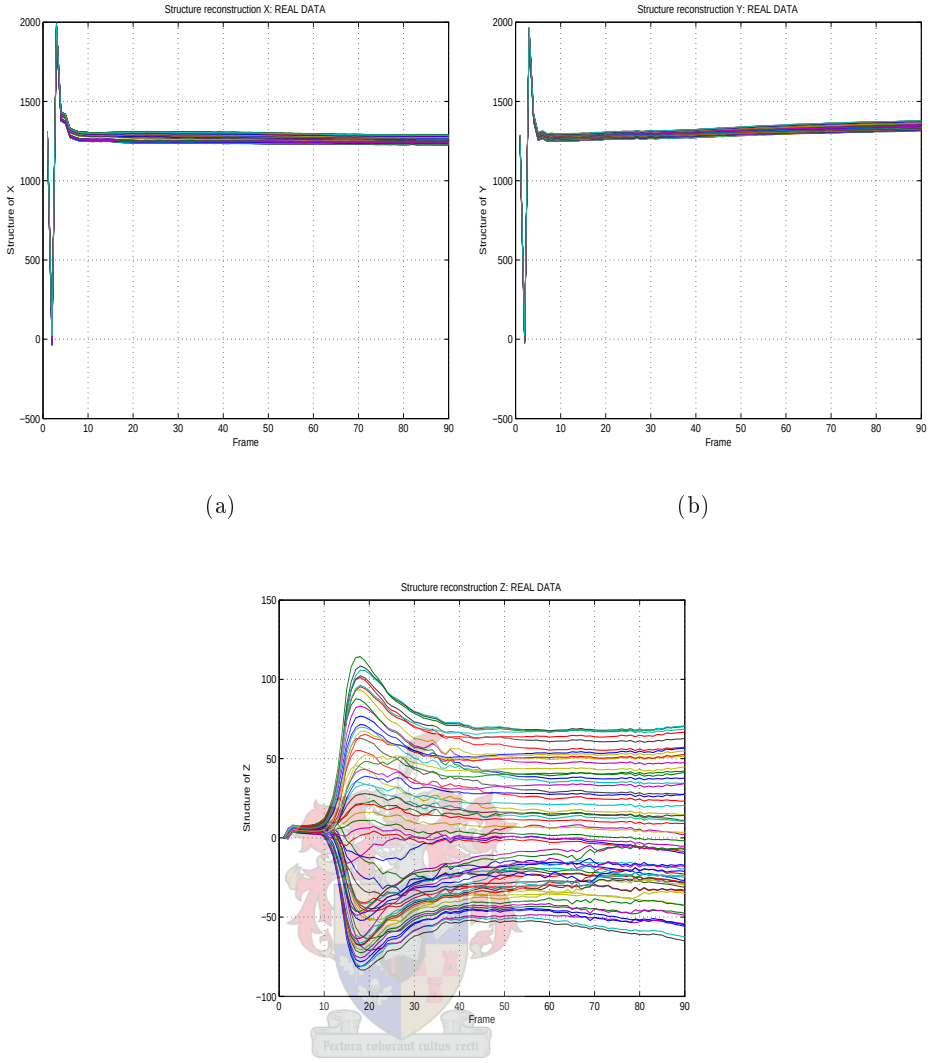
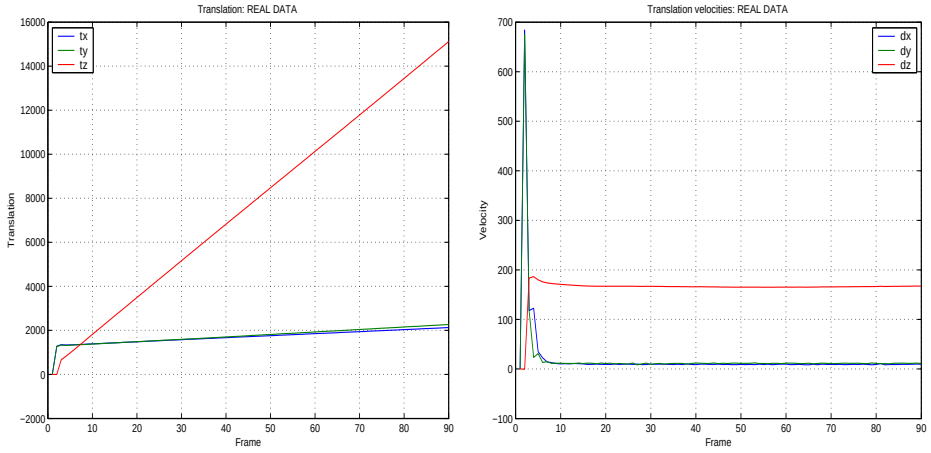


Figure 6.16: *Structure recovering of objects in data set 01: (a) x-coordinate, (b) y-coordinate and (c) z-coordinate*

6.5 Discussion

The problem of vehicle monitoring can be seen as a problem of feature finding, clustering and speed computation over a given time interval. There are two main problems in object tracking. The first is related to which prominent features to track, and the second to the length of time those can be tracked. The computation of corners has another advantage because one can determine what kind of object[3]

[3] with a large number of detected features



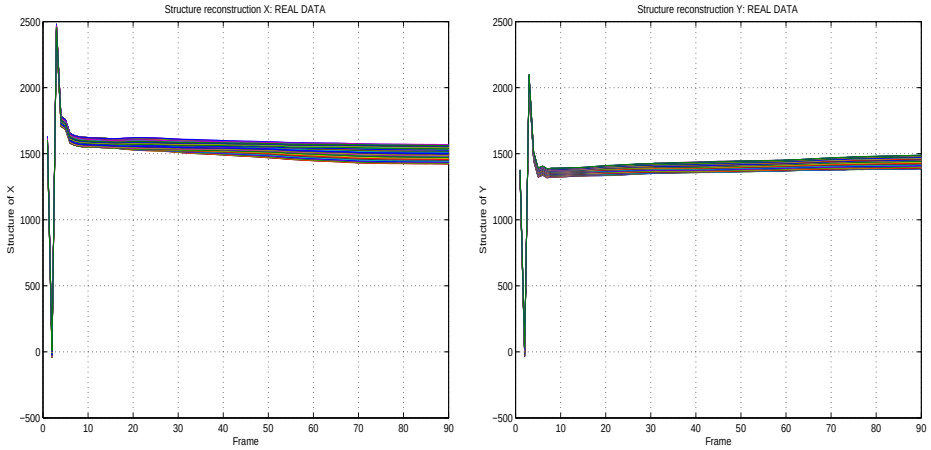
(a) (b)

Figure 6.17: *Structure recovering of objects in data set 01: (a) Translation and (b) Linear velocity*

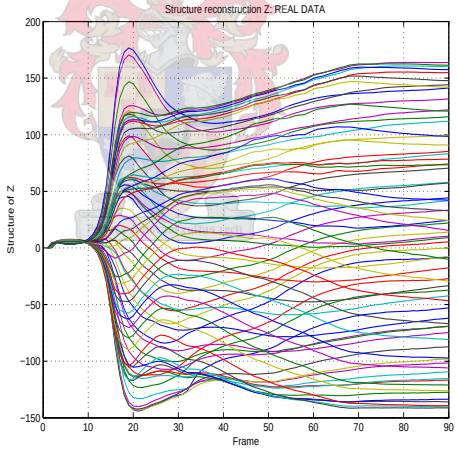
is in the scene. Clearly, it is possible to distinguish between a car, a person and a motorbike. This makes the system suitable for many surveillance applications. The computation of an object's speed was computed for every eighty frames. We were able to determine the speed of each car involved in the scene. In figures 6.26 and 6.27 we can see how the shadow has some influence on features detection and reconstruction.

Apart from these contributions, another important feature is the ability to recover the shape of that object (see Figure 6.28). From the shape we can see that it is one object, as two or more parts will have almost the same speed and correlated dimensions. Now we consider a case in which more than one object is involved. Again our system was able to recover the shape efficiently; see Figures 6.27 and 6.28.

Vehicle counting was also tested. We defined an area of interest in which we were able to count objects entering or leaving the scene.

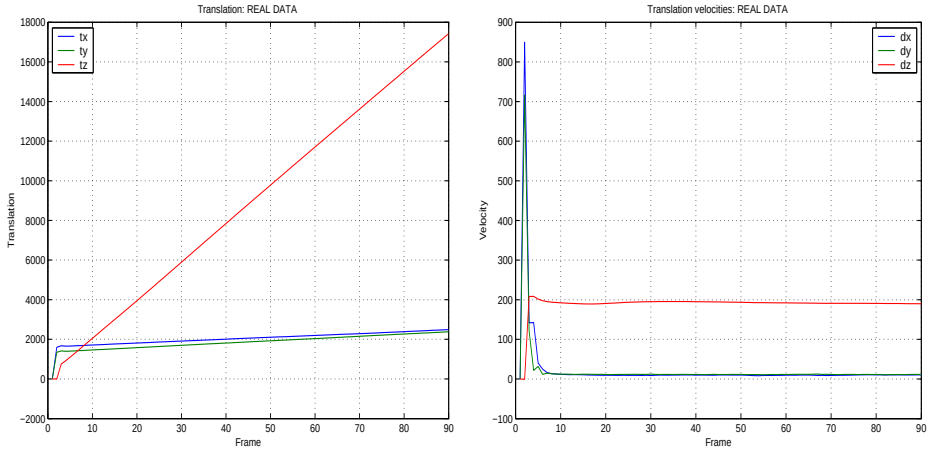


(a) (b)



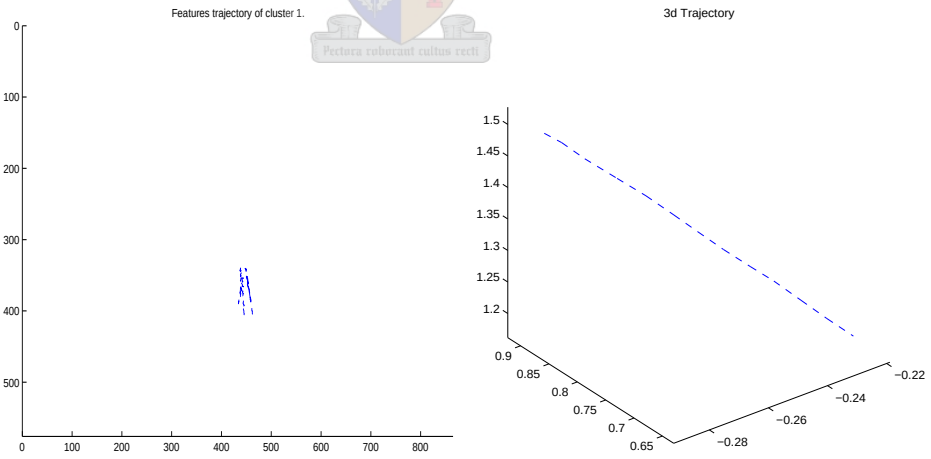
(c)

Figure 6.18: *Structure recovering of objects in data set 03: (a) x-coordinate, (b) y-coordinate and (c) z-coordinate*



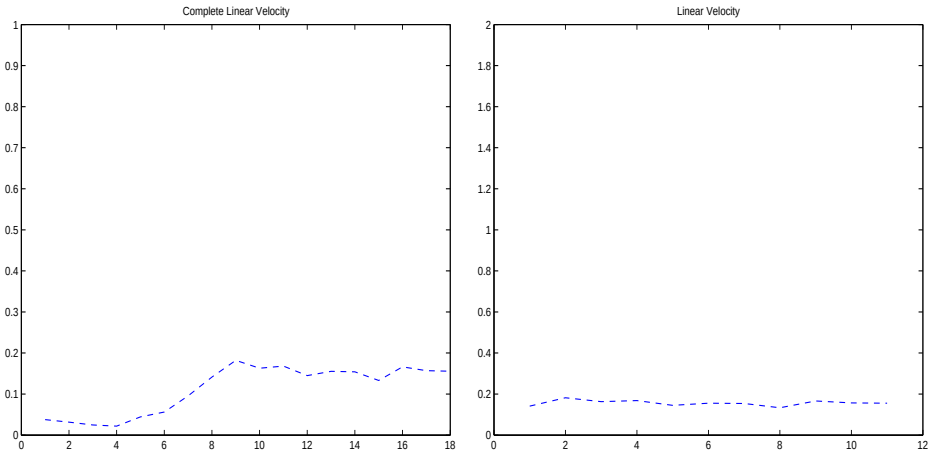
(a) (b)

Figure 6.19: *Structure recovering of objects in data set 03: (a) Translation and (b) Linear velocity*

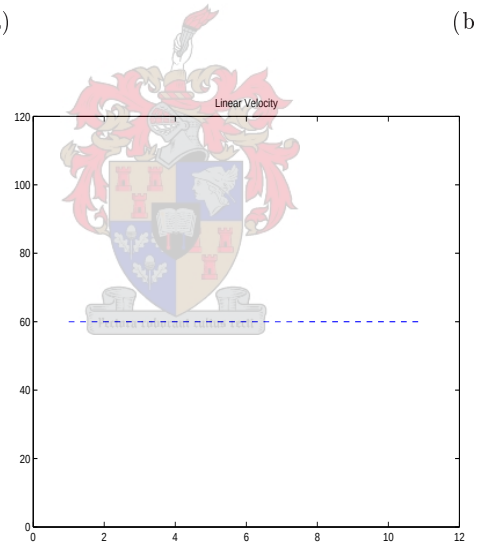


(a) (b)

Figure 6.20: *Calibration: (a) Feature trajectories; (b)* 3D *stabilized trajectory*

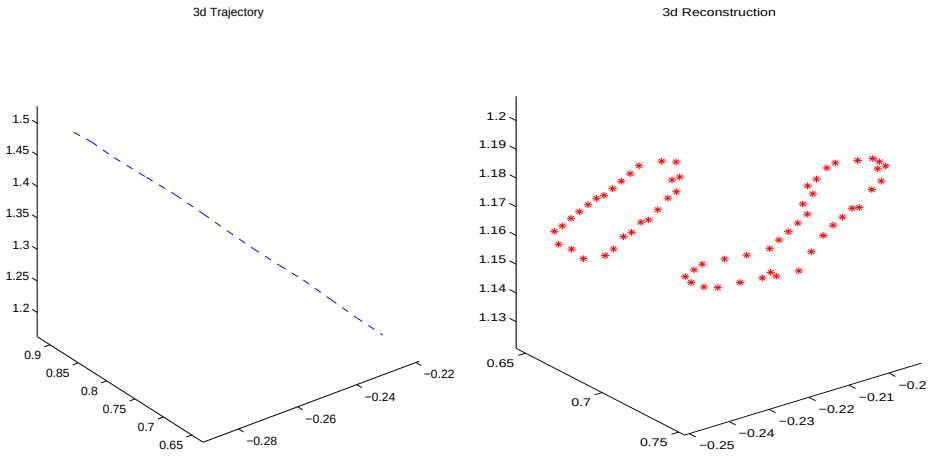


(a) (b)



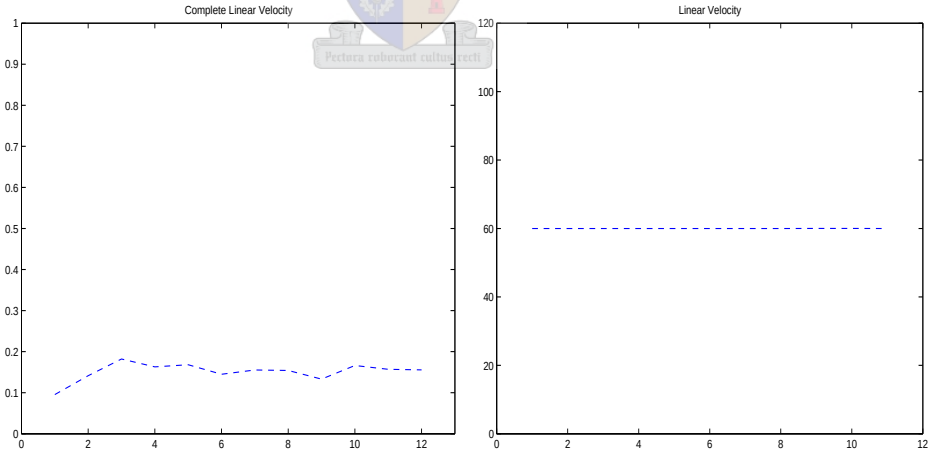
(c)

Figure 6.21: *Calibration: (a) Velocity computed over entire interval; and (b) Unscaled and stabilized velocity, (c) Scaled and stabilized velocity*



(a) (b)

Figure 6.22: 3D *Reconstruction of: (a) Trajectory and (b) Shape*



(a) (b)

Figure 6.23: *Calibration: (a) Non calibrated speed (b) Calibrated velocity*

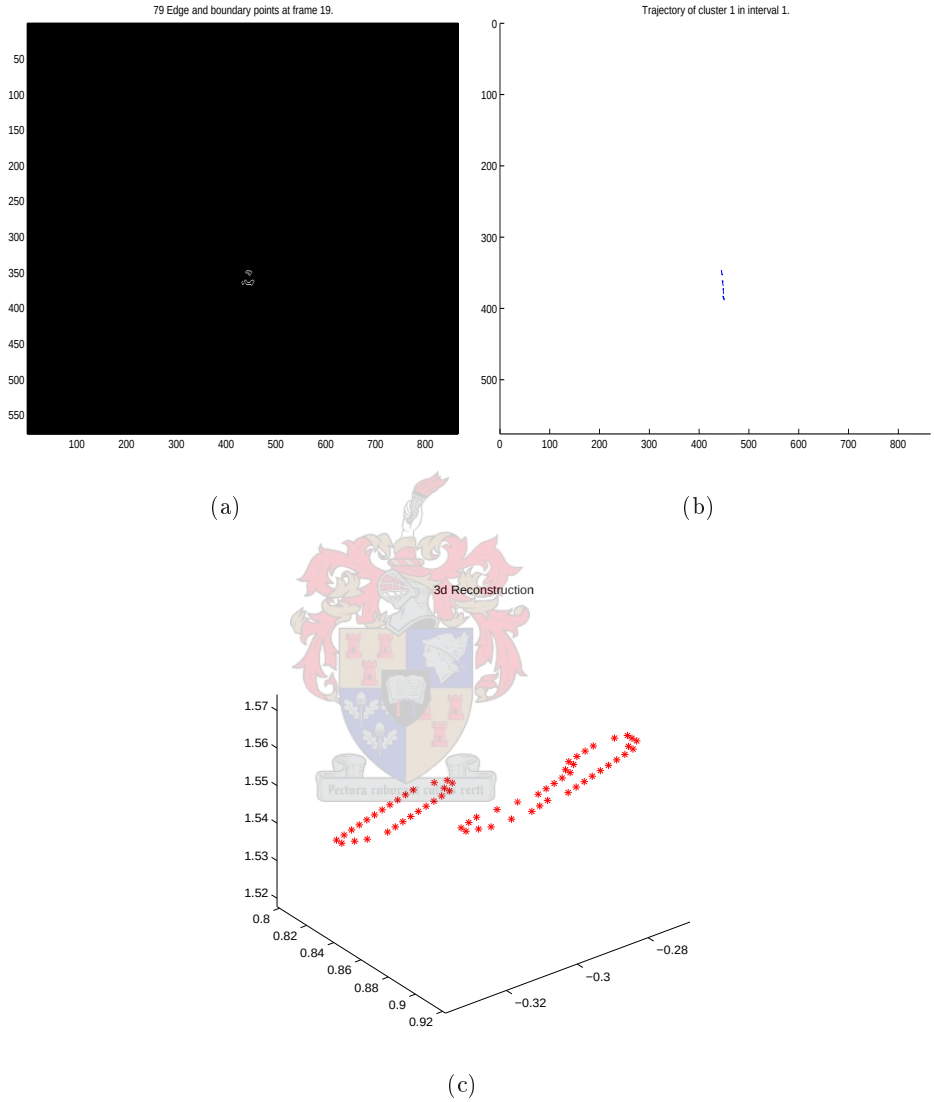


Figure 6.24: *(a) Original image, (b) Trajectory and (c) Shape*

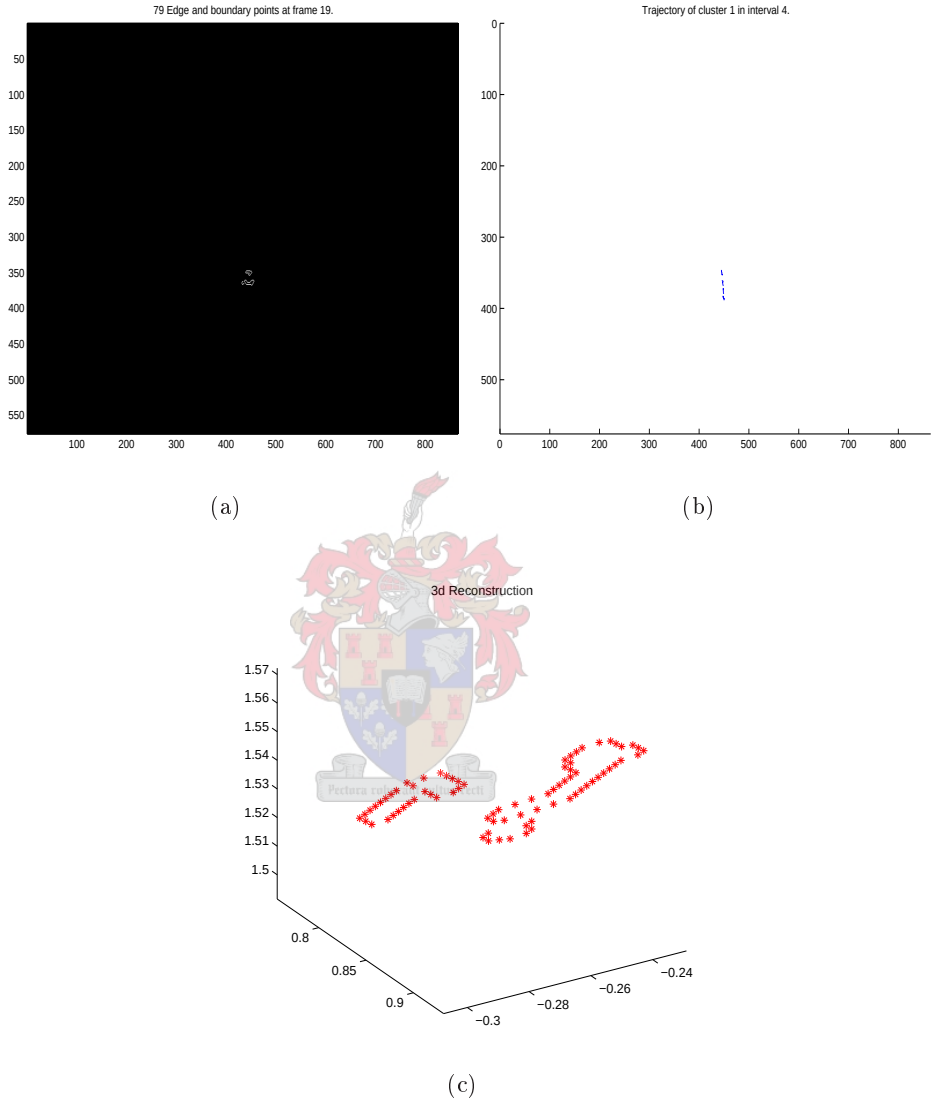


Figure 6.25: *(a) Original image, (b) Trajectory and (c) Shape*

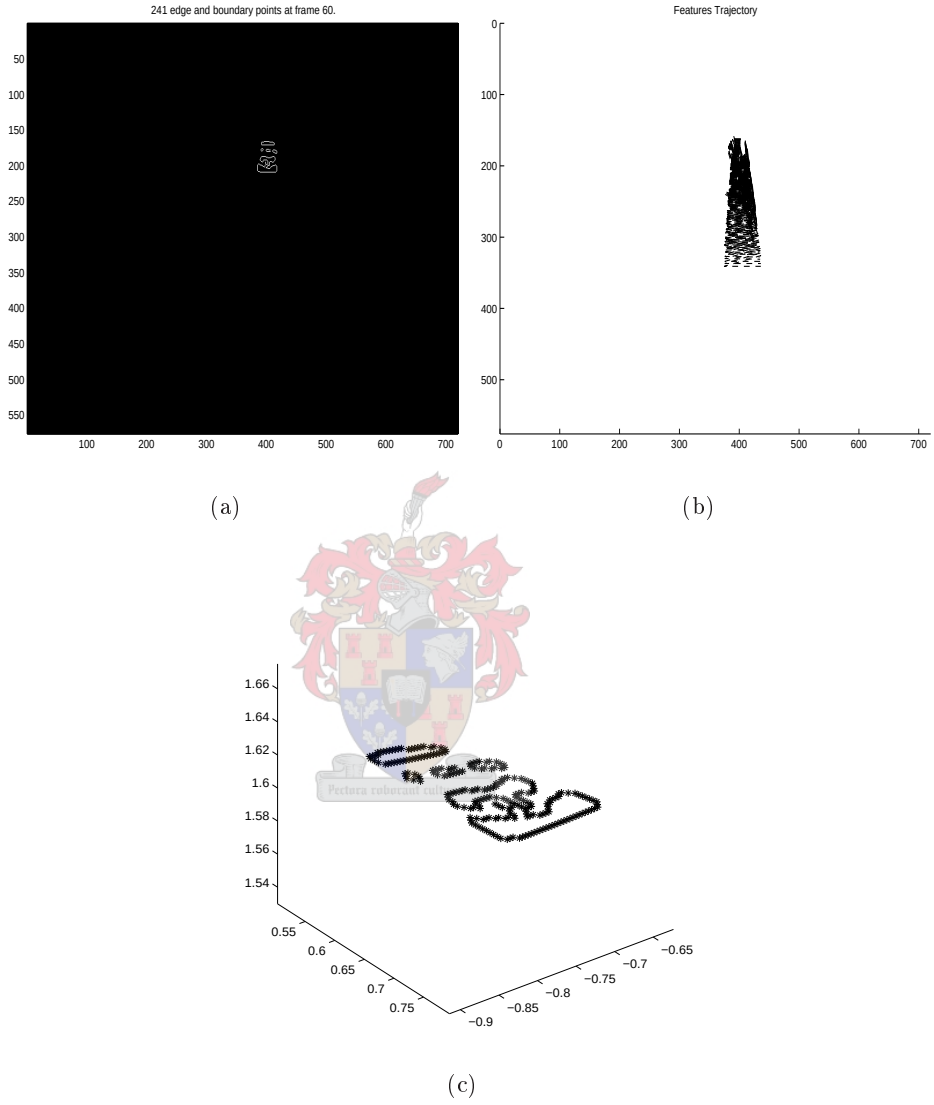
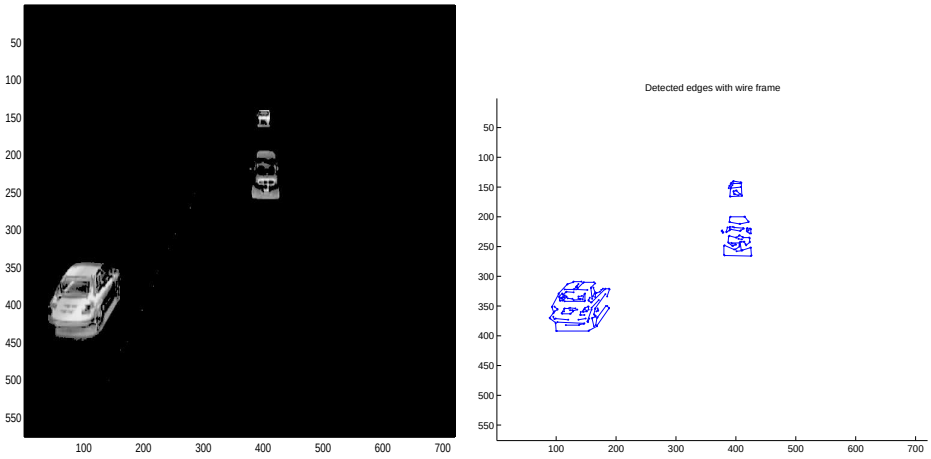
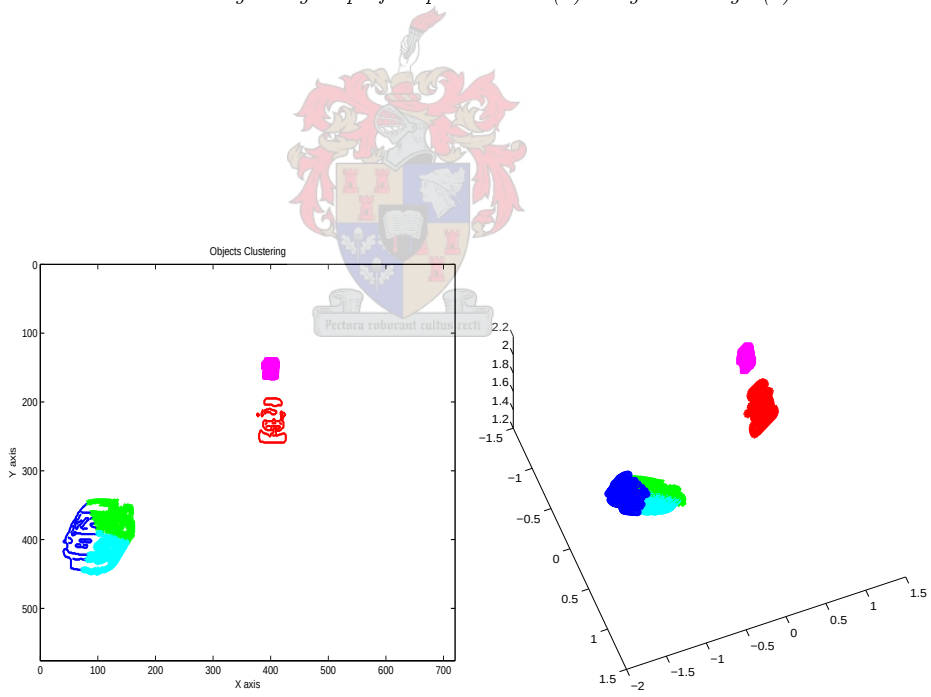


Figure 6.26: *(a) Corners detection, (b) Trajectory and (c)* 3D *reconstruction*



(a) (b)

Figure 6.27: *Clustering using imperfect parameter: (a) Original image (b) Detected corners*



(a) (b)

Figure 6.28: *Clustering using imperfect parameter: (a) One object misclassified (b) Recovered shapes*

Chapter 7

Experimental Results

In this section, the results of various practical experiments are discussed. These experiments include:

- Synthetically generated input, ranging from pure translation in all three axes, to pure rotation around all three axes, to various combinations of translation and rotation.
- A synthetically generated video sequence, processed with feature tracking software, and fed into the algorithm.
- Real-world video-footage, processed with a feature tracker, and fed into the algorithm.
- Analysis of convergence of the *Unscented Kalman Filter* algorithm using both synthetic and real data sets.

7.1 General Goal

The main goal is to develop the video-based traffic monitoring system. In order to achieve this goal we will use the following techniques (i) multiple object clustering, (ii) object tracking, (iii) *3D* reconstruction from a sequence of video images.

We are also interested to evaluate this algorithm in terms of structure estimation accuracy as well as speed computation.

7.2 Results

The video sequences were taken in the following places

- on one road in Stellenbosch (Merriman Avenue),

- on several Western Cape Roods (Provincal Roads), namely the R44, the R102 and the R310,
- on one of South Africa's long National Freeways (N2).

7.2.1 Merriman Avenue - Sequence 01

This sequence consisted of 200 frames and the background was computed using a histogram approach. Then the moving objects were computed by successively subtracting the background from each frame, filtering the result and then cutting some undesirable features (statistical approach). Figure 7.1a shows the general view of Merriman Avenue in Stellenbosch, Figure 7.1b shows the computed background. Figure 7.1c was obtained first by computing the difference between the background and the road.

7.2.2 Merriman Avenue - Sequence 02

This sequence had only 99 frames. The background was computed using the histogram algorithm. Then the moving objects were computed by successively subtracting the background from each frame. Filtering the result gives Figure 7.2.

7.2.3 Merriman Avenue - Sequence 03

This sequence had 122 frames. The background was also computed using the histogram. Then the moving objects were computed by successively subtracting each frame from the background, filtering the result as seen in Figure 7.4.

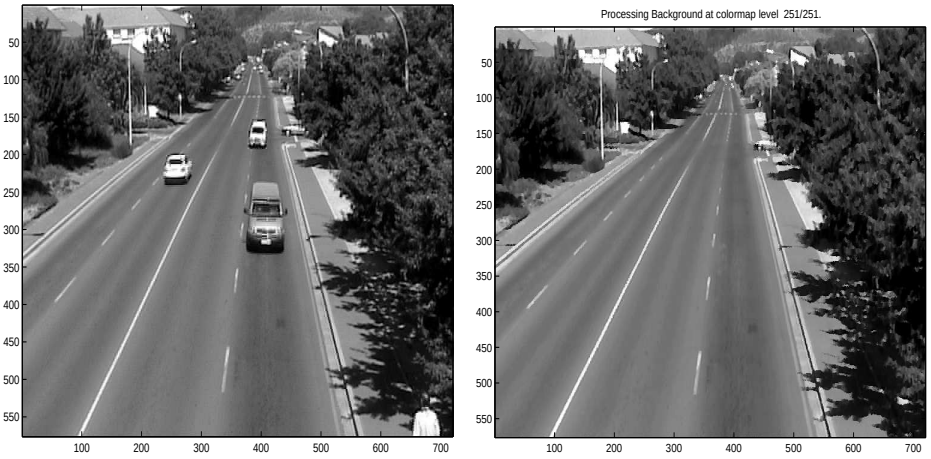
7.2.4 Merriman Avenue - Sequence 04

The sequence consisted of 200 frames. This is one of the sequences where both methods (histogram and frame by frame difference) gave the same result. As a result we ended up with moving objects only, as in Figure 7.5. In Figures 7.5b and 7.5c we can see that even two moving people were clearly detected.

7.2.5 National Freeway N2

This sequence is one of the longest with 802 frames. The moving objects were computed by successively subtracting each frame from the next one. As a result we ended up with the background, as is seen in Figure 7.6. This figure illustrates the original image and moving objects only.

We will not describe other sequences, as we have used the same procedures. Next we show the results obtained from other Western Cape Roads, the R102 and R310.



(a) (b)

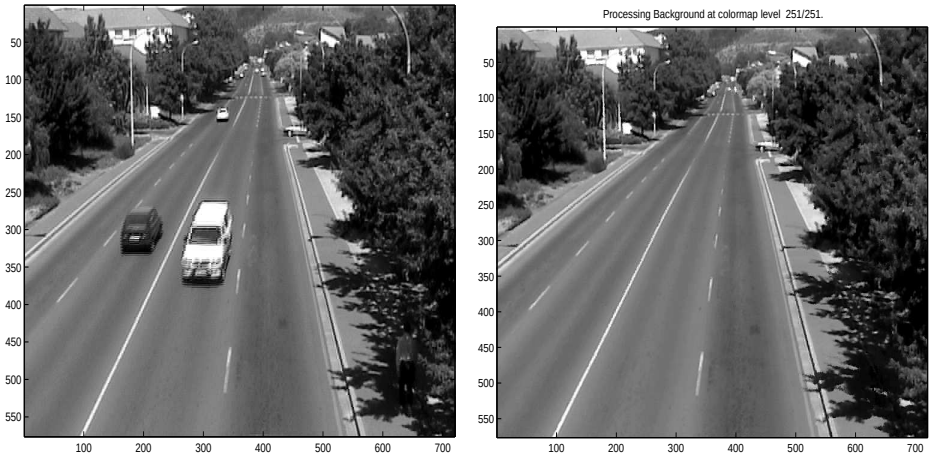
(c)

Figure 7.1: *(a) General view of Merriman Avenue, (b) Computed background and (c) Moving objects - sequence 01*

7.3 *3D* Reconstruction and System Convergence

In this section we tested the time taken for the algorithm (UKF) to converge. This information is crucial to determine the number of frames required for the system to deliver the desired results. First we tested the system with purely synthetic data, as was done in Chapter 2, and then with real data.

Next we tested the algorithm on some vehicle models.



(a) (b)

Figure 7.2: *(a) General view of Merriman Avenue and (b) Computed background using histogram approach- sequence 02*

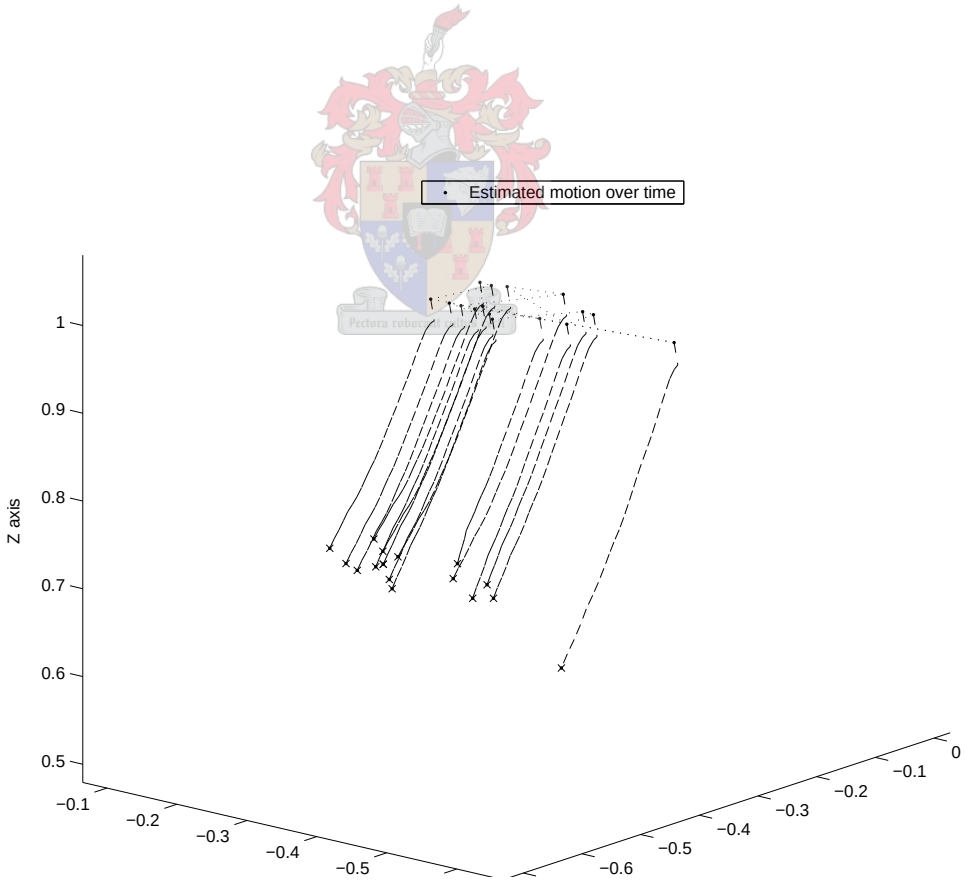
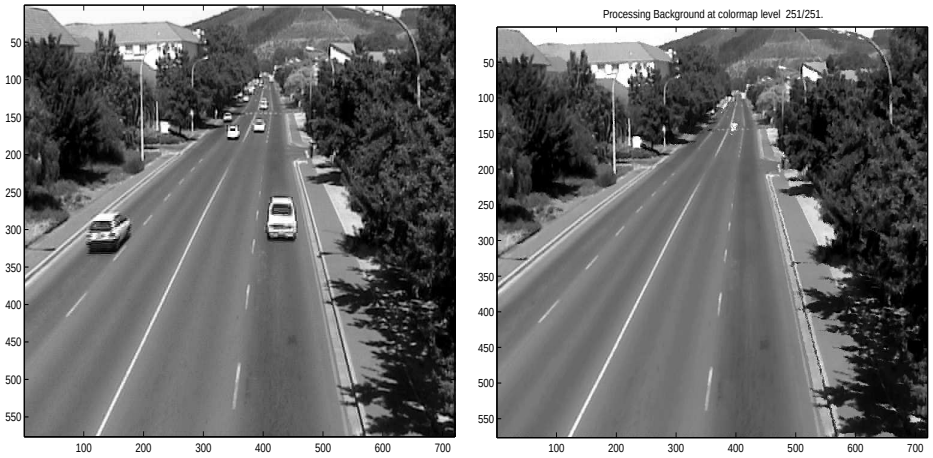
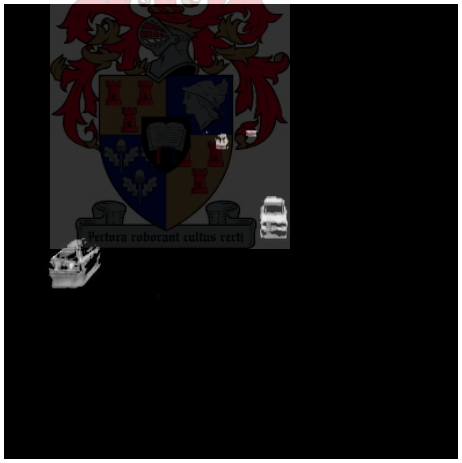


Figure 7.3: *Reconstruction of observation - Merriman Avenue - sequence 02*

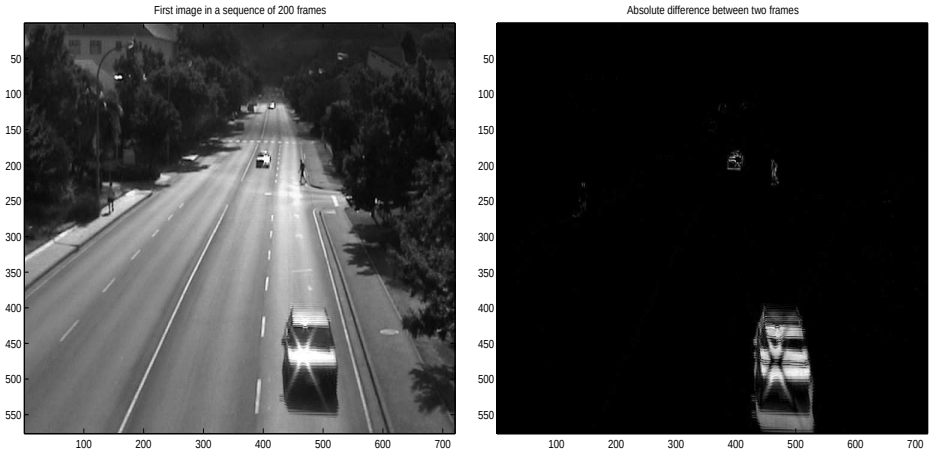


(a) (b)

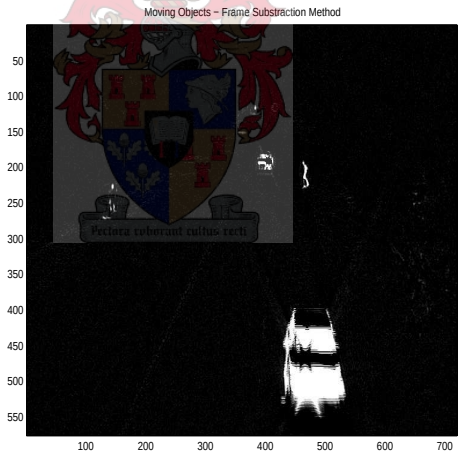


(c)

Figure 7.4: *(a) General view of Merriman Avenue and (b) Computed background and (c) Moving objects - sequence 03*

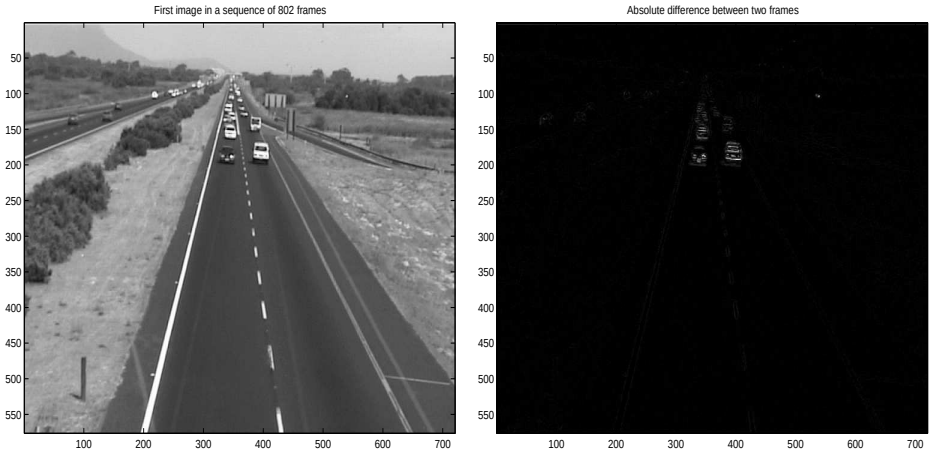


(a) (b)



(c)

Figure 7.5: *(a) General view of Merriman Avenue, (b) Moving objects by histogram and (c) Moving objects by frame difference - sequence 04*

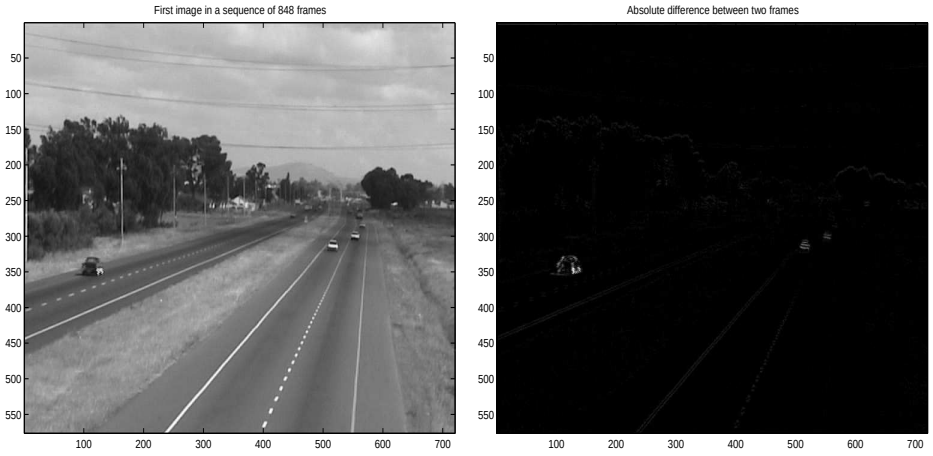


(a) (b)

Figure 7.6: *(a) General view of N2 and (b) Moving objects - sequence 04*

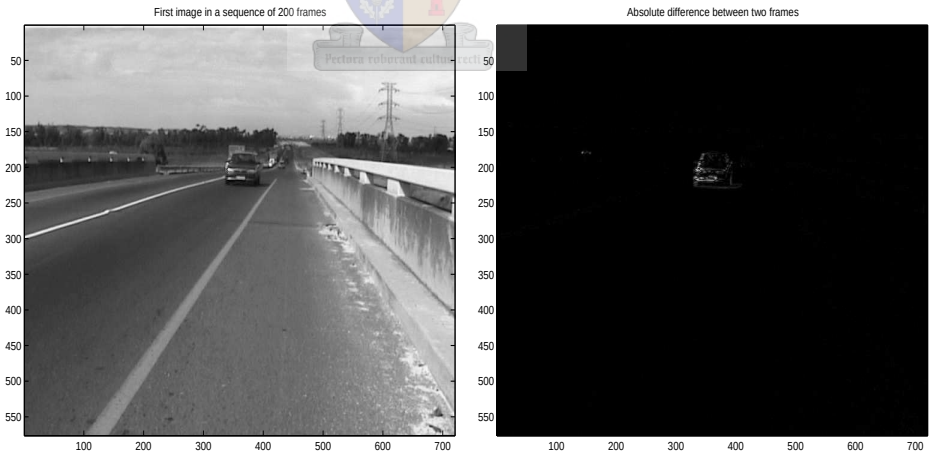
(a) (b)

Figure 7.7: *(a) General view of R102 and (b) Moving objects - sequence 01*



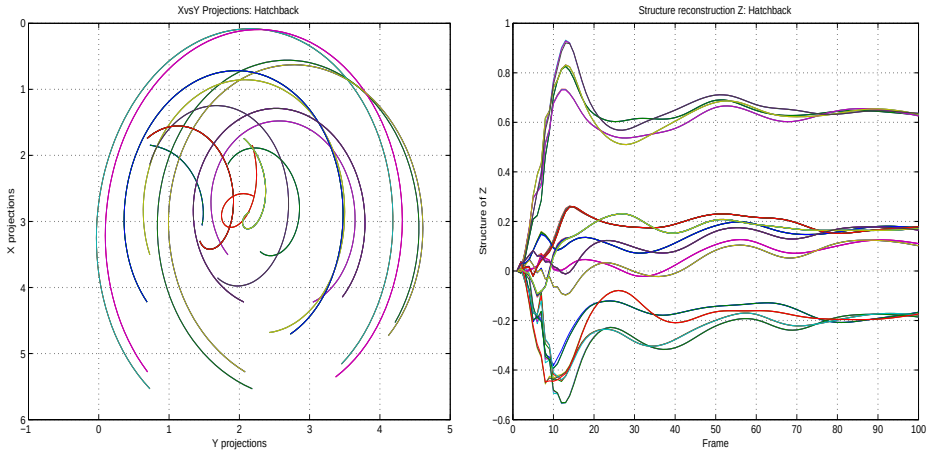
(a) (b)

Figure 7.8: *(a) General view of R102 and (b) Moving objects - sequence 02*

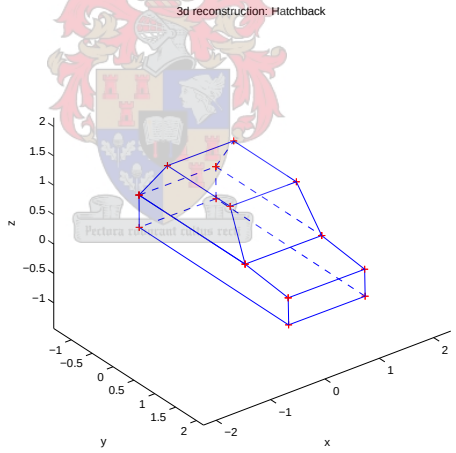


(a) (b)

Figure 7.9: *(a) General view of R310 and (b) Moving objects - sequence 01*

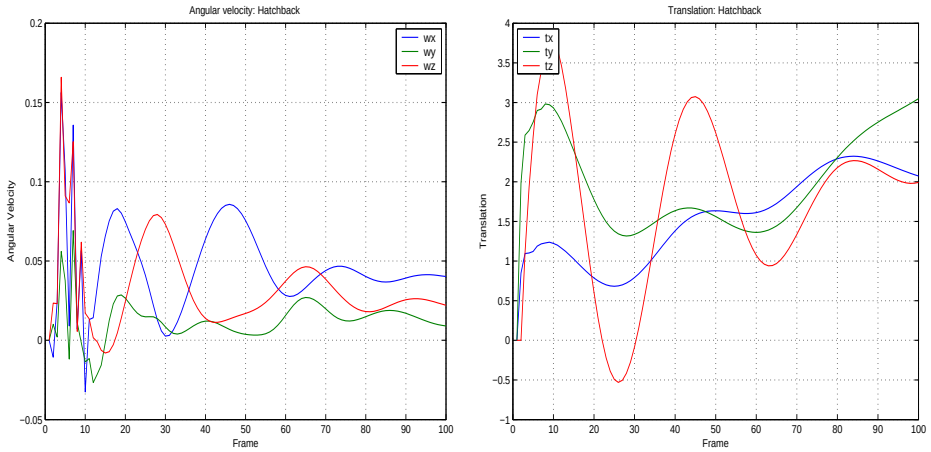


(a) (b)

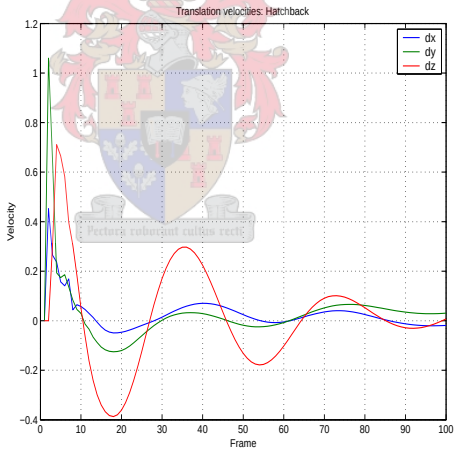


(c)

Figure 7.10: *Hatchback* ($4.4 \times 1.6 \times 1.4m^3$) *without noise; (a) Projections over time, (b) z-component and (c)* 3D *reconstruction*

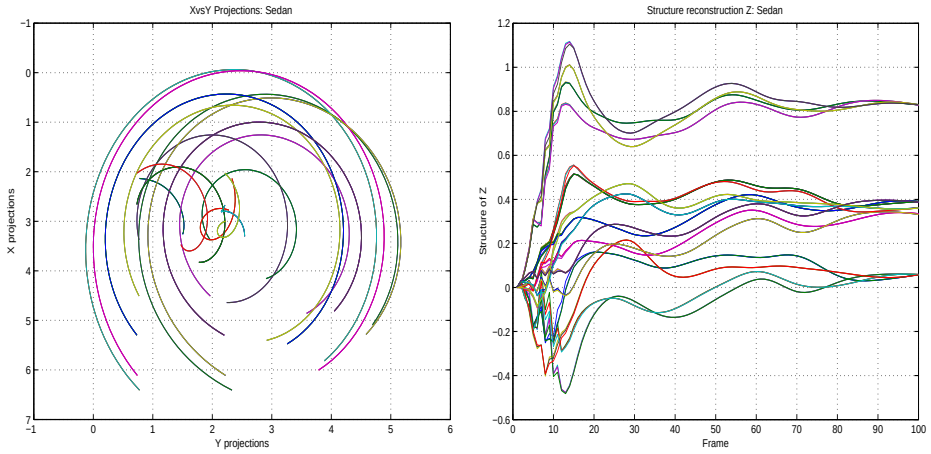


(a) (b)

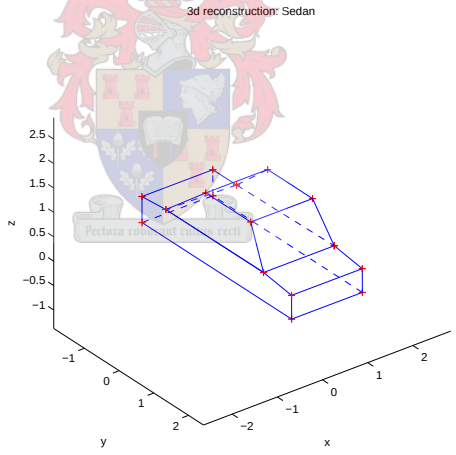


(c)

Figure 7.11: *Hatchback* ($4.4 \times 1.6 \times 1.4m^3$) *without noise; (a) Angular velocity, (b) Translation and (c) Linear velocity*



(a) (b)



(c)

Figure 7.12: *Sedan* ($4.4 \times 1.6 \times 1.4m^3$) *without noise; (a) Projections over time, (b) z-component and (c)* 3D *reconstruction*

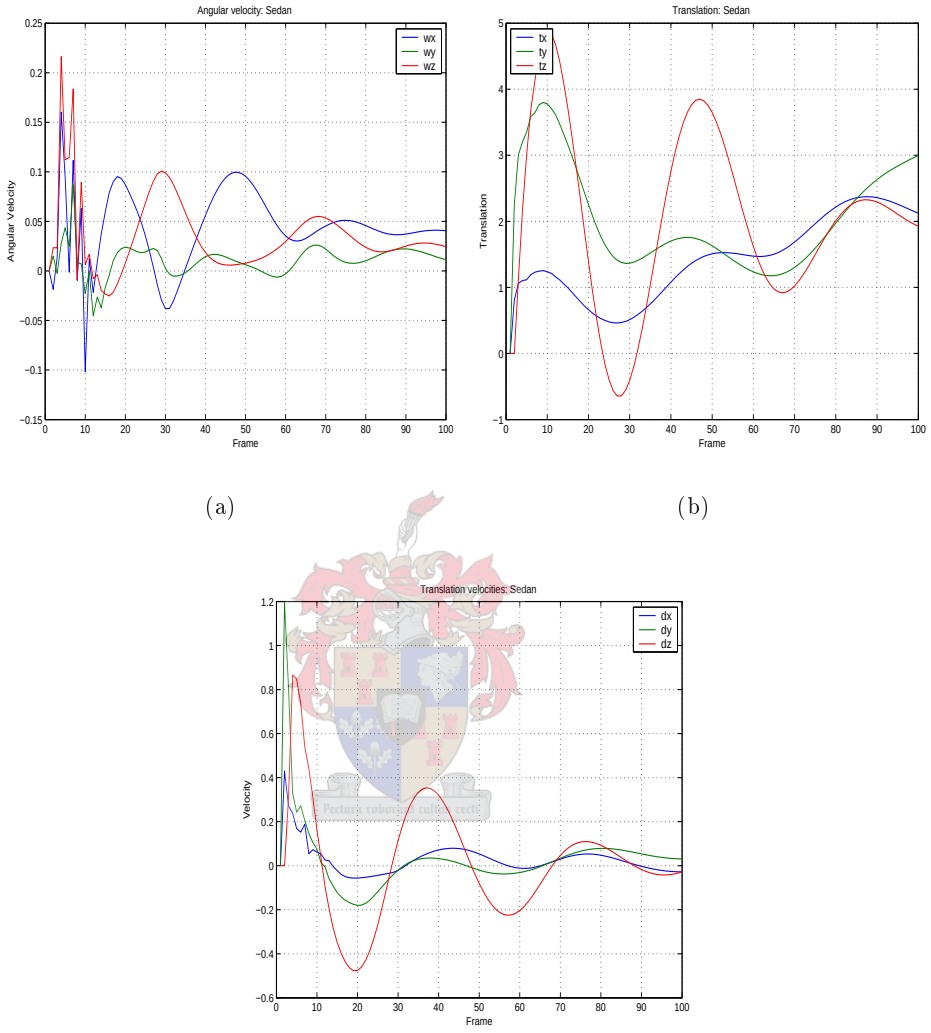
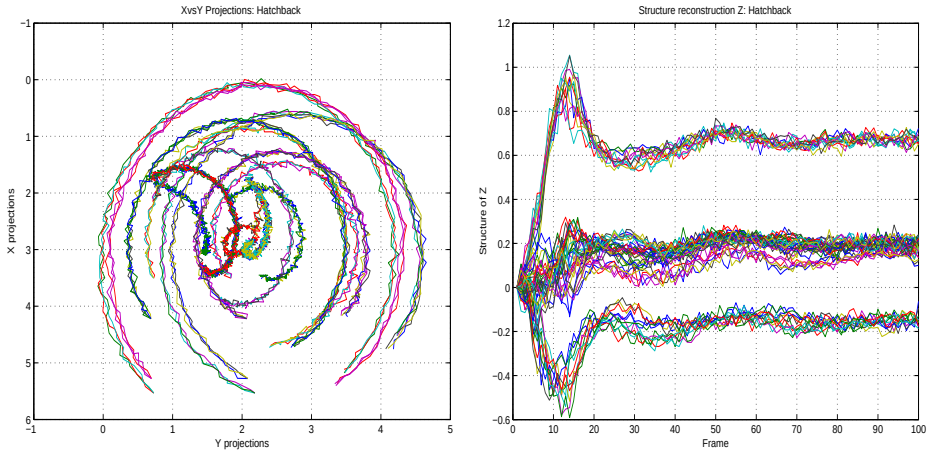
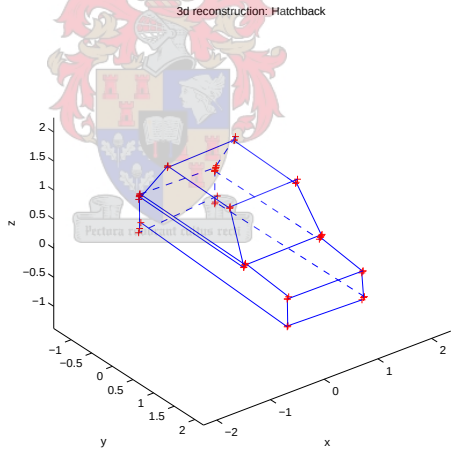


Figure 7.13: *Sedan ($4.4 \times 1.6 \times 1.4m^3$) without noise; (a) Angular velocity, (b) Translation and (c) Linear velocity*

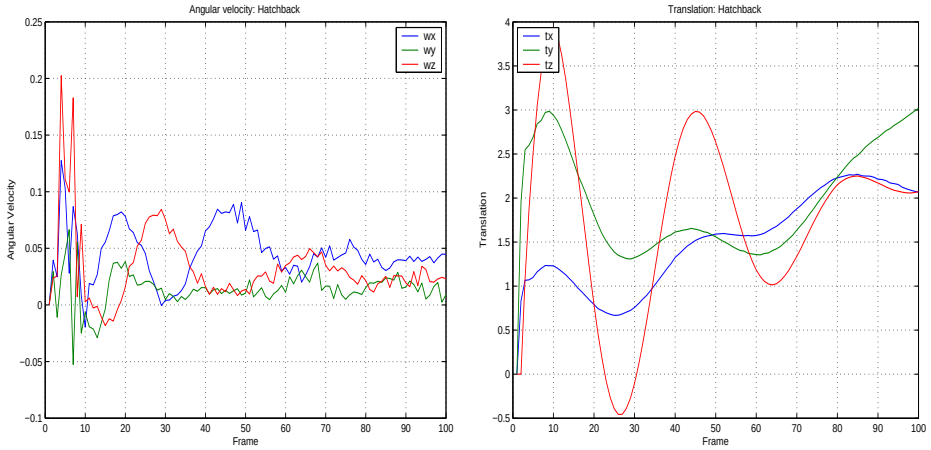


(a) (b)

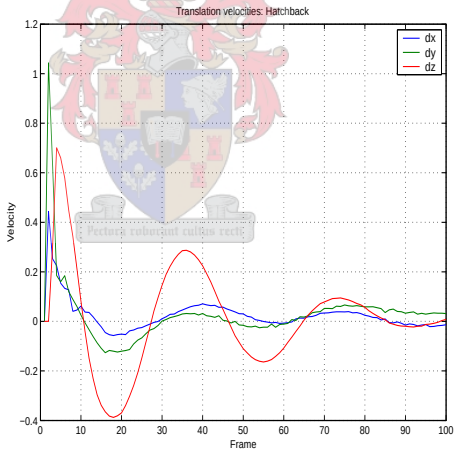


(c)

Figure 7.14: *Hatchback* ($4.4 \times 1.6 \times 1.4m^3$) *with noise of 0.001; (a) Projections over time, (b) z-component and (c)* 3D *reconstruction*

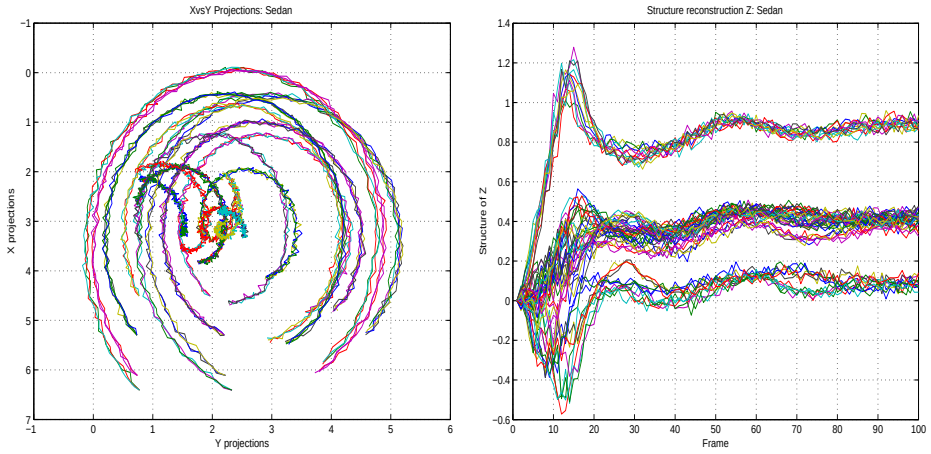


(a) (b)

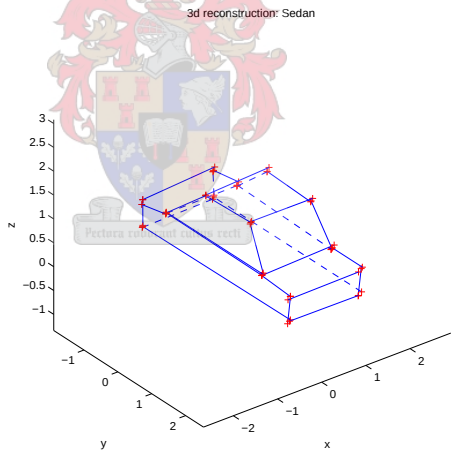


(c)

Figure 7.15: *Hatchback* ($4.4 \times 1.6 \times 1.4m^3$) *with noise of 0.001; (a) Angular velocity, (b) Translation and c: Linear velocity*



(a) (b)



(c)

Figure 7.16: *Sedan* ($4.4 \times 1.6 \times 1.4m^3$) *with noise of 0.001; (a) Projections over time, (b) z-component and (c)* 3D *reconstruction*

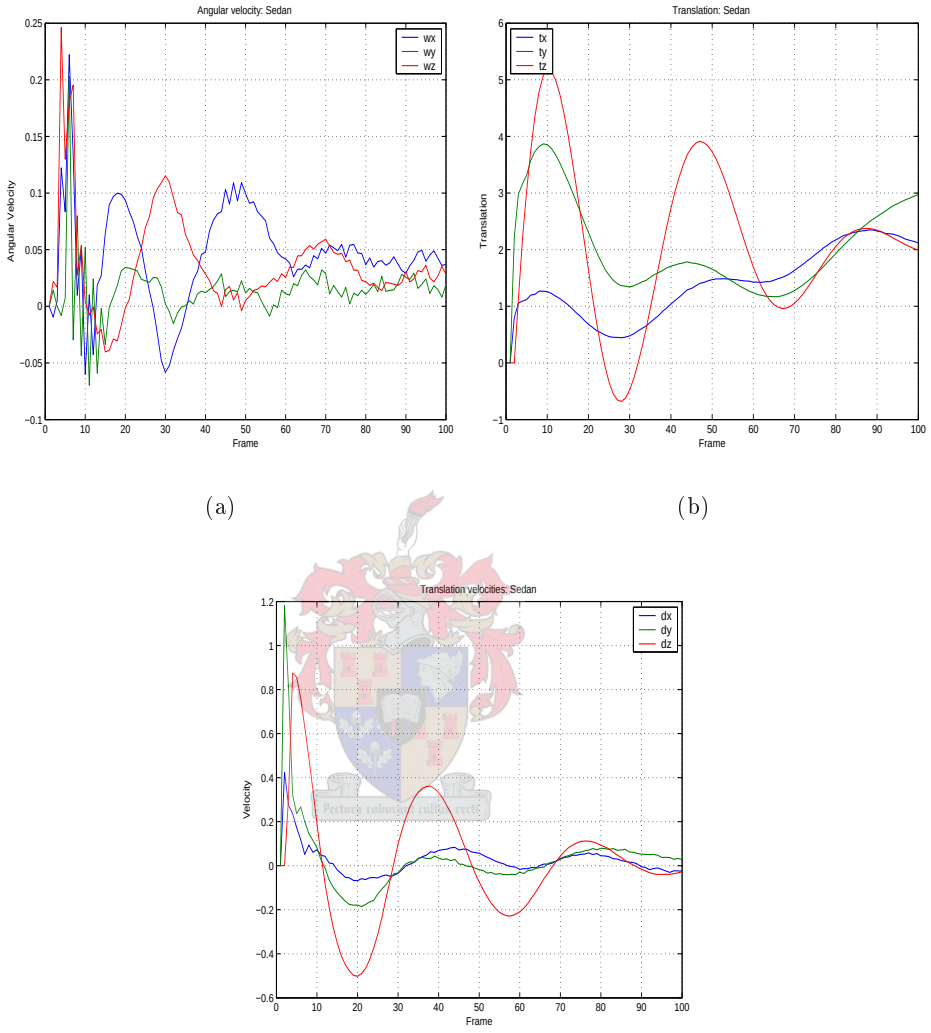
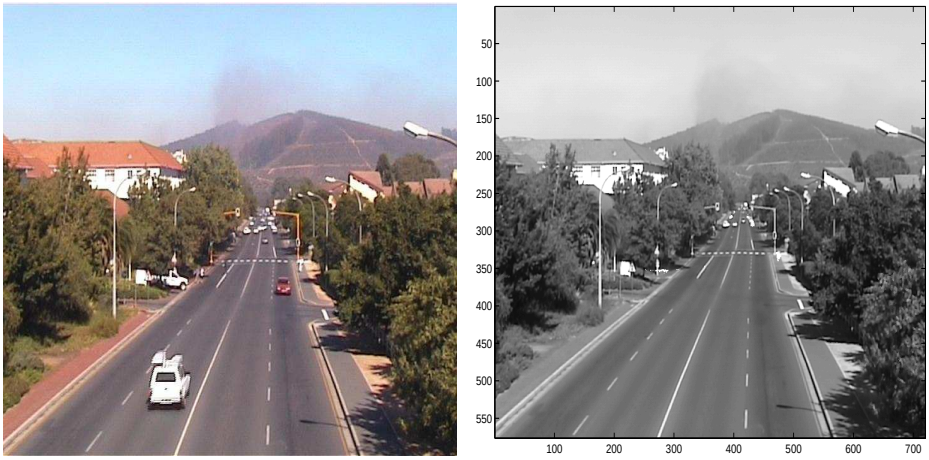
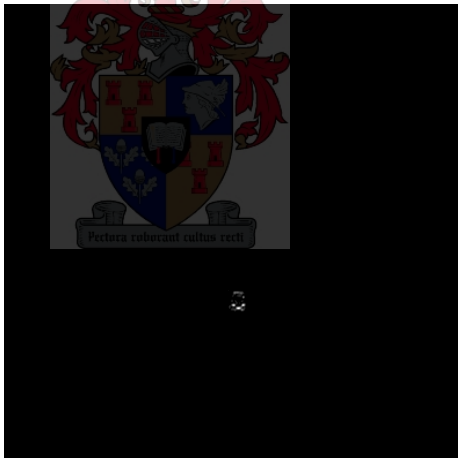


Figure 7.17: *Sedan ($4.4 \times 1.6 \times 1.4m^3$) with noise of 0.001; (a) Angular velocity, (b) Translation and (c) Linear velocity*

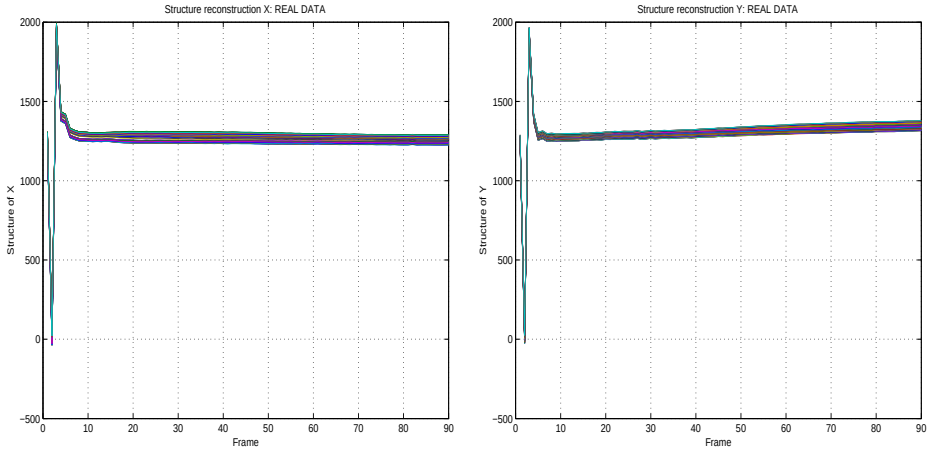


(a) (b)

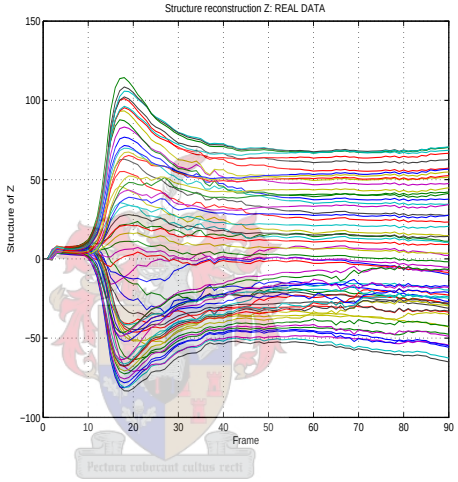


(c)

Figure 7.18: *Experiments using real data set 1; (a) Original image, (b) Background and (c) Moving (red car)*



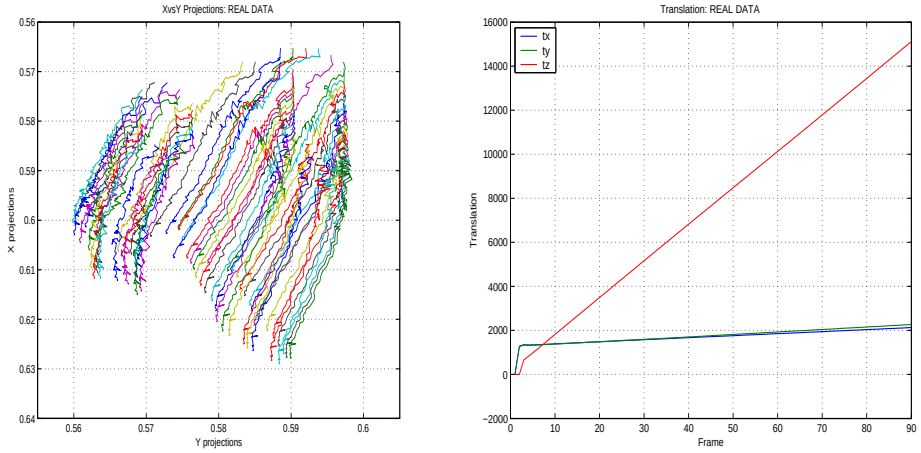
(a) (b)



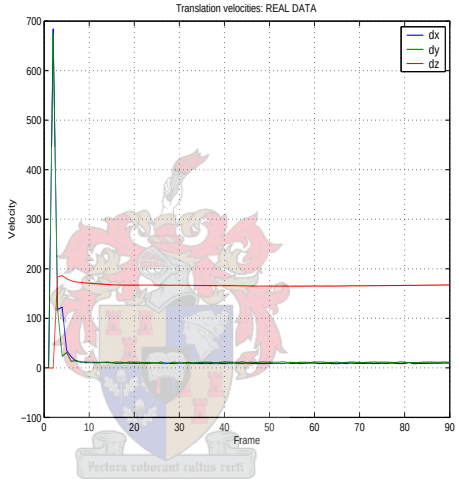
(c)

Figure 7.19: 3D *reconstruction using real data set 1; (a) x-component, (b) y-component and (c) z-component*

In Figure 7.19 we have processed real data (calibration car shown in Figure 7.18c). We observe that good results are obtained just before the eighteenth frame. This also almost corresponds to the result we got using synthetic data (car model in Figures 7.10 and 7.16). Now the question is whether we can speed up the convergence of *Unscented Kalman Filter*.



(a) (b)



(c)

Figure 7.20: 3D *reconstruction using real data set 1; (a) Features trajectory, (b) Translation and (c) Linear velocity*

7.3.1 UKF Re-initialization

In this subsection we investigate whether re-initializing[1] *UKF* could improve the convergence rate. The idea behind the re-initialization is to accelerate the convergence of *Unscented Kalman Filter* by using pre-processed data. In Figure 7.23(c) we see that *Unscented Kalman Filter* take some frames to begin take desirable structure.

[1] Re-initialization is a process in which we run *Unscented Kalman Filter* some frames, i.e. 10 frames. Next we use the computed parameters as new input in *Unscented Kalman Filter*. The goal is to reduce the number of frames to be used for the entire process.

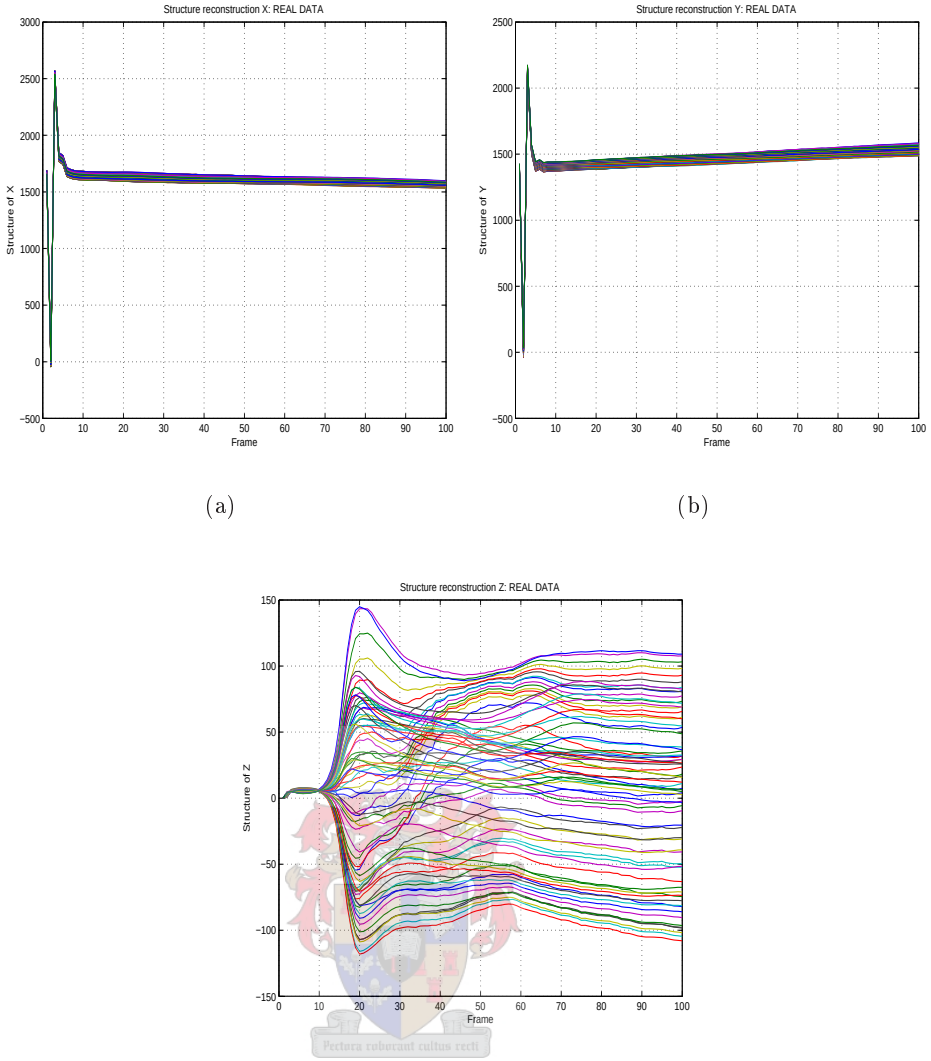
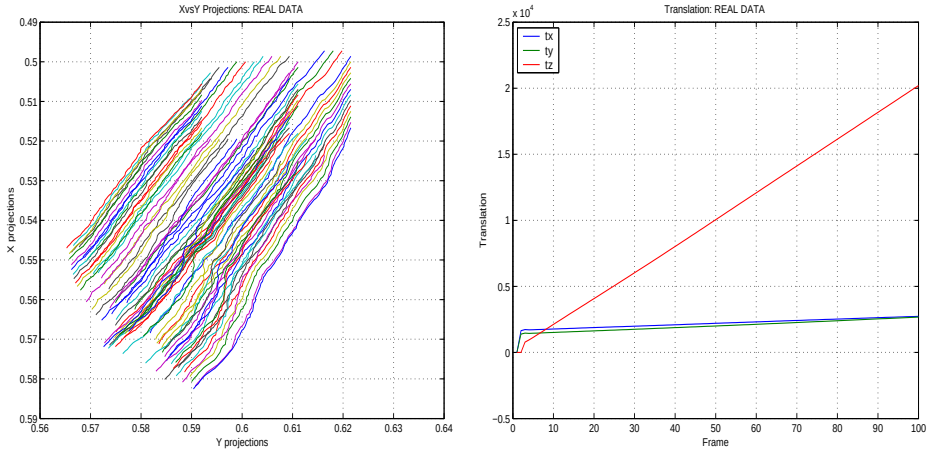


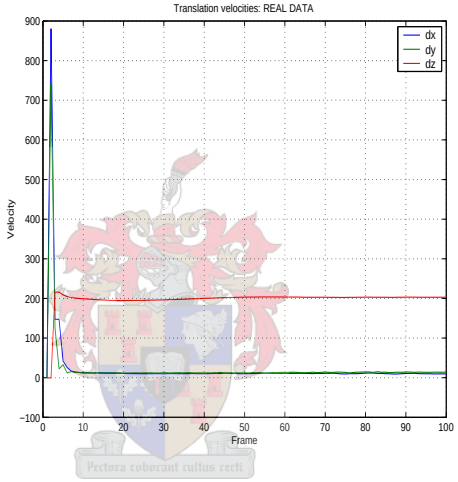
Figure 7.21: 3D *reconstruction using real data set 2; (a) x-component, (b) y-component and (c) z-component*

The figure observations suggest that *UKF* converges faster if we provide it with an initial guess different from a flat object. In order to confirm this we decide to perform several tests depicted in Figure 7.25 through Figure 7.32.

In Figure 7.32 the algorithm was re-initialized with 30 frames. We can see clearly that the system will converge after fifty frames. Again the total number of frames required to converge will be about eighty frames. From Figure 7.25 through Figure 7.32 we see that, in fact the convergence of *UKF* occurs after eighty frames. This suggests that the speed can only be computed after eighty frames. That means the question of accelerating the *Unscented Kalman Filter* is still open. One should find



(a) (b)



(c)

Figure 7.22: 3D *reconstruction using real data set 2; (a) Features trajectory, (b) Translation and (c) Linear velocity*

other ways of speeding up the process. Next we investigate the relationship between the convergence of *UKF* and the *camera frame rate*.

7.3.2 UKF and Frame Rate

In this sub-section we investigate whether the camera rate has some influence on the convergence of the *Unscented Kalman Filter*. Our goal is to use fewer frames for *UKF* to converge. If it is true, then a camera with higher or lower frame rate should probably be employed. Since we only have real data taken by one camera there is no way we can test the higher rate, although on real data lower frame rate

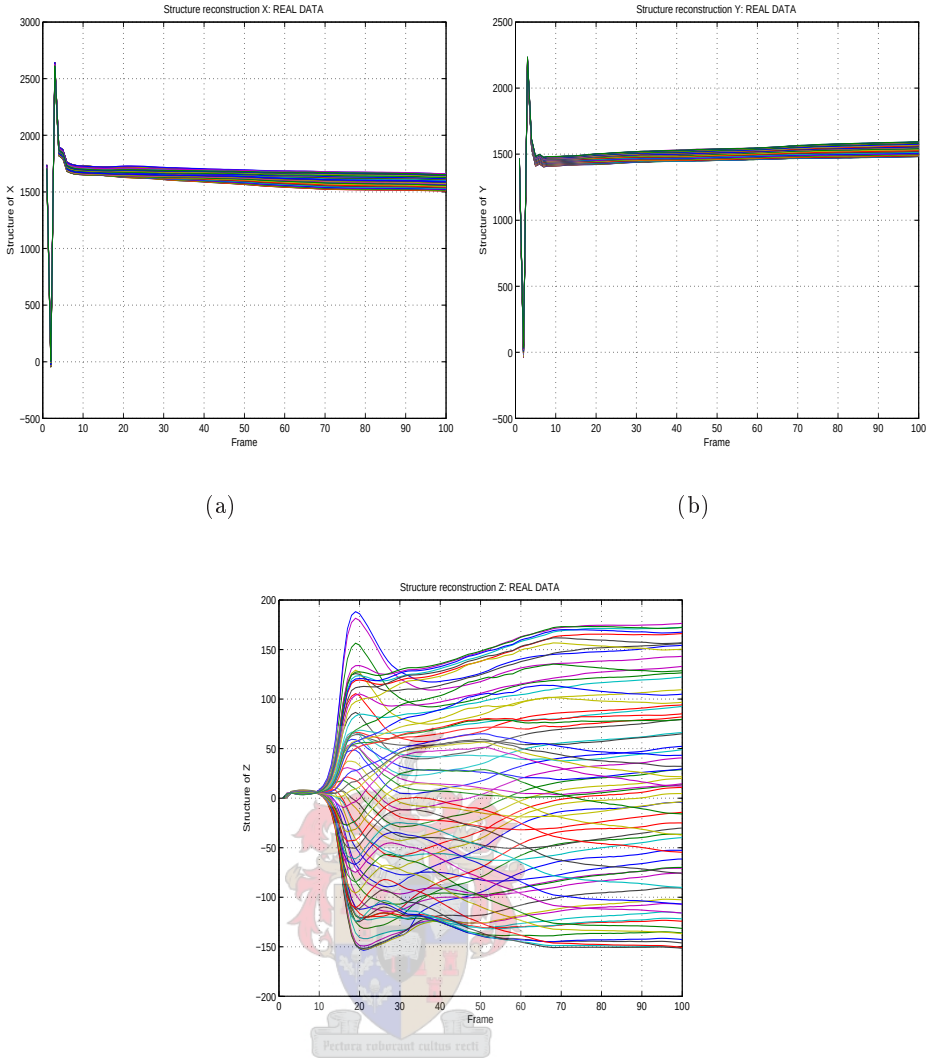
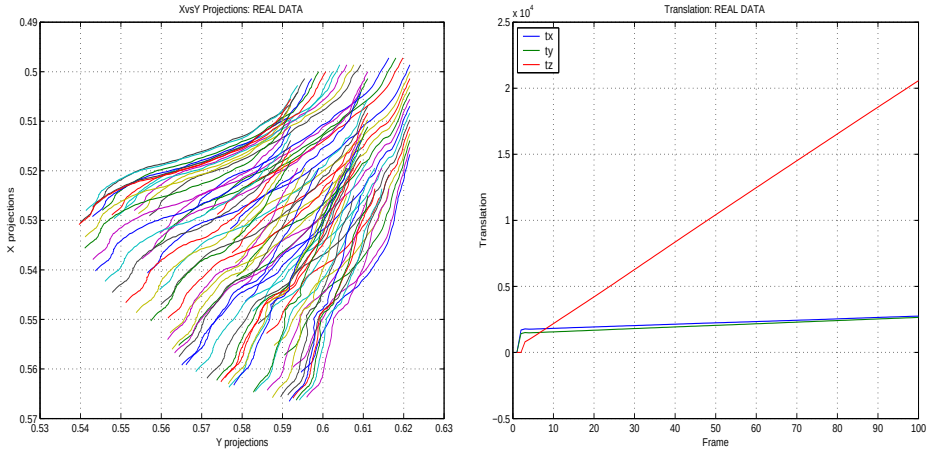


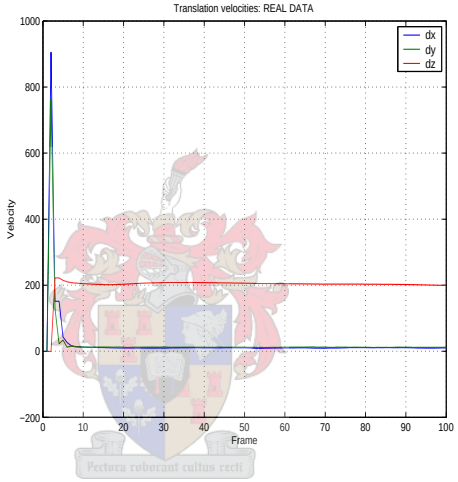
Figure 7.23: 3D *reconstruction using real data set 3; (a) x-component, (b) y-component and (c) z-component*

can easily be modeled. On the other hand, we can explore the fact that for synthetic data we can generate as many frames as we want. Thus we can simulate any kind of video camera by changing the rotation and translation. First, we ran a *UKF* for two hundred frames. And then we simulated the frame rate to a half[2]. The results show that the convergence of *UKF* will happen somewhere after a hundred frames (see Figures 7.33 and 7.36). Figure 7.37 clearly shows that if we leave *UKF* running for a very long time it may diverge or we will not recognize the object.

[2]Simulation of a camera with lower frame rate.



(a) (b)



(c)

Figure 7.24: 3D *reconstruction using real data set 3; (a) Features trajectory, (b) Translation and (c) Linear velocity*

Next we simulated the fast camera rate using synthetic data. In order to achieve this goal, we made the rotation and translation two times slower over two hundred frames. This corresponds to using a video camera two times faster than the previous one. The maximum data size is 0.05, and we will also check the influence of noise.

We checked the *UKF* convergence using real data. Some of the procedures we applied in the case of purely synthetic data cannot be applied to the real data due to the length of the image sequence. Thus we can only simulate a lower frame rate camera. The first difficulty is that we do not have a very long sequence to test this

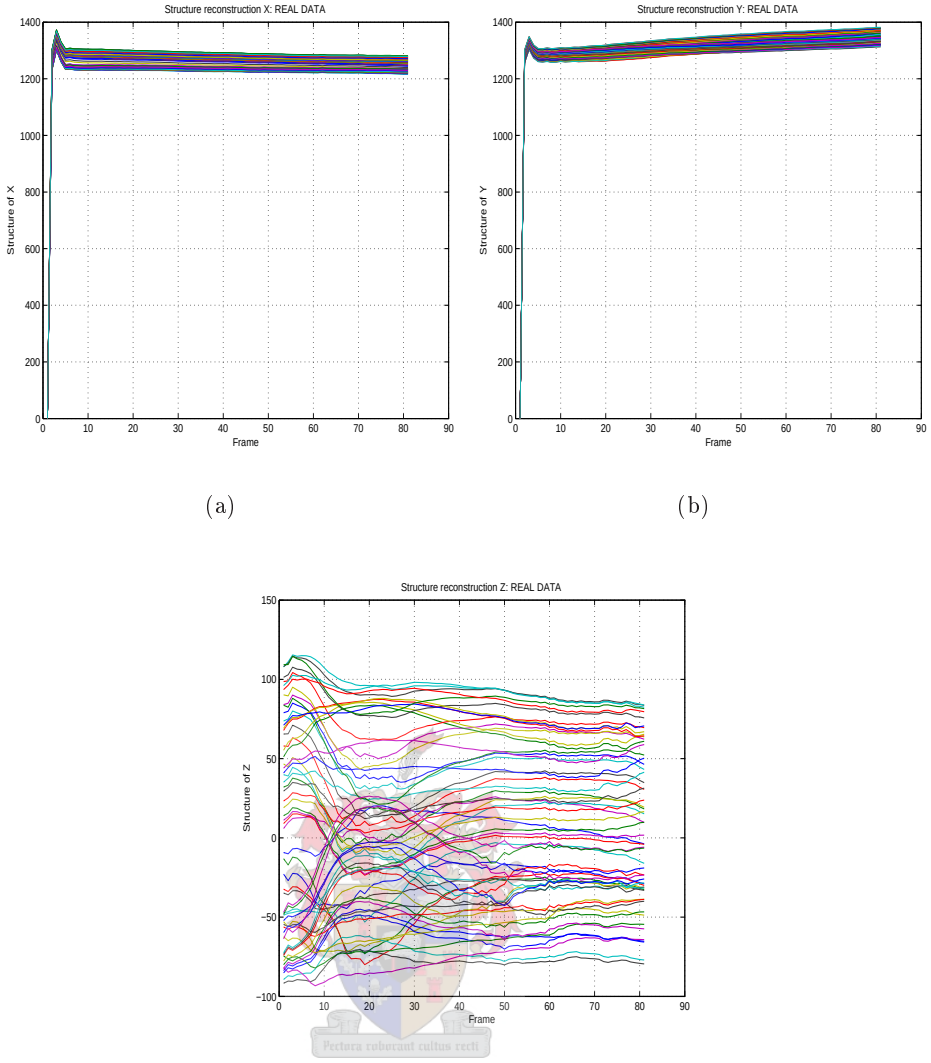
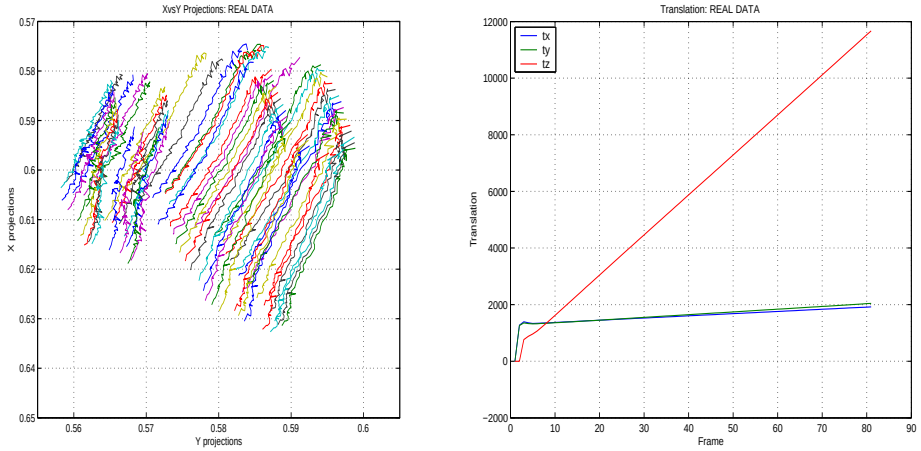
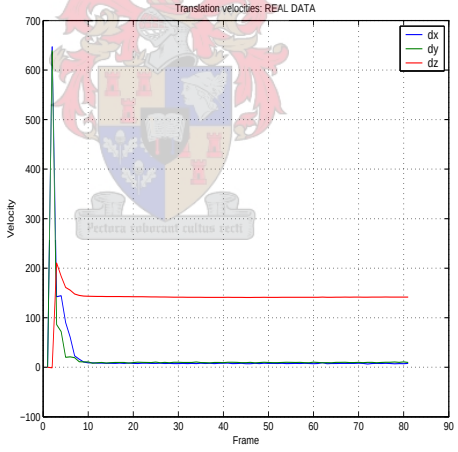


Figure 7.25: *Re-initialization of UKF using 20 frames - real data set 1; (a) x-component, (b) y-component and (c) z-component*

frame rate. First, for frame rate equal to 2 this means we need a video sequence of at least two hundred frames. From previous results we also know that if we want to test the frame rate greater or equal to three we need more than 380 frames. This suggests that because there is no way an object will remain in the sequence over that a long time, consequently, we can not check the frame rate using real data.



(a) (b)



(c)

Figure 7.26: *Re-initialization of UKF using 20 frames - real data set 1; (a) Features trajectory, (b) Translation and (c) Linear velocity*

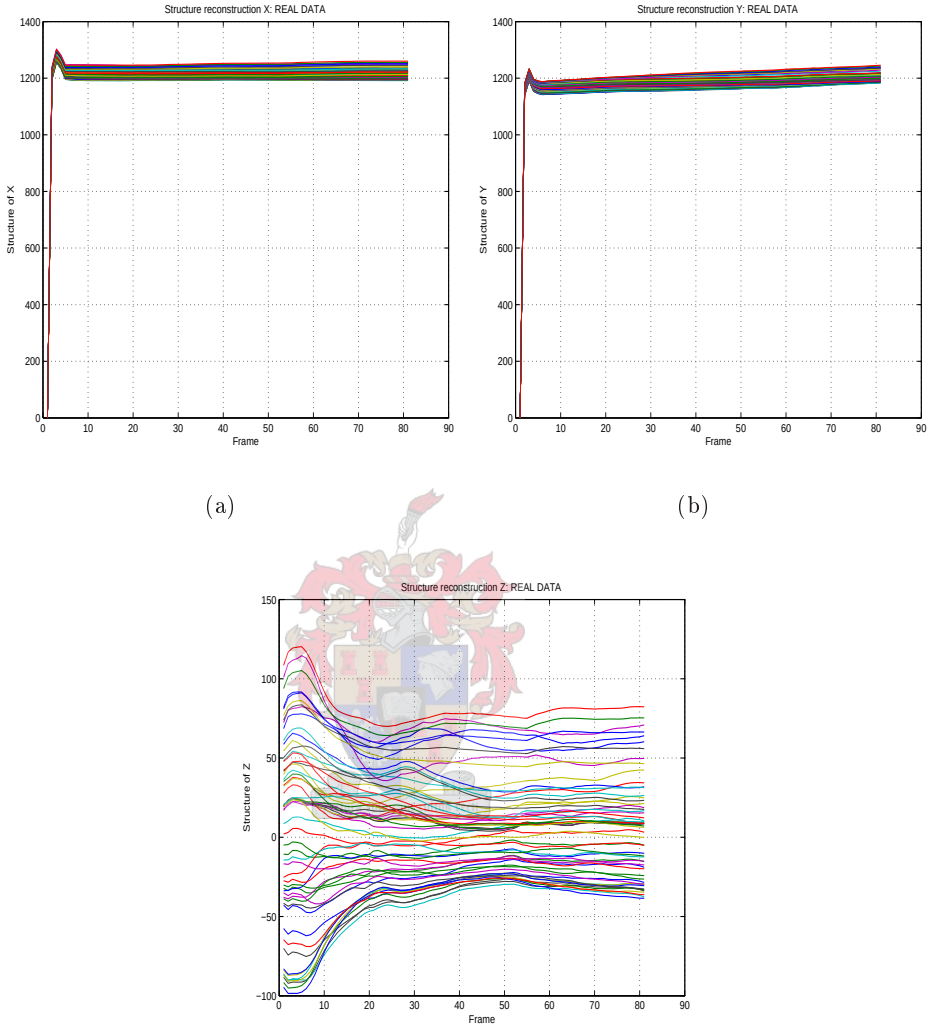


Figure 7.27: *Re-initialization of UKF using 20 frames - real data set 2; (a) x-component, (b) y-component and (c) z-component*

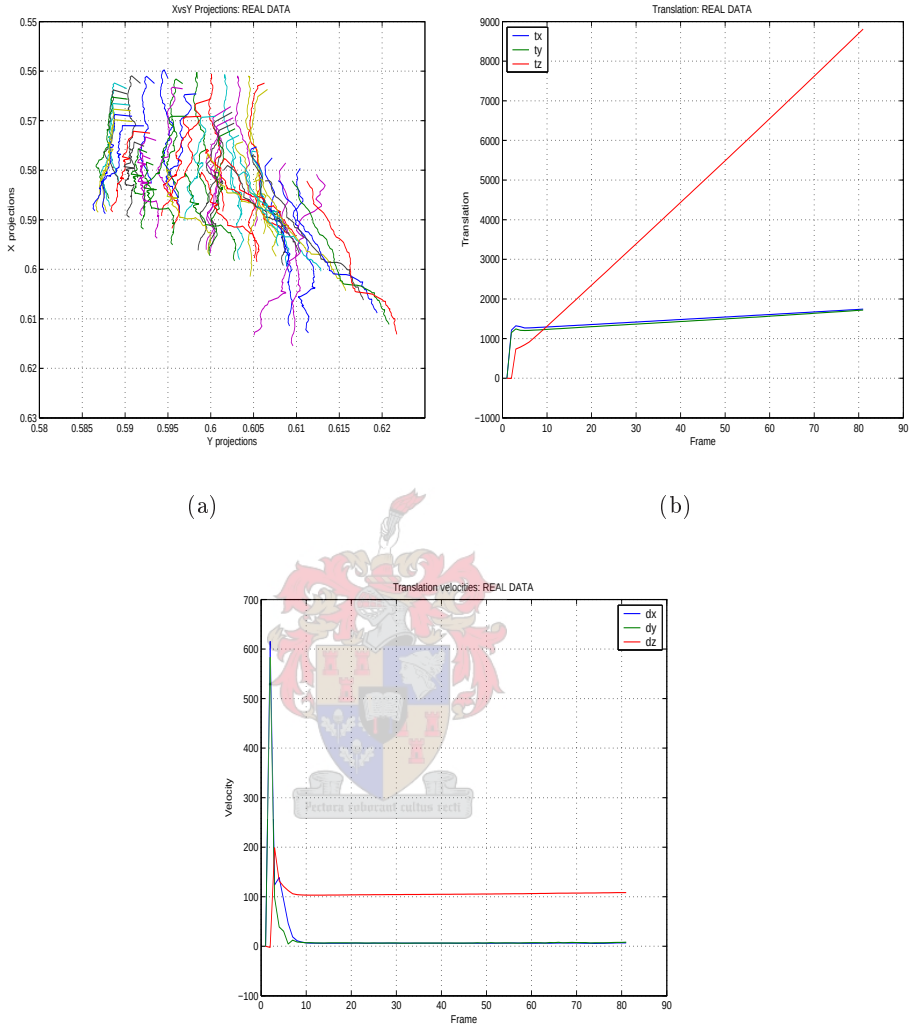


Figure 7.28: *Re-initialization of UKF using 20 frames - real data set 2; (a) Features trajectory, (b) Translation and (c) Linear velocity*

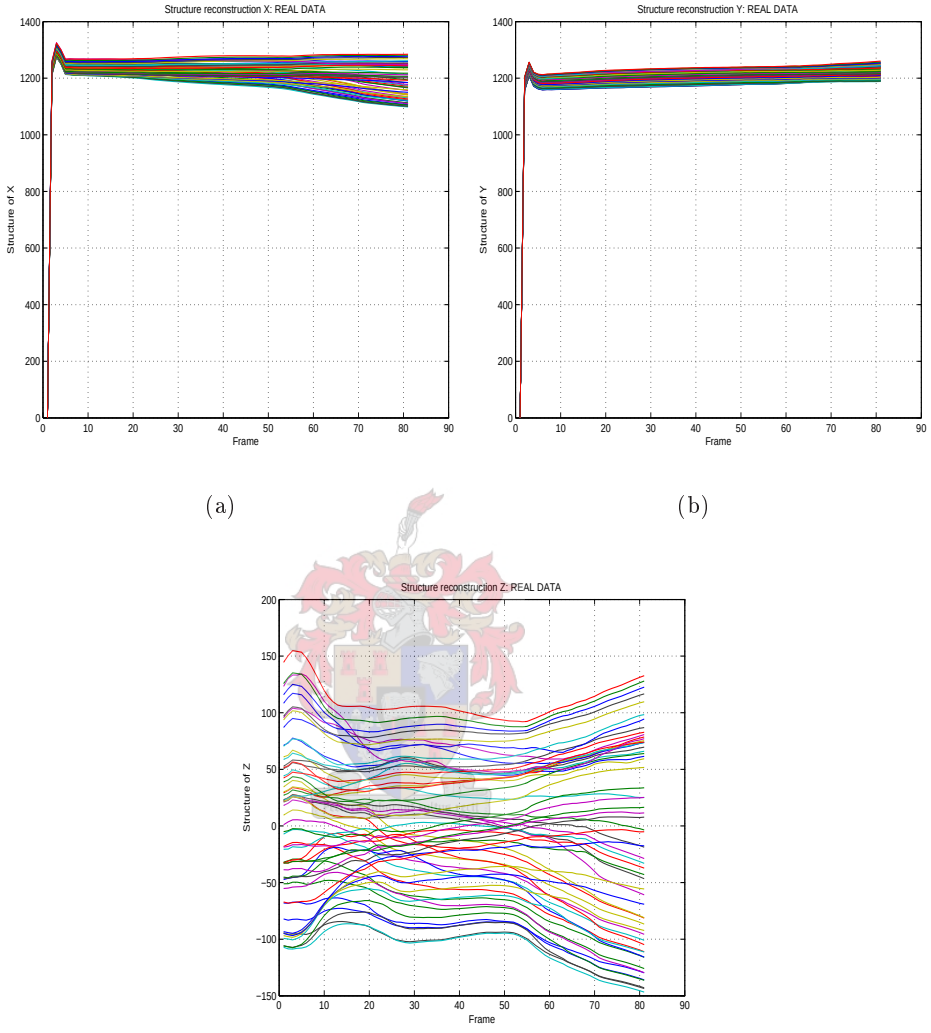
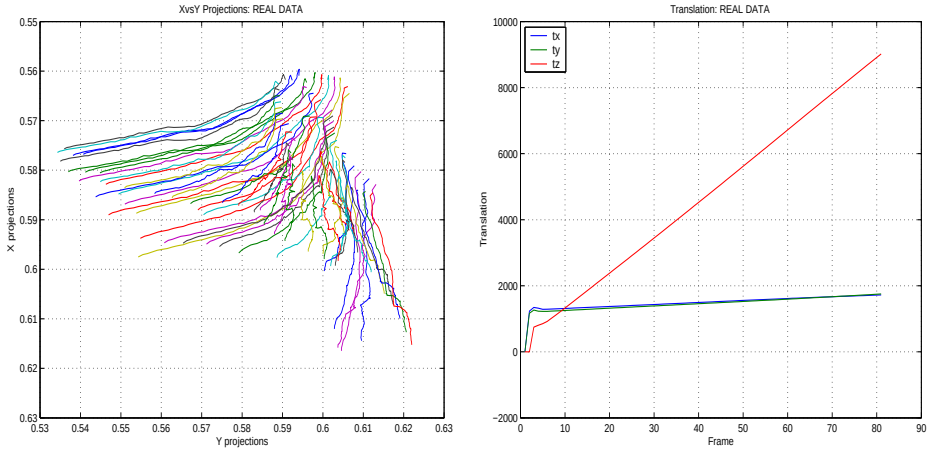
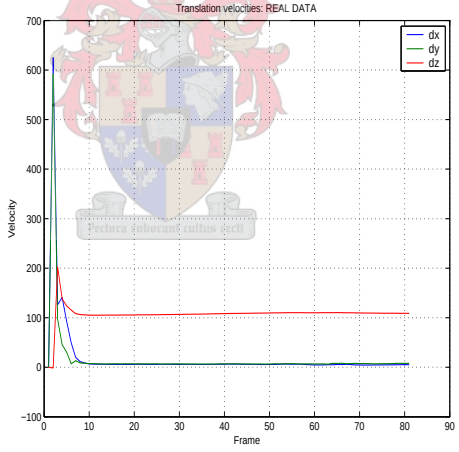


Figure 7.29: *Re-initialization of UKF using 20 frames - real data set 3; (a) x-component, (b) y-component and (c) z-component*



(a) (b)



(c)

Figure 7.30: *Re-initialization of UKF using 20 frames - real data set 3; (a) Features trajectory, (b) Translation and (c) Linear velocity*

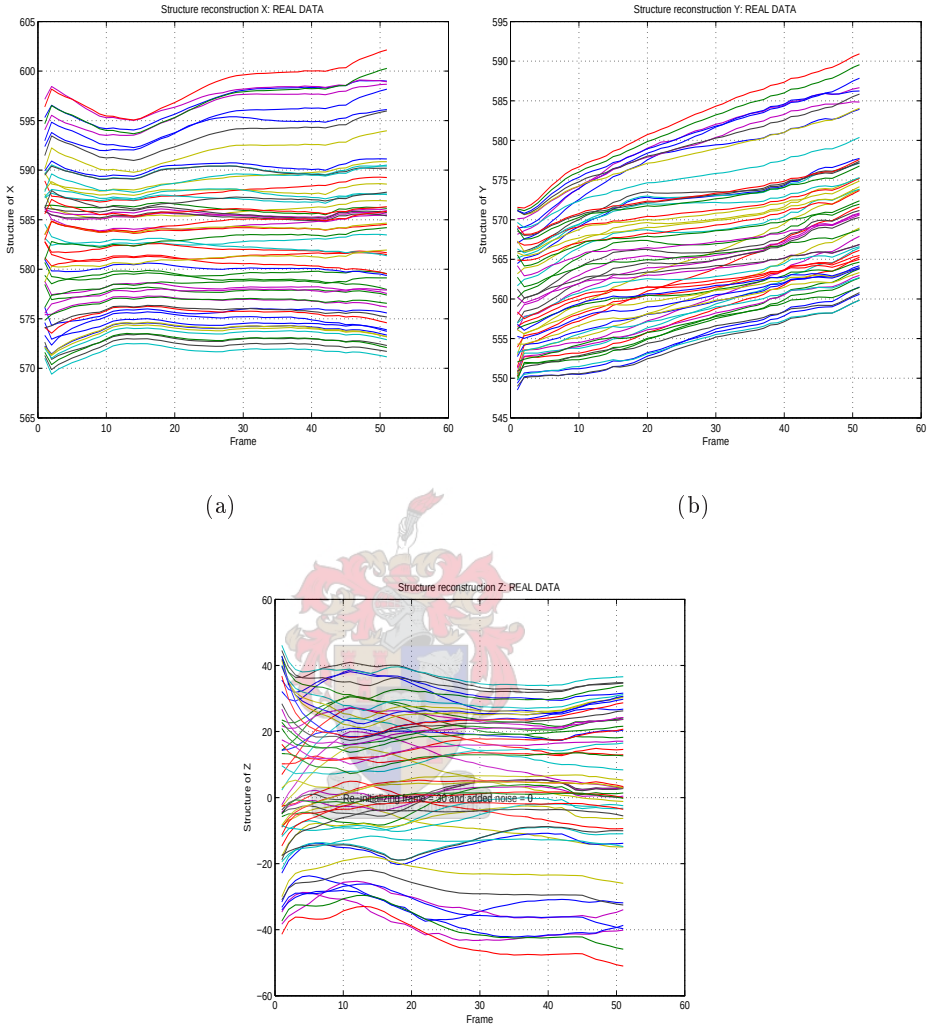


Figure 7.31: *Re-initialization of UKF using 30 frames - real data set 3; (a) x-component, (b) y-component and (c) z-component*

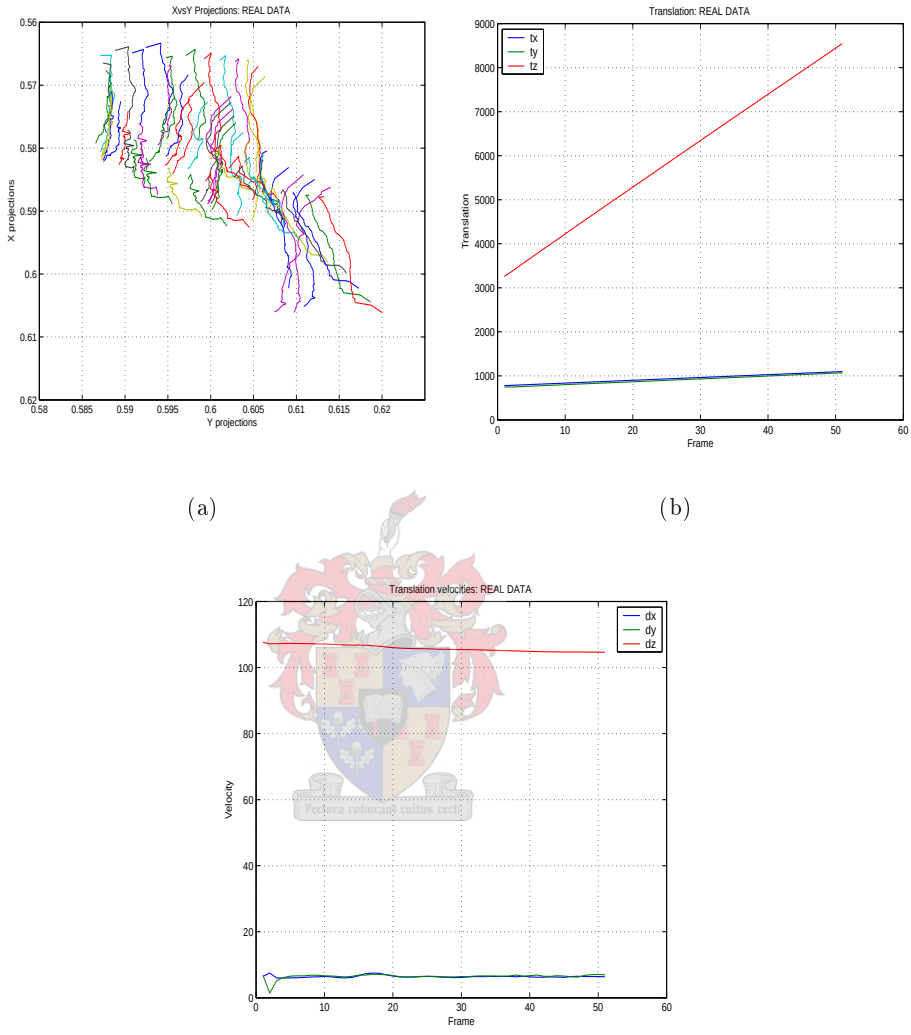
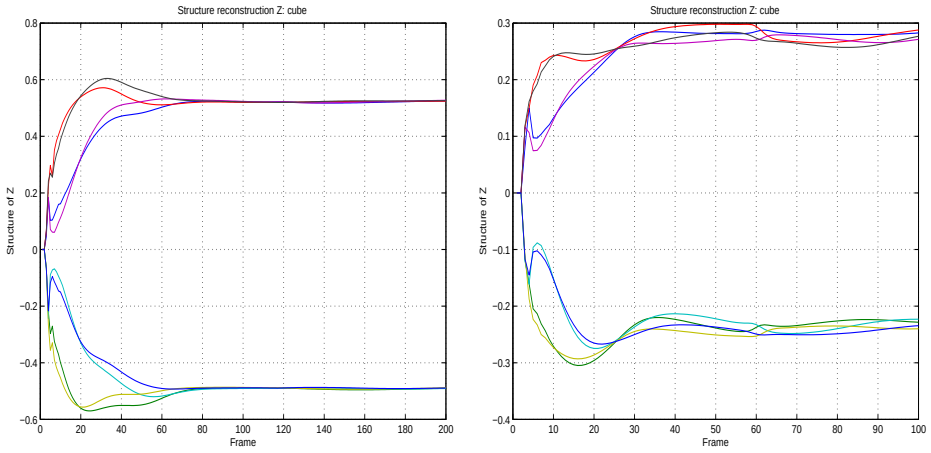
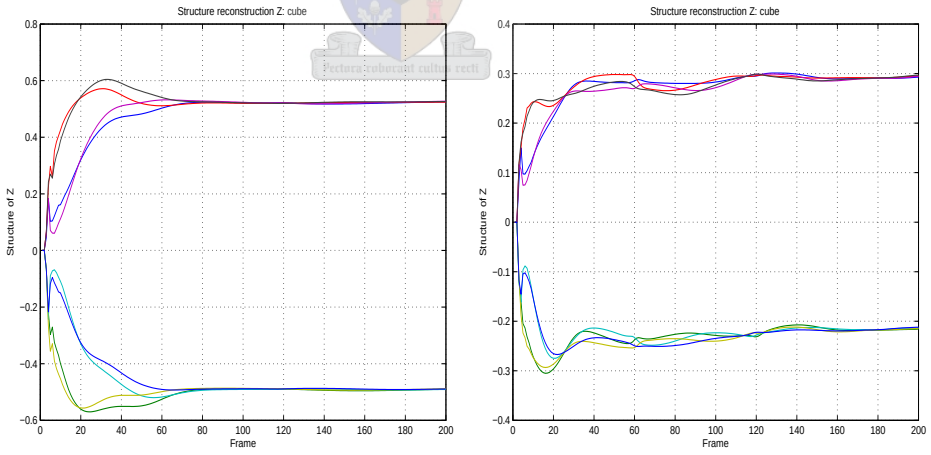


Figure 7.32: *Re-initialization of UKF using 30 frames - real data set 2; (a) Features trajectory, (b) Translation and (c) Linear velocity*



(a) (b)

Figure 7.33: *Simulation of UKF and camera frame rate; (a) Using original frame rate - convergence at 80th frame , (b) Setting the frame rate to a half - no convergence noticed at 100 frames*



(a) (b)

Figure 7.34: *Simulation of UKF and camera frame rate; (a) Using frame rate 1, (b) The frame rate 1/2 - convergence at 80 and 160 frames respectively*

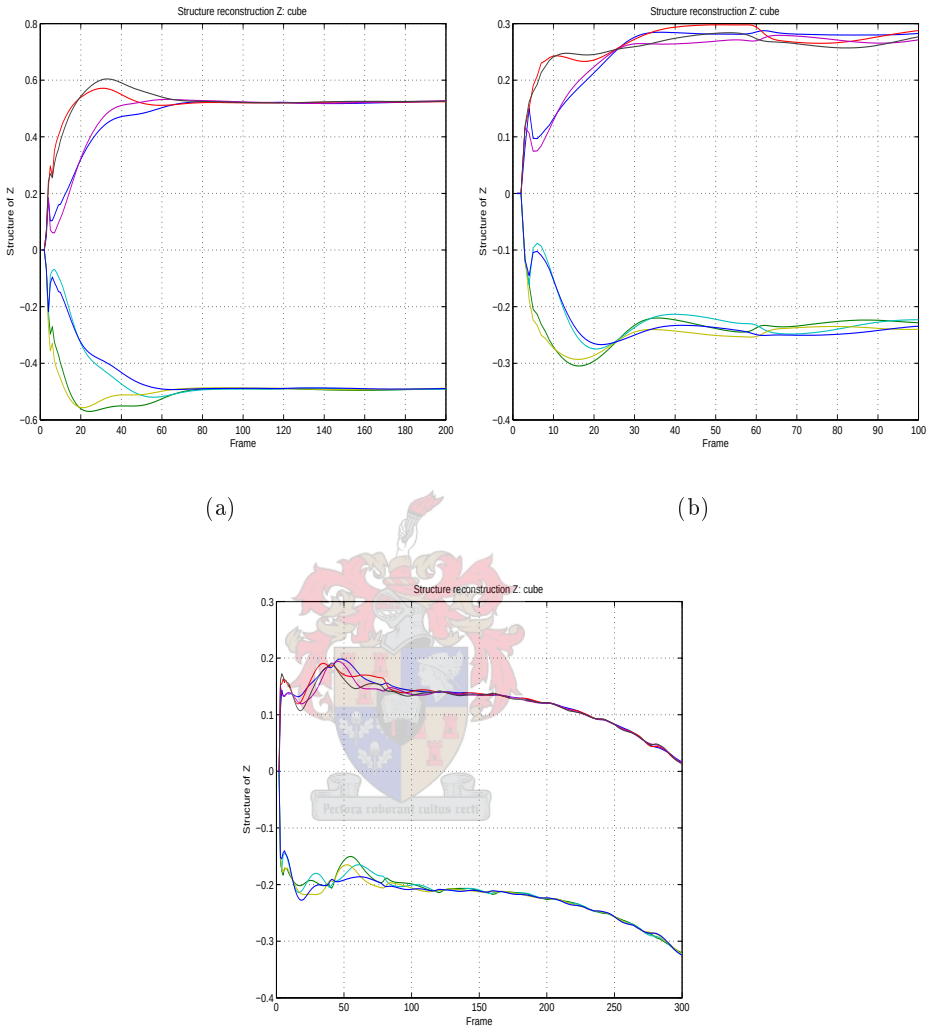


Figure 7.35: *Simulation of UKF and camera frame rate; (a) Using frame rate 1, (b) Frame rate 1/2 and (c) Frame rate 1/3 - convergence at 80, 160 and unknown frames respectively*

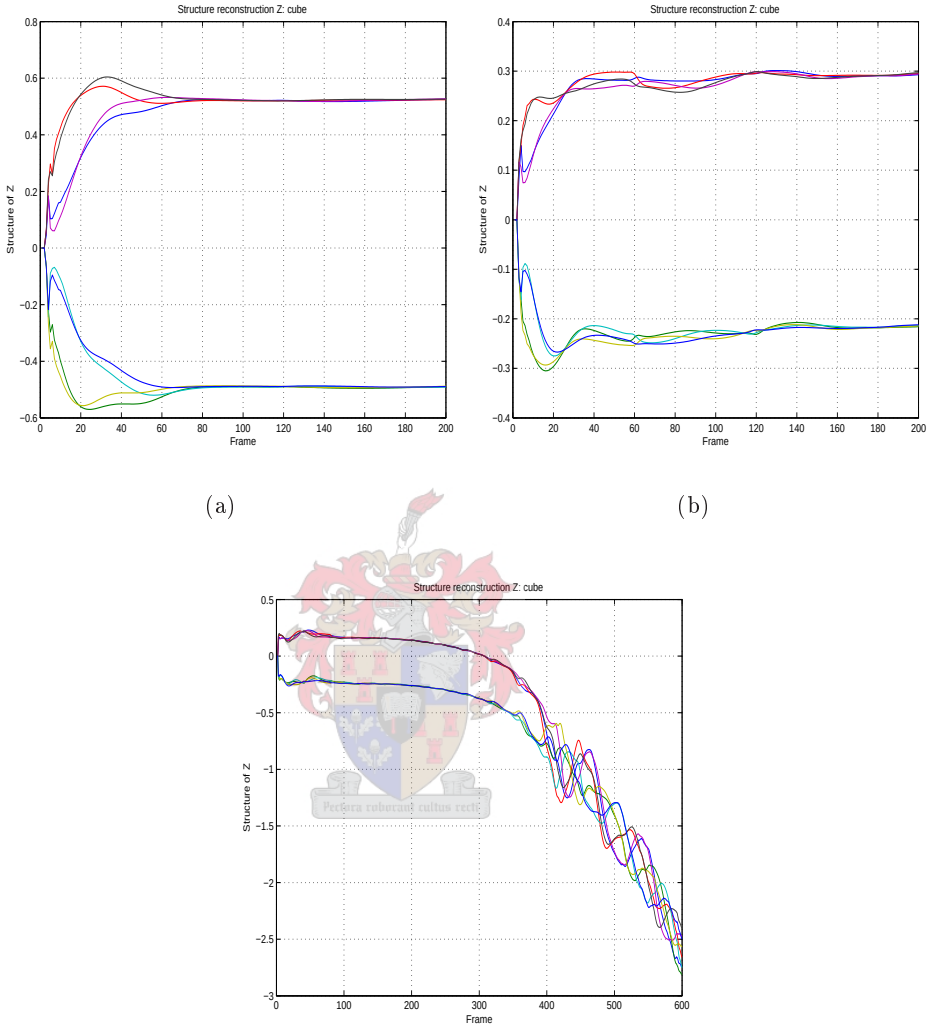
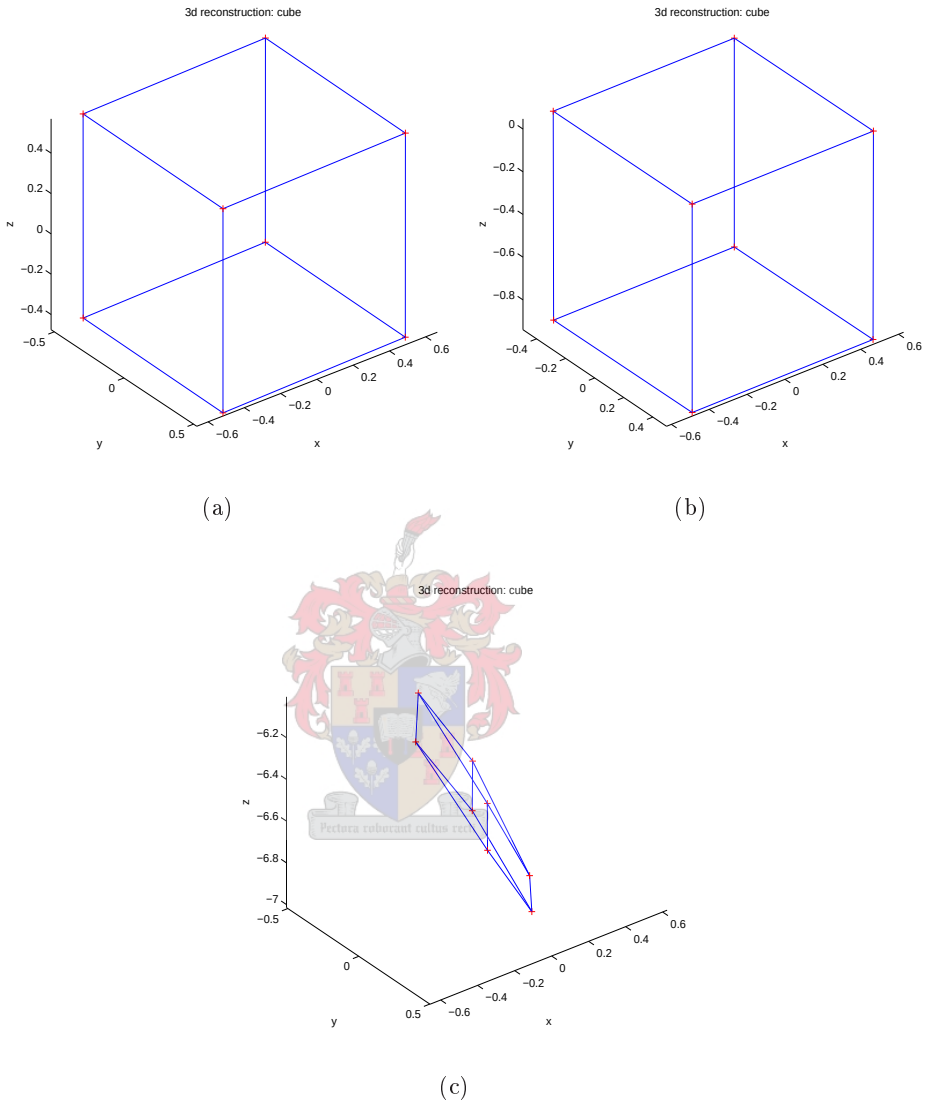


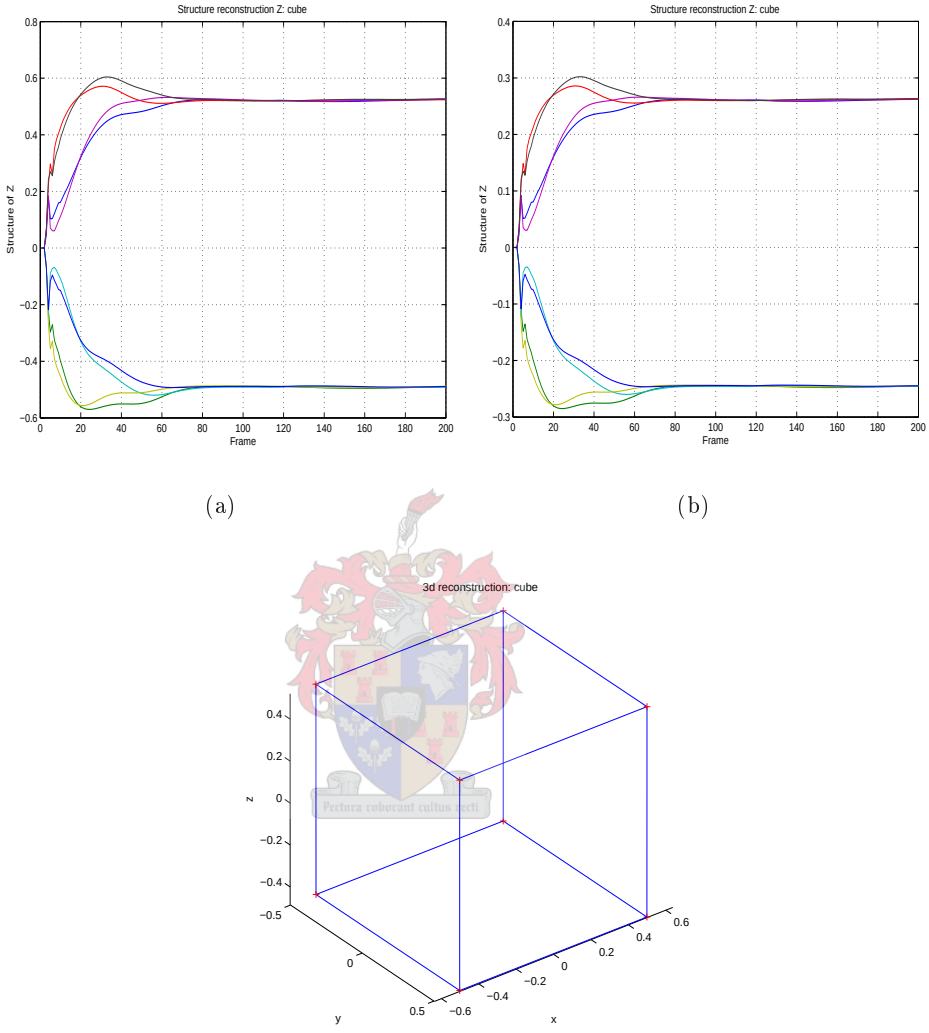
Figure 7.36: *Simulation of UKF and camera frame rate; (a) Using frame rate 1, (b) Frame rate 1/2 (c) Frame rate 1/3 - convergence at 80, 160 and 300 frames respectively*



(a) (b)

(c)

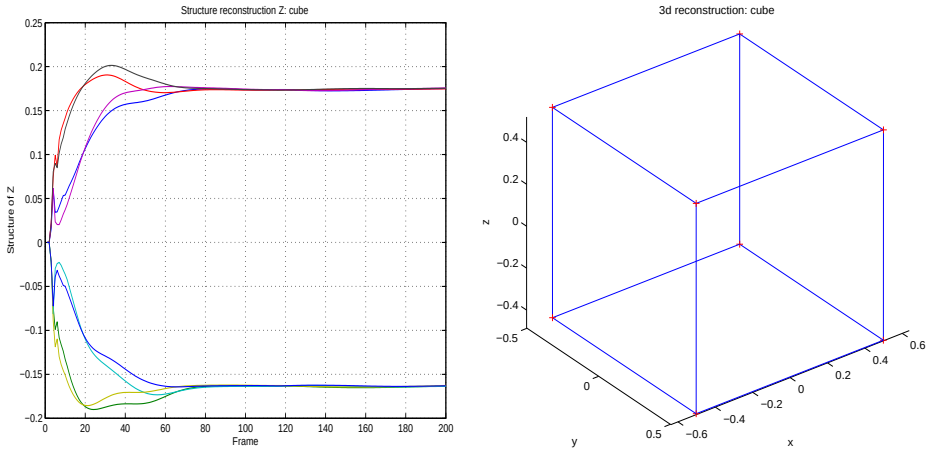
Figure 7.37: *Simulation of UKF and camera frame rate; (a)* 3D *reconstruction frame rate 1/2 at 160th frame, (b)* 3D *reconstruction at frame rate 1/3 at 300th frame (c)* 3D *reconstruction frame rate 1/3 at 400th frame*



(a) (b)

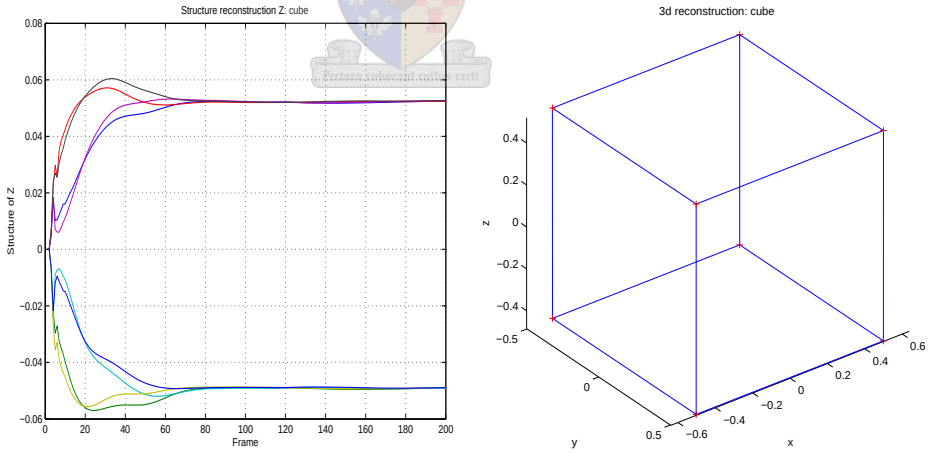
(c)

Figure 7.38: *Simulation of UKF and camera frame rate; (a) Using frame rate 1, (b) Frame rate 2 - convergence at 80 and 70 frames respectively, (c)* 3D *reconstruction*



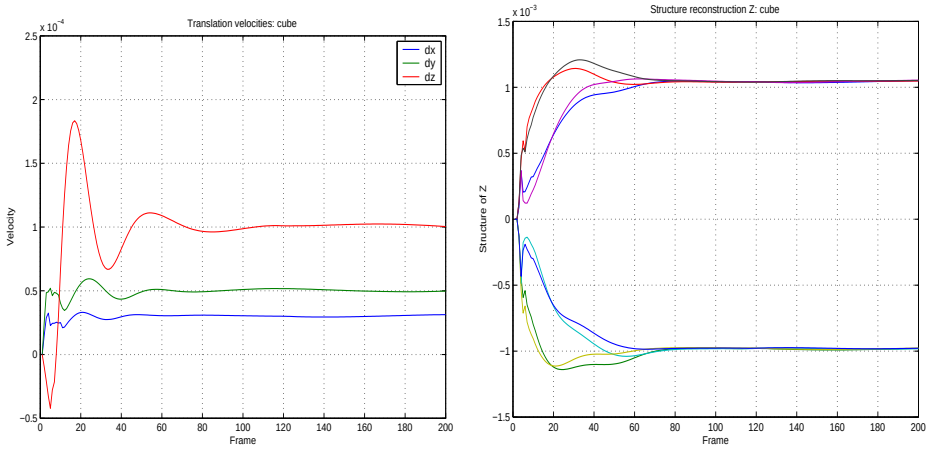
(a) (b)

Figure 7.39: *Simulation of UKF and camera frame rate; (a) Using frame rate 3 - convergence at 68 frames and (b)* 3D *reconstruction*

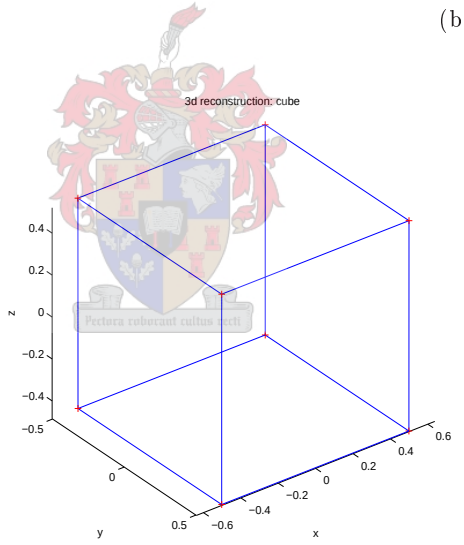


(a) (b)

Figure 7.40: *Simulation of UKF and camera frame rate; (a) Using frame rate 10 - convergence at 68 frames and (b)* 3D *reconstruction*

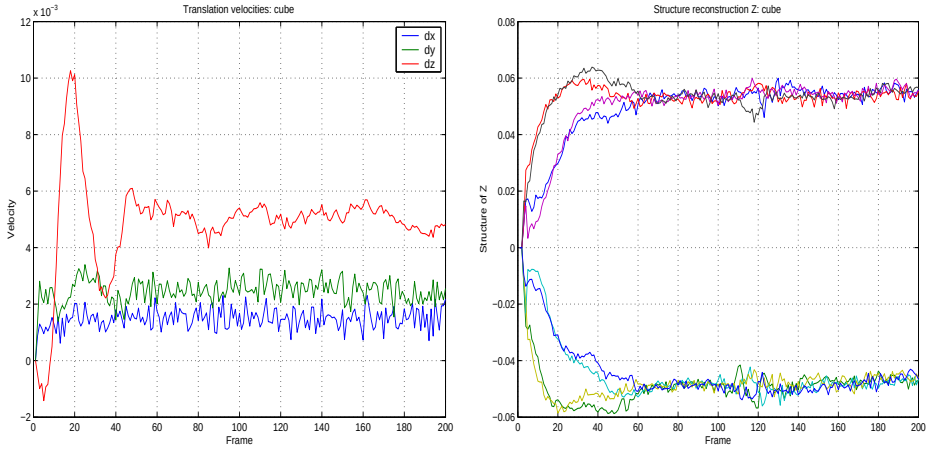


(a) (b)

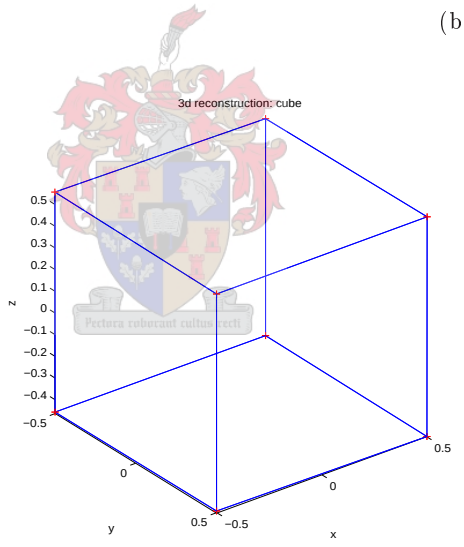


(c)

Figure 7.41: *Simulation of UKF and camera frame rate; (a) Using frame rate 1, (b) Frame rate 15 - convergence at 80 and 68 frames respectively, (c)* 3D *reconstruction*

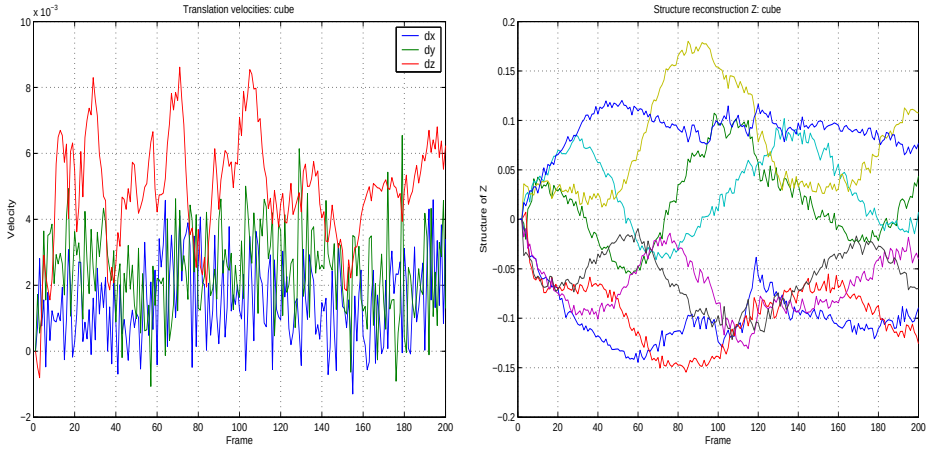


(a) (b)

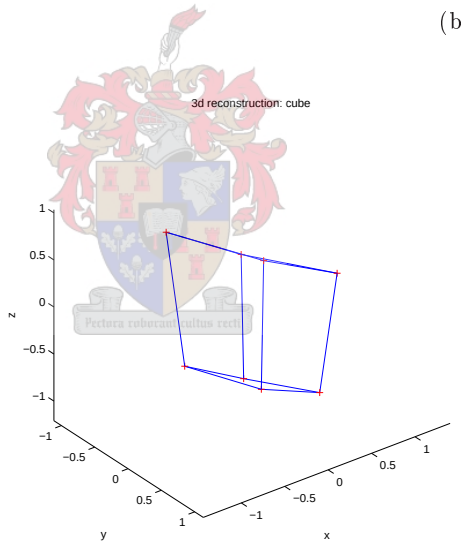


(c)

Figure 7.42: *Simulation of UKF, camera frame rate and noise of 0.0001; (a) Translation Velocities, (b) Convergence with frame rate 10 and (c)* 3D *reconstruction*

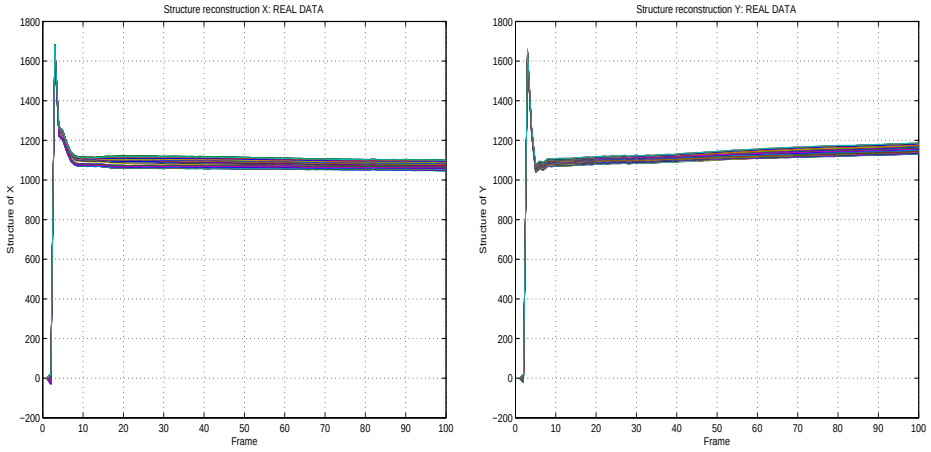


(a) (b)

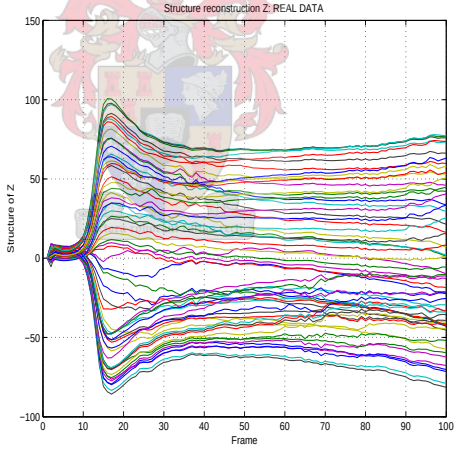


(c)

Figure 7.43: *Simulation of UKF, camera frame rate and noise of 0.001; (a) Translation Velocities, (b) Convergence with frame rate 10 and (c)* 3D *reconstruction*

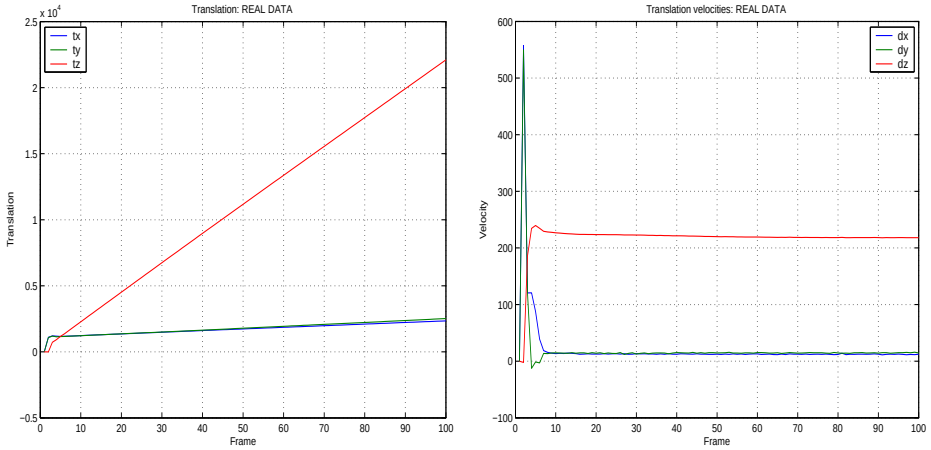


(a) (b)

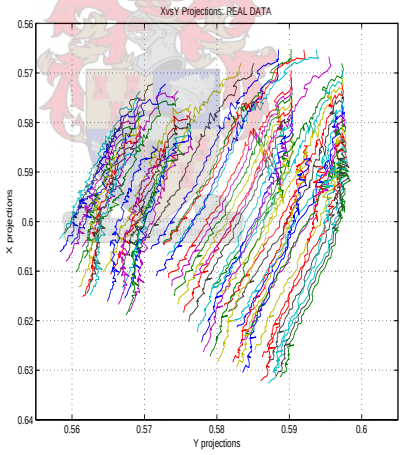


(c)

Figure 7.44: *Real data set 1 - Simulation of UKF and camera frame rate using frame rate 1; (a) x-component, (b) y-component, (c) z-component*



(a) (b)



(c)

Figure 7.45: *Real data set 1 - Simulation of UKF and camera frame rate using frame rate 1; (a) Translation, (b) Linear velocity, (c) Observation*

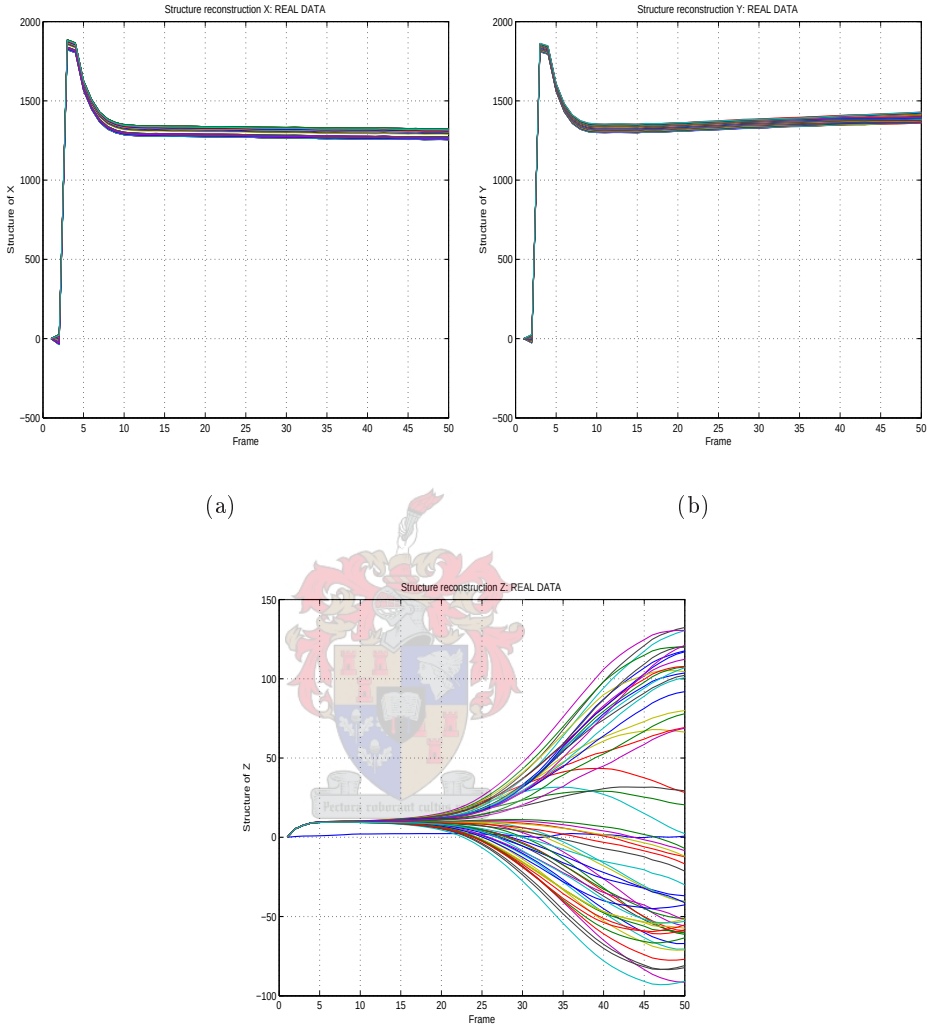
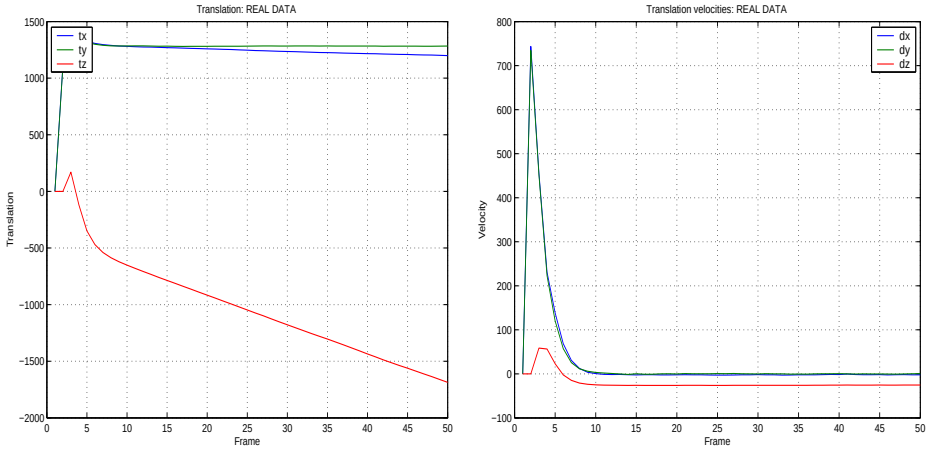
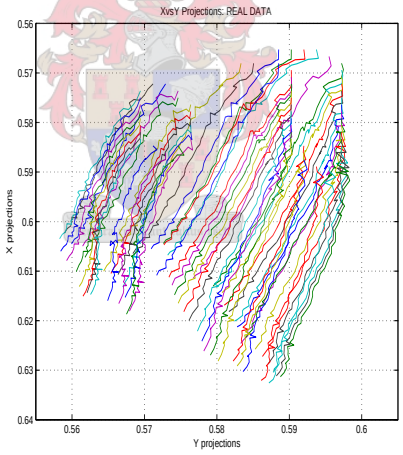


Figure 7.46: *Simulation of UKF and camera frame rate using frame rate 2; (a) x-component, (b) y-component, (c) z-component*

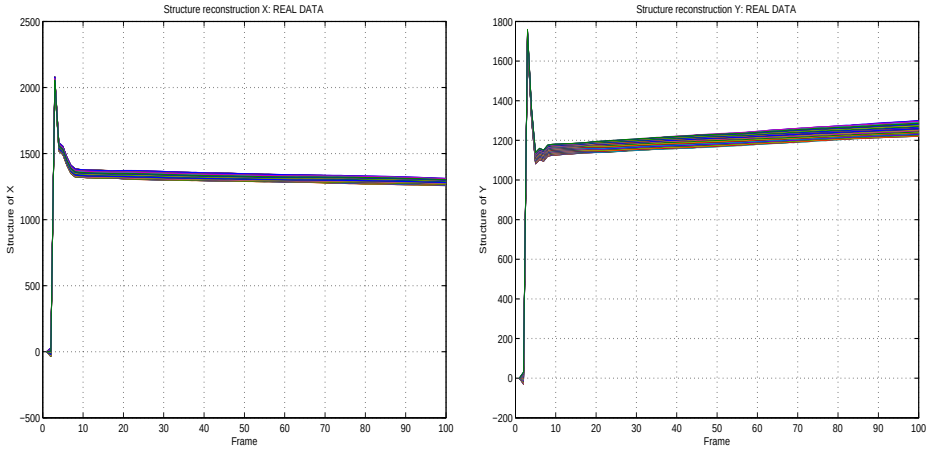


(a) (b)

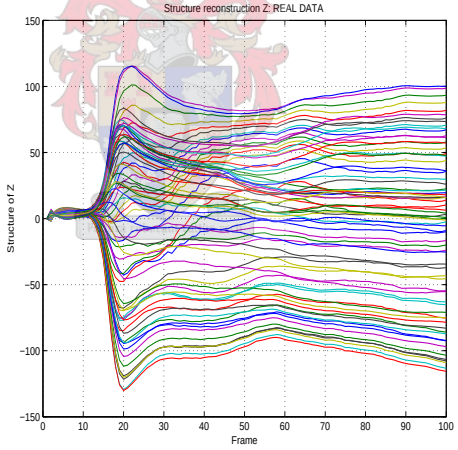


(c)

Figure 7.47: *Real data set 1 - Simulation of UKF and camera frame rate using frame rate 2; (a) Translation, (b) Linear velocity, (c) Observation*



(a) (b)



(c)

Figure 7.48: *Real data set 2 - Simulation of UKF and camera frame rate using frame rate 1; (a) x-component, (b) y-component, (c) z-component*

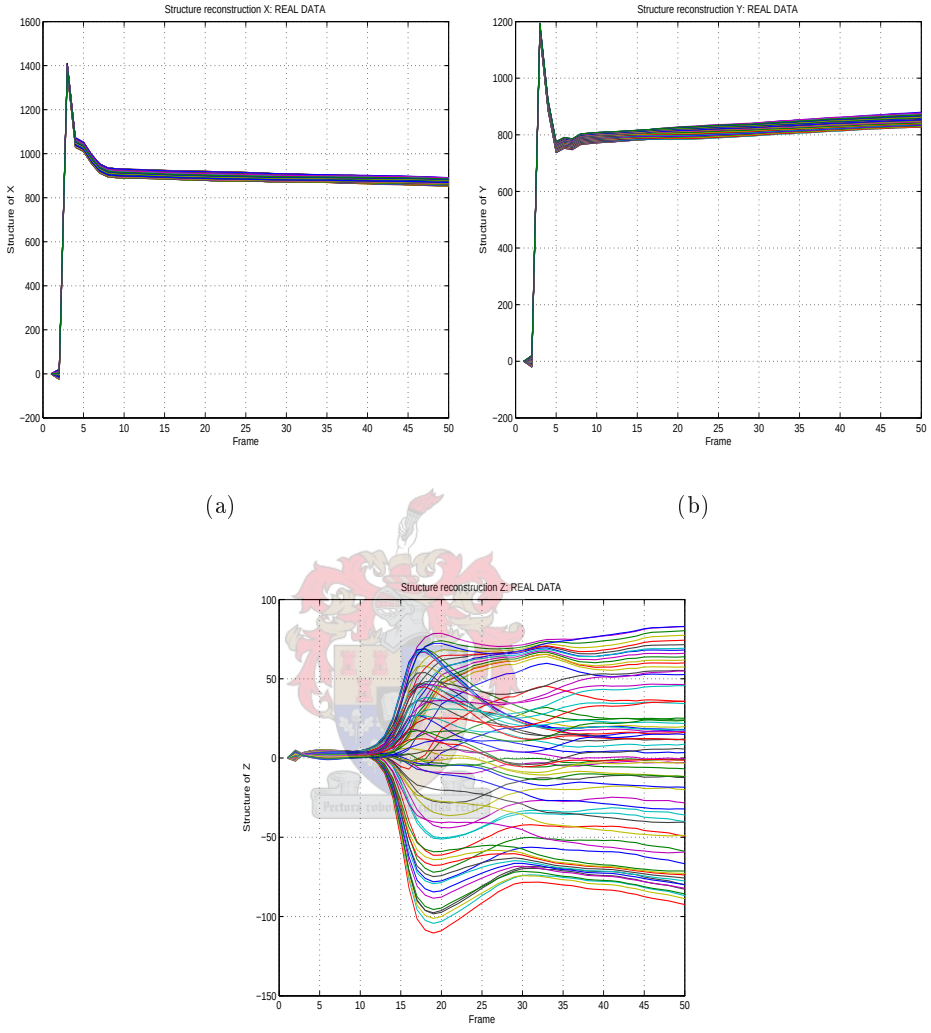
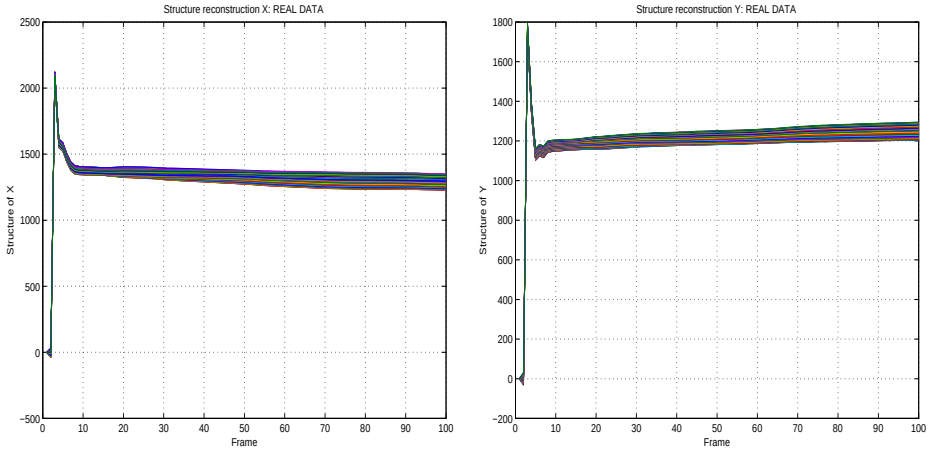
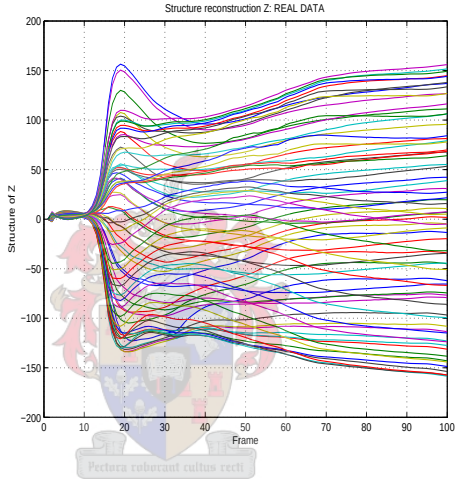


Figure 7.49: *Real data set 2 - Simulation of UKF and camera frame rate using frame rate 2; (a) x-component, (b) y-component, (c) z-component*



(a) (b)



(c)

Figure 7.50: *Real data set 3 - Simulation of UKF and camera frame rate using frame rate 1; (a) x-component, (b) y-component, (c) z-component*

7.4 Discussion

In this section, we discuss the experimental results obtained in all the above sequences. The background computation was achieved using either histogram or frame by frame differencing. We have shown that the histogram approach is sensitive to noise. The clustering was possible by employing the $k-means$ algorithm. We have also shown that the *3D* reconstruction does not depend on mis-clustering. The resulting image is always perfect. We have employed Euclidean distance as clustering measurement. A parameter of 12 has worked correctly for most cases. The use of

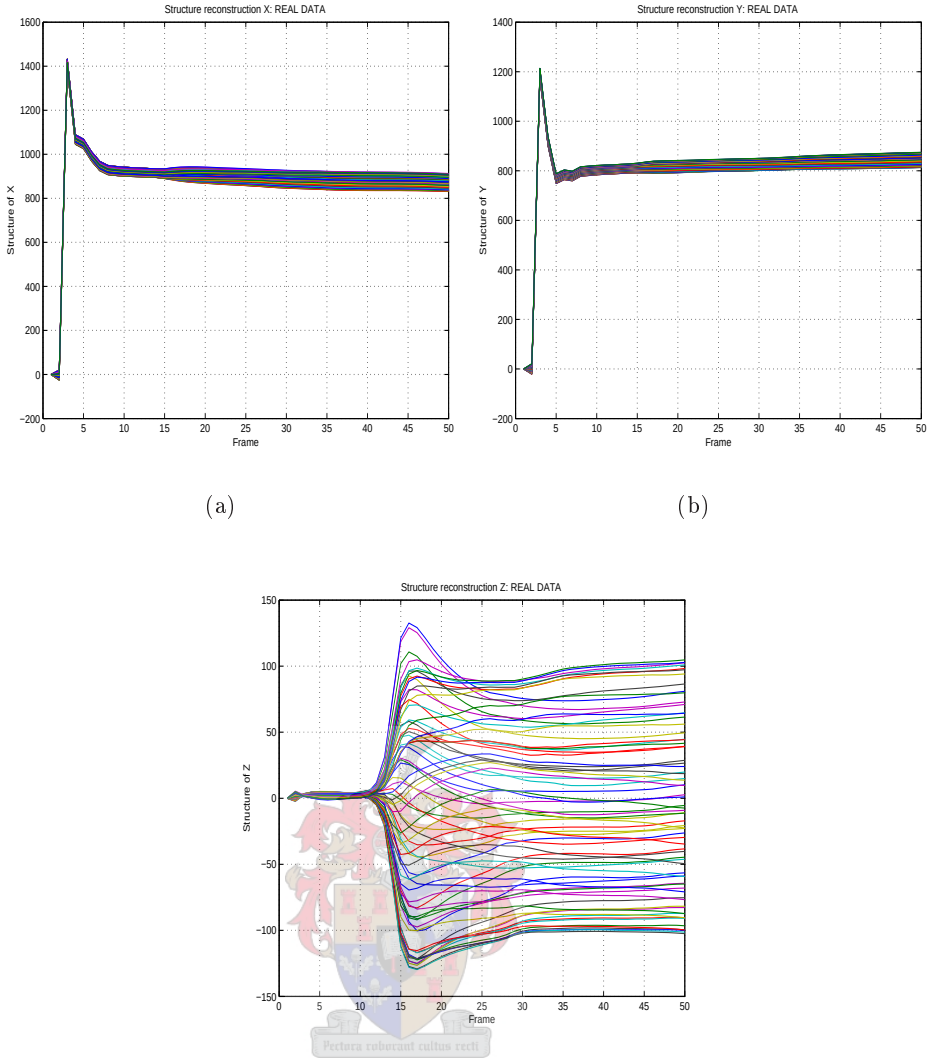


Figure 7.51: *Real data set 3 - Simulation of UKF and camera frame rate using frame rate 2; (a) x-component, (b) y-component, (c) z-component*

dense optical flow as feature tracker enabled us to track as many features as possible. The tracking procedure took less than five minutes.

The *3D* reconstruction program was written in Matlab. As a result, one should expect the time for the final result to depend on how many features are being reconstructed. For instance, for 219 features it takes about fifteen minutes, but for 1062 features the processing time was about 126 minutes of which 4.85 minutes were spent tracking. We were able to recover the full *3D* structure using *Unscented Kalman Filter* from 80 frames. For calibration, we take a video sequence of a known car and

we measure its speed and the result of *3D* reconstruction. We observed where the *UKF* converges and from that point we consider all *3D* data to be correct. Next we used the correct *3D* data to determine the *3D* scale factor. As a result, we are able to compute the car's velocity at a given time. We have simulated the relationship between the frame rate and the convergence of *Unscented Kalman Filter*. We found out that the employment of a higher speed camera could significantly improve the convergence of *Unscented Kalman Filter*. Figure 7.41 shows that the system could converge in less that sixty frames. We were not able to verify this using real data due to the number of frames required in the sequence[3].

The system calibration is only possible after the correct computation of *3D* data

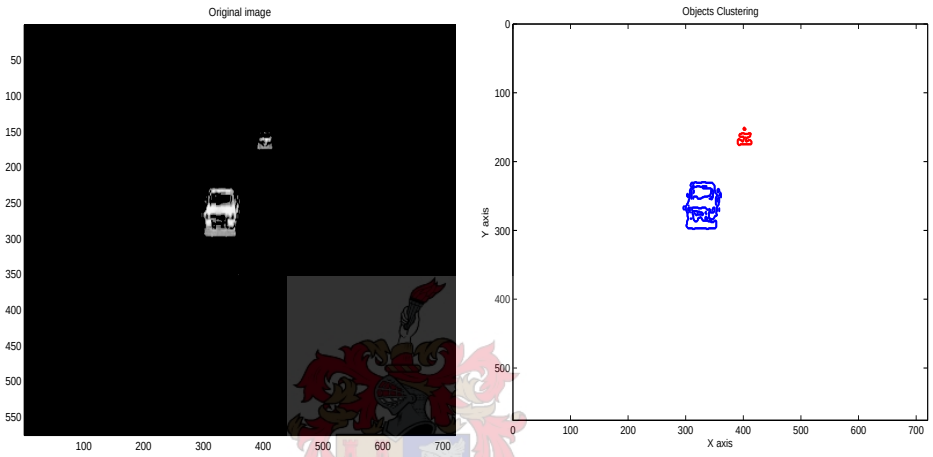
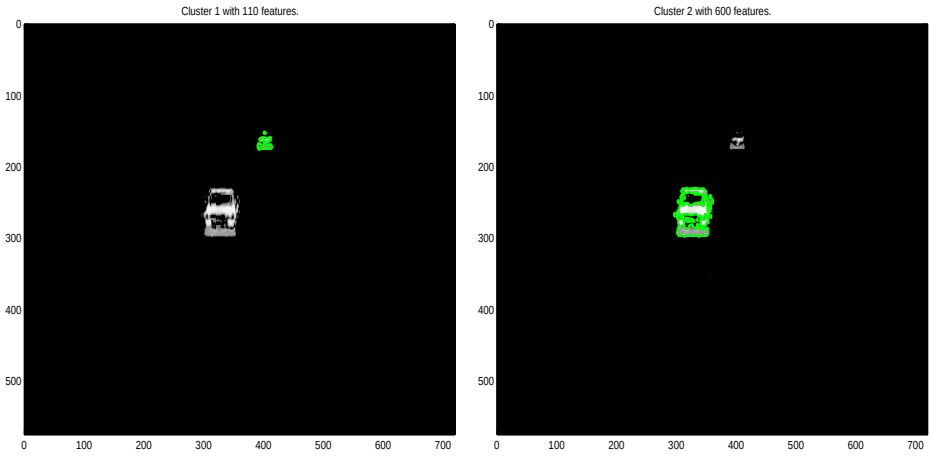


Figure 7.52: *Clustering using a correct parameter: (a) Original image and (b) Respective clusters*

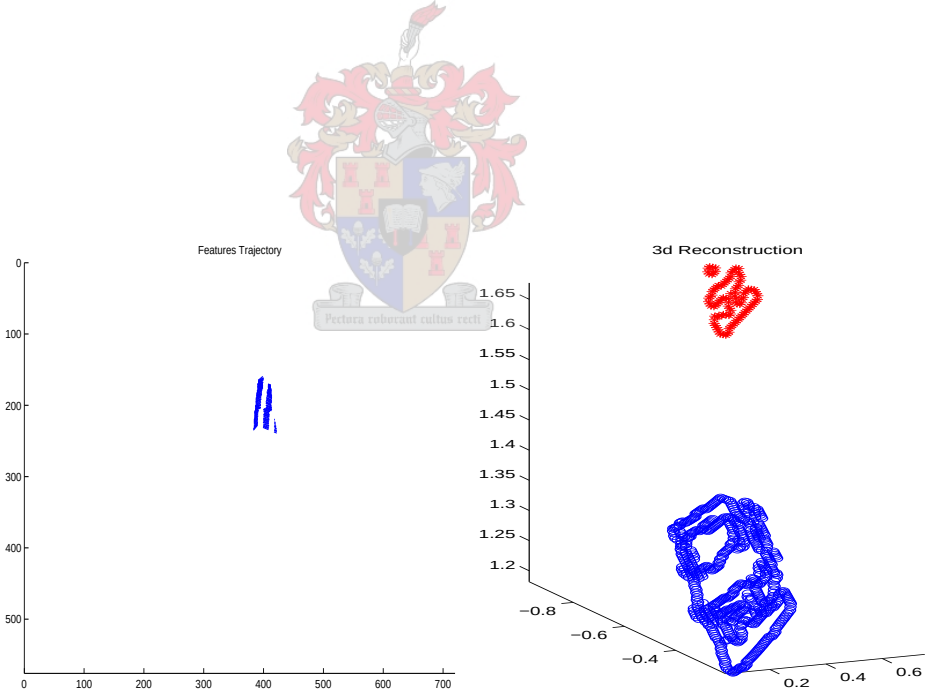
structure of the objects.

[3]Note that we needed a car to be visible for at least 300 frames which is long time.



(a) (b)

Figure 7.53: *Clustering using a correct parameter: (a) First cluster and (b) Second clusters*



(a) (b)

Figure 7.54: *Clustering using a correct parameter: (a) Observation recovering and (b)* 3D *reconstruction*

Chapter 8

Conclusion and Recommendations

8.1 Conclusion

There is one main theme running through this thesis: It presents a traffic monitoring system based on a long video sequence.

The main goal of the research is to ascertain the feasibility of *Structure-from-Motion* techniques to solve traffic monitoring problems. In this regard, we designed our system so that it should provide useful information about traffic such as speed and number of cars.

In order to achieve our goal we have included the following aspects in our approach

- background extraction,
- features selection,
- robust features tracking,
- employment of noise modelling,
- obtaining accurate *3D* information and
- system calibration.

During our research, we explored *Structure-from-Motion* techniques to build a *video-based traffic monitoring system*. We found that the orthographic model cannot be used to solve our problem due to the fact that it assumes infinite focal length only. Next, we applied the pinhole camera model which includes noise modelling.

The background extraction problem was solved by the use of frame differencing. It is important to point out that the histogram technique did not perform correctly

over all sequences.

For the feature selection, we applied *Harris Corner Detector* in order to get desirable features. We opted for this detector because the *Kanade-Lucas-Tomasi* feature-detection method does not consider good features to track when they are on the corners and we were particularly interested in tracking corners because they describe an object better.

A further aspect of the system was clustering. We wanted to compute the speed of vehicles and also to count vehicles. In order to compute the speed of an object we had to track the features of the object. We applied dense optical flow for features tracking.

The clustering technique failed when vehicles were very close to each other. As a result, the vehicle counting was not performed correctly.

The system achieved the goal of speed computation. After feature tracking, we applied a *3D* reconstruction model which employed the *Unscented Kalman Filter*. The proposed models for the *motion* and the *structure* of the object corresponded to the dynamic state and static model of a dual-estimation system, as described in Chapter 5. In these models, noise was modeled and the system computed both the structure and the velocity of the object. When the structure was achieved automatically we had succeeded in computing the correct speed.

The last problem in the system was calibration. Calibration consisted of determining the *3D* scale factor. To achieve this goal, first we took the speed of a known car. Secondly we computed its structure using a *Unscented Kalman Filter*. Finally, using the knowledge of structure and real speed we could calculate the *3D* scale factor. This could be applied to the speed of other vehicles on the same image sequence. It is important to point out that if the camera moves then a new *3D* scale factor must be determined.

In general, the system takes about eighty frames to converge. This can be seen as a long time for real situations. We simulated a higher speed camera using synthetic data and the results showed that if we can employ a higher speed camera our system will converge at almost half the frames or that we could reduce the system convergence time by half. We were not able to test this using real data because we did not have a higher speed camera and we did not have an object for a very long sequence.

The system has the following advantages:

1. The sequence may contain one or more moving objects.
2. The object must be rigid but can be subject to perspective projection.
3. The system can be used for real-data applications.
4. The system can easily be adapted for other surveillance systems.
5. The system can be adapted for vehicle classification.

Although we have just shown some advantages of the system, the system also has the following constraints:

1. The sequence must have a minimum of eighty frames.
2. The background needs to be updated for every long image sequence.
3. Each iteration depends on the previous measurements.

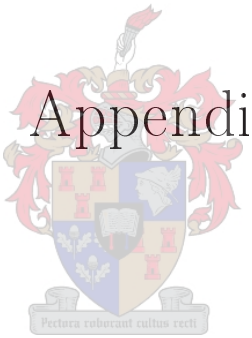
Finally, we emphasize the fact that besides **our contribution** given in Section 1.9 the following holds: (i) the adaptability of the system for vehicle classification and (ii) the versatility of the system which means that it can be easily used for other surveillance systems.

8.2 Recommendations

The system still has some shortcomings. Therefore, we make the following recommendations:

- For object clustering we propose the use of *3D* data, but more criteria, such as color and optical flow velocities, should be investigated.
- Robust optical flow was applied to track features, but it is still expensive. Therefore other tracking techniques should be explored.
- The system is unable to track objects under occlusion. In order to overcome this difficulty it is necessary to include a process of recovering occluded features.
- The frame rate using *real data* must also be investigated. For this purpose a higher speed camera should be used.

Appendices



Appendix A

Mathematical Background

A.1 Random Noise

The *probability density function (pdf)* is the statistical function that shows how the density of possible observations in a population is distributed. The *pdf* has a different meaning depending on whether the distribution is discrete or continuous.

The *pdf* (discrete distributions) is the probability of observing a particular outcome.

The *pdf* (continuous distribution at a value) is not the probability of observing that value. In order to get probabilities we must integrate the pdf over an interval of interest.

A *pdf* has two theoretical properties:

- The *pdf* is zero or positive for every possible outcome.
- The integral of a *pdf* over its entire range of values is one.

We will refer the *Gaussian noise* or *Gaussian white noise* as noise which has a *pdf* of the Normal distribution (Gaussian distribution).
White Random Vector: $\mathbf{w}$ is white random vector if and only if its mean vector and autocorrelation matrix have the following properties:

$$\begin{aligned} \mu_w &= \mathbf{E}\{\mathbf{w}\} = 0, \\ R_{ww} &= \mathbf{E}\{\mathbf{w}\mathbf{w}^T\} = \sigma^2\mathbf{I}. \end{aligned} \tag{A.1}$$

That is, *white noise* has a zero mean, and its autocorrelation matrix is a multiple of the identity matrix. This implies that noise is uncorrelated.

A.2 Rotation Matrix

Consider R_{xyz} to be the general (full *3D* or of rank 3) rotation matrix. Then it can be written as: R_{xyz}=$R_x R_y R_z$, where R_x is a rotation matrix over the $x-axis$ and

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}, \tag{A.2}$$

R_y is a rotation matrix over the $y-axis$ and

$$R_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \tag{A.3}$$

R_z is a rotation matrix over the $z-axis$

$$R_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.4}$$

A.3 Review of the Singular Value Decomposition

We briefly review the SVD and several of its properties that are important for our applications. This discussion is restricted to real matrices. Among excellent references on SVD are Demmel [45], Golub and van Loan [52], Trefethen and Bau [143]. The SVD of a matrix X of size $m \times n$ is

$$X = U\Sigma V^T, \tag{A.5}$$

where U and V are $m\times m$ and $n\times n$ orthonormal matrices and Σ is an $m\times n$ diagonal matrix with the nonnegative singular values σ_j, $j = 1,\ldots,\min(m,n)$, arranged in non increasing order along the diagonal. The columns of U and V are denoted by the vectors $\boldsymbol{u}_j$, $j = 1,\ldots,m$, and $\boldsymbol{v}_j$, $j = 1,\ldots,n$. An important geometrical interpretation of SVD can be found in Muller et al [103]. A direct consequence of the geometric interpretation is that the largest singular value, σ_1, measures the "magnitude" of X is a 2-norm:

$$\|X\|_2 = \sup_{\|\boldsymbol{x}\|_2=1} \|X\boldsymbol{x}\|_2 = \sigma_1. \tag{A.6}$$

Expressions for U, V, and Σ follow readily from (A.5),

$$XX^TU = U\Sigma\Sigma^T \text{ and } X^TXV = V\Sigma^T\Sigma, \tag{A.7}$$

demonstrating that the columns of U are the eigenvectors of XX^T and the columns of V are the eigenvectors of X^TX.

The rank of X is the number of nonzero singular values. Thus if $\text{rank}(X) = r$, it is possible to rewrite the SVD in its reduced form

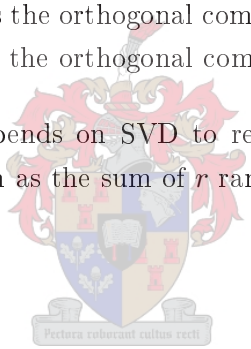
$$X = U_+ \Sigma_+ V_+^T, \tag{A.8}$$

where Σ_+ is the $r \times r$ diagonal matrix with the r nonzero singular values of X, and U_+ and V_+ consists of the first r columns of U and V, respectively. It is straightforward to prove that the first r columns of U form an orthonormal basis for the column space of X, the first r columns of V form an orthonormal basis for the row space of X, the remaining $m - r$ columns of U form an orthonormal basis for the left null-space of X, and the last $n - r$ columns of V form an orthonormal basis for the null-space of X. Thus we arrive at the following important result:

- The column and row spaces of X both have dimension r. The null-space of X has dimension $n - r$, and the left null-space of X has dimension $m - r$.
- The null-space of X is the orthogonal complement of the row space in R^n. The left null-space of X is the orthogonal complement of the column space in R^m.

The *3D* reconstruction depends on SVD to reveal the effective rank of a matrix. Note that X can be written as the sum of r rank-1 matrices

$$X = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T. \tag{A.9}$$



This implies that the zero singular values may be ignored since they carry no "information." As it is illustrated in Muller et al [103], X can be approximated using the ν-dimensional ellipsoid as

$$X_\nu = \sum_{j=1}^{\nu} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T. \tag{A.10}$$

Each term is associated with one of the principal directions $\boldsymbol{u}_j$ of the hyper ellipsoid. Since the difference $X - X_\nu$ is a matrix with singular values $\sigma_{\nu+1}, \ldots, \sigma_r$, the 2-norm is again given by the largest singular value

$$\|X - X_\nu\|_2 = \sigma_{\nu+1}. \tag{A.11}$$

In this expression $\sigma_{\nu+1} = 0$ if $\nu > r$. Indeed, it can be shown that X_ν is the best approximation to X (in the 2-norm sense) over all matrices with rank $\leq \nu$. If $\sigma_{\nu+1}$ is sufficiently small, it is safe to keep only ν singular values and the rank of X is ν.

A.4 The Quaternion

The quaternions are members of a non-commutative division algebra invented by William Rowan Hamilton. The idea for quaternions occurred to him while he was walking along the Royal Canal on his way to a meeting of the Irish Academy. Hamilton discovered the fundamental formula of quaternion algebra, $i^2 = j^2 = k^2 = -1$. Quaternions are a single example of a more general class of hyper-complex numbers which are associative. They form a group known as the quaternion group.

Quaternions have been used in computer vision as early as 1983, Pervin and Webb [110]. Faugeras describes the use of quaternions to represent rotation in Faugeras [47]. Wheeler and Ikeuchi describe an iterative rotation and translation estimation algorithm which uses quaternions, Wheeler and Ikeuchi [162]. Real-world use of quaternions is popular in computer graphics and games, since they allow easy interpolation between two rotations, Bobick [22]. Quaternions form an essential part of *SfM* estimation, since it provides a simple and efficient way to describe the rotation of a tracked object, Choukroun et al [31], Vicci [151]. A normalized quaternion $\mathbf{q}$ has four parameters, but only three degrees of freedom, enforced by the normality constraint. It is represented by

$$\mathbf{q} = [s, \mathbf{v}] = (s, v_1, v_2, v_3) \tag{A.12}$$

where

$$\begin{aligned} \|\mathbf{q}\| &= \sqrt{s^2 + v_1^2 + v_2^2 + v_3^2} && \text{(A.13)} \\ &= 1. && \text{(A.14)} \end{aligned}$$

A.5 Rotations Using Quaternions

A normalized quaternion is related to the axis-and-angle representation of rotation by

$$\mathbf{q} = [\cos(\theta/2), a\sin(\theta/2), b\sin(\theta/2), c\sin(\theta/2)], \tag{A.15}$$

so that

$$\begin{aligned} s &= \cos(\theta/2) && \text{(A.16)} \\ \mathbf{v} &= \mathbf{n}\sin(\theta/2) && \text{(A.17)} \\ (v_1, v_2, v_3) &= (a, b, c)\sin(\theta/2), && \text{(A.18)} \end{aligned}$$

where $\mathbf{n} = (a, b, c)$ is the normalized axis of rotation[1], and θ describes the angle of rotation about the axis. Therefore, the quaternion representation is very close to the minimal axis-and-angle rotation, and can be easily transformed into a rotation matrix. The 3×3 rotation matrix $\mathbf{R}$ can be computed from the quaternion as

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} s^2 + v_1^2 - v_2^2 - v_3^2 & 2(v_1 v_2 + s v_3) & 2(v_1 v_3 - s v_2) \\ 2(v_1 v_2 - s v_3) & s^2 - v_1^2 + v_2^2 - v_3^2 & 2(v_2 v_3 + s v_1) \\ 2(v_1 v_3 + s v_2) & 2(v_2 v_3 - s v_1) & s^2 - v_1^2 - v_2^2 + v_3^2 \end{bmatrix}. \quad \text{(A.19)}$$

A.6 Optical Flow Terms

A.6.1 Lambertian Surface

Lambert's cosine law states that the reflected or transmitted luminous intensity in any direction from an element of a perfectly diffusing surface varies as the cosine of the angle between that direction and the normal vector of the surface. As a consequence, the luminance of that surface is the same regardless of the viewing angle.

A Lambertian surface is a surface of perfectly matte properties, which means that it adheres to Lambert's cosine law.

A.6.2 Luminous Intensity

The luminous intensity is the luminous flux emitted from a point per unit solid angle into a particular direction. The standard unit of luminous intensity (base unit for light) is Candela (cd), also expressed as Lumen per Steradian (lm/sr).

A.6.3 Normal Vector to Surface

The vector perpendicular to a surface determines the orientation of the surface. In the case of polygons, this direction is usually determined by the right hand rule. In computer graphics, manipulations of the normal vector are often used as a way to simulate geometrical detail on planar surfaces.

A.7 Lagrange Multiplier

One of the most common problems in calculus is that of finding maxima or minima (extrema) of a function, but it is often difficult to find a closed form for the function being extremized (maximized or minimized). Such difficulties often arise when one wishes to maximize or minimize a function subject to fixed outside conditions or

[1] Note that $\|q\| = 1$ implies $\|n\| = 1$ because $\sin^2 \alpha + \cos^2 \alpha = 1$.

constraints. The method of Lagrange multipliers is a powerful tool for solving this class of problems without the need to explicitly solve the conditions and use them to eliminate extra variables.

An Example of Lagrange Multipliers

What shape should a rectangular box with a specific volume (in *3D*) be in order to minimize its surface area?

Questions like this are very important for businesses and applied industrial sciences that deal with problem of money saving on materials and resources.

Consider the lengths of the box's edges be x, y, and z. Then the constraint of constant volume (V) is simply $\mho(x, y, z) = xyz - V = 0$, and the function to minimize is $f(x, y, z) = 2(xy + xz + yz)$. The method is straightforward to apply:

$$2\begin{bmatrix} y+z \\ x+z \\ x+y \end{bmatrix} = \bigtriangledown(f(x,y,z)) \tag{A.20}$$

$$= \lambda \bigtriangledown (\mho(x,y,z)) \tag{A.21}$$

$$= \lambda \begin{bmatrix} yz \\ xz \\ xy \end{bmatrix}. \tag{A.22}$$

Now just solve those three equations; the solution is $x = y = z = 4/\lambda$. We could eliminate λ from the problem by using $xyz = V$.

Appendix B

Equipment, Data and Code Directories

This appendix describes the hardware and computer software used in this thesis. We also describe the directory tree.

B.1 Hard- and Software Specifications

The author's system, with the following specifications, was used for feature tracking, clustering and reconstruction:

Computer	**teak.tw.sun.ac.za**
Processor	Intel(R) Celeron(R) CPU 1.70GHz and 512Mb RAM
Operating system	Redhat Linux (Fedora Core 3)
Kernel version	2.6.8-1.521
Matlab version	6.5.0.10183 (R13)
C/C++ Compiler	gcc version 3.4
Video Camera type	Canon Digital Camcorder, Model MV550i

B.2 The Directory Tree: `thesis`

This directory contains all documents, scripts, codes and data used in this thesis.

B.2.1 `thesis/codes`

This directory contains data images, all Matlab and C/C++ source codes. The codes and data are stored in different subdirectories.

B.2.1.1 `thesis/codes/data`

The `data` directory contains the data to be used in the experiments.

B.2.1.2 `thesis/codes/matlab`

Contains some Matlab programs, such as background extraction and region of interest definition.

B.2.1.3 `thesis/codes/ttrack`

Most subdirectories containing source codes can be found under this directory. This directory contains the main programs which call the other routines in the various subdirectories.

B.2.1.4 `thesis/codes/ttrack/oflow`

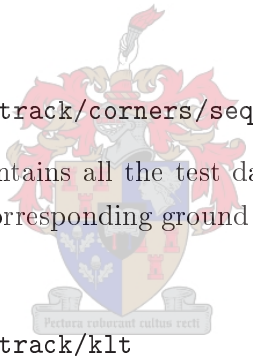
The `thesis/ttrack/oflow/` directory and subdirectories contain all the Matlab and C/C++ source code used for optical flow computation.

B.2.1.5 `thesis/codes/ttrack/corners`

In this directory are stored C/C++ codes for corner detection and tracking. The resulting data are stored in a file called corners which can easily be accessed by Matlab read-feature file.

B.2.1.6 `thesis/codes/ttrack/corners/sequence`

The `sequence` directory contains all the test data for the corner-tracker algorithm. It contains source images corresponding ground truth and feature lists for the experiments.



B.2.1.7 `thesis/codes/ttrack/klt`

This directory contains the feature tracking software based on the Kanade-Lucas-Tomasi feature-detection. A tracking algorithm was used. The code was written as a library of functions by Stan Birchfield and can be downloaded from [15]. The version used in this document has been adapted to our needs and saved in the directory `thesis/codes/ttrack/klt` as *track.c*. We have enabled feature tracking setup by the user.

B.2.1.8 `thesis/codes/ttrack/data`

The `data` directory contains all the test data used. It contains source images and corresponding feature lists for the experiments. `my_object_*.pgm`: Source images for *KLT Tracker*. `my_frame_*.png`: Source images for background computing. `feat_*.mat` (produced by KLT-Tracker): Source images for *3D* reconstruction and clustering. Final feature lists of tracked features using KLT Tracker. In this directory are four important subdirectories;

- cube: contains synthetic data used to check the algorithm
- M01: first sequence taken of Merriman Avenue in Stellenbosch
- M02: second sequence of Merriman Avenue
- M03: third sequence of Merriman Avenue
- M04: fourth sequence of Merriman Avenue
- M05: fifth sequence of Merriman Avenue
- M06: Calibration sequence taken on Merriman Avenue
- M07: Last sequence taken on Merriman Avenue
- N201: first sequence taken of the National Highway N2. The sequences of type N2[xx] were taken about 30 km from Cape Town.
- N202: second sequence taken of the National Highway N2
- N203: third sequence taken of the National Highway N2
- N204: fourth sequence taken of the National Highway N2
- N205: fifth sequence taken of the National Highway N2
- R10201: first sequence of the National Road R102. The sequences of type R102[xx] and R310[xx] were taken about 15 km from Stellenbosch towards Somerset-West.
- R10202: second sequence of the National Road R102
- R31001: first sequence taken of the National Road R310
- R31002: second sequence taken of the National Road R310
- R4401: first sequence taken of the National Road R44. The sequence was taken near the Stellenbosch train station.

Each of the above directories contains three more subdirectories, namely

- feats: contains `feat_*.mat` files
- my_frame: here `my_frame_*.png` files can be found
- my_objects: contains files with moving objects; `my_object_*.pgm`

B.2.1.9 `thesis/codes/data/merriman01/feats`

Once the images have been processed with feature tracking software, the feature lists are automatically copied into this pre-defined directory, so that Matlab scripts can easily locate and process them.

B.2.1.10 `thesis/codes/ttrack/general`

This directory contains scripts for the code shared between different *SfM* algorithm implementations. Examples include the error generating code, plotting functions and other general use functions.

B.2.1.11 `thesis/codes/ttrack/libmat`

Matlab code library for quaternion mathematics and the Unscented Kalman Filter code.

B.2.1.12 `thesis/codes/kltC`

In this directory there are collection of C/C++ source codes and Matlab designed to track features including boundaries and edges.

B.2.2 `thesis/paper`

In this directory, `thesis/paper`, there are images/figures and other documents sourced for this thesis. It is a standard LaTeX [83] document.

B.2.2.1 `thesis/paper/fig`

The Encapsulated Postscript graphics used in the document are found in `thesis/paper/fig`.

B.2.2.2 `thesis/paper/lib`

The styles and Bibliography files used in this thesis are located in `thesis/paper/lib`.

Appendix C

Software Usage and Demonstration

This appendix describes the code used in this thesis. We describe how to reproduce the results.

C.1 Software Usage

The user should follow the following steps:

- Acquire a video sequence, if not available
- Compute the background
- Compile the programs `track.c`, `track_edge.c`, `corner.c` and `gnc.c`
- Track features using the above programs
- Apply the algorithm
- Separate objects using clustering algorithms (k-means procedure)

C.1.1 Compiling `track` Programs

The feature tracking programs, `track.c` or `track_edge.c`, are compiled by executing

```
$ cd thesis/codes/ttrack/klt
$ make
```

then

```
$ cd thesis/codes/ttrack/oflow/RobustFlow
$ make
```

and

```
$ cd thesis/codes/kltC
$ make
```

at the prompt in the same directory. If the user wants to track edges, one should first run the program called *edges.m*. This will create files named `my_objects_*.edge` in the indicated output directory, for example,

`thesis/codes/ttrack/data/merriman02/edges`

C.1.2 Input File Type

The images must be in Portable gray-map format, with filenames of the form `base_[x].pgm` or `base_[x].edge` where `base` is a shared pre-fix and `x` is an integer number of the frame in the sequence. Therefore `my_frame_1.pgm` is a valid filename, which is the first frame in the `my_frame` sequence but it does not necessarily have to start at 1.

For example, to track features in an image sequence with the filenames:
$my\_frame\_1.pgm$, $my\_frame\_2.pgm$, ...,$my\_frame\_122.pgm$
enter the following command at the prompt:

```
./track merriman02/data/my_objects 1 500 99 data/merriman02/data/feats
merriman02/data/features.txt 2
```

This will produce the output files
$feat\_1.mat, feat\_2.mat, feat\_3.mat$, ..., $feat\_99.mat$ and
$feat\_0.ppm, feat\_1.ppm, feat\_2.ppm$, ..., $feat\_98.ppm$
in the directory `thesis/codes/ttrack/data/feats` enabling Matlab programs to find and use them. In this command line, the parameters are:

- `my_objects`: the directory containing tracking data (moving objects), `my_objects_*.pgm`;
- 1: the start frame;
- 500: the maximum number of features to try to find and track;
- 99: the number of frames to track over;
- feats: the output directory for $feat\_1.ppm$ and $feat\_1.mat$;
- features.txt: the output text file, if we want to read data from a text file;
- 2: the replacement interval between frames. If it is omitted then a default value of 10 will be assumed automatically (see line 63 of *track.c*) ;

The images in the `thesis/codes/ttrack/data/merriman02` directory, for example, are already in "PGM" and "PNG" format [114].

C.1.3 Running an Experiment

We first convert a video sequence to a series of frames. Then we process the background and modify the `initialise.m` and `buildhist0.m` files to meet the required experiment for instance, define the region of interest in appropriate lines. The experiment is run using

```
» main
```

Matlab program. The output should be similar to

```
Created by LL MAGAIA, Neil MULLER and Ben HERBST
Stellenbosch, June 2004

Given a sequence of  images, this computes the region of interest.

Processing! Please wait ...

Loaded 122 frames and processed in 3.37 minutes
```

At this point, the region of interest has been extracted and the frames have been saved. The next screen,

```
Created by LL MAGAIA, Neil MULLER and Ben HERBST
Stellenbosch, June 2004

Given a sequence of RGB images, this will extract background.
Reading images! Please wait ...

Now processing ...
```

At this stage the user can view the background processing. After the program finishes computing the background it will save it and automatically begin to find moving objects. Again, the user can view/display the computation process. At the end the program will show a message similar to

```
All 122 frames of moving objects shown in 1.58 minutes
```

Next, the program will copy the `my_objects_*.pgm` to the corresponding directory. At this point, the user is ready to process them with *KLT* or another *Tracker*. Note that for case of edge points, the user should run `edges.m` and then use `track_edge.c`.

Once the images are in the directory `thesis/codes/ttrack/data/merriman02` or `thesis/codes/ttrack/data/merriman02/edges`, for example, and they have been processed with the `track` program, there should be a collection of `feats_*.mat` files in the output directory `thesis/codes/ttrack/data/merriman01/feats`

or `thesis/codes/ttrack/data/merriman02/edges`
Change directory to `thesis/codes/ttrack/` and run the program

» `main` . This will call other subprograms such as
`main_corners, initialise_flow and main_ukf_optic1` to do the reconstruction.
The program should now start, displaying the following

```
Filter: Unscented Kalman Filter
Using data from [Unscented Kalman Filter] with images of size 576x720
A sequence of 20 consecutive frames selected --> from [1] to [80]
[25] number of features per frame over [80] frames
Motion[5.0e-07, 5.0e-06, 1.6e-05]
Shape [5.0e-08, 5.0e-07, 1.6e-05]

###################
rms 2d error:      0.0020669044
rms scaling error: -1.0000000000
```

or

```
Filter: Manual Filter
Using data from [Manual tracking] with images of size 576x720
A sequence of 20 consecutive frames selected --> from [1] to [20]
[25] number of features per frame over [20] frames
Motion[5.0e-07, 5.0e-06, 1.6e-05]
Shape [5.0e-08, 5.0e-07, 1.6e-05]

###############################
rms 2d error:      0.0024908123
rms scaling error: -1.0000000000
```

Note that for manual tracking the program will prompt the user to select manually the features. In this regards, the user is advised to specify in *initialise.m* the number of frames and number of features to track. The program should terminate with a result similar to

```
Distance between Cars is  14.9070 meters approximately
                        and
    the Velocity of front car is about 65.37 km/h.

          Processing time is 18.58 seconds.

     ############################################
```

```
                    Thanks!
                   ###-o-###
                Lourenco, Neil and Ben
>>
```

where the e_d value is given by the `rms 2d error` value, and the e_s value is given by the `rms scaling error` value and -1 means that in the beginning no initial error information has been given.
As can be seen, the system processed all the tasks (except KLT tracker timing) in less than six minutes which is acceptable since the programs were written in Matlab. Contrary to the KLT, the *optical flow* showed to be more robust in estimating the motion. The program should begin similar to

```
 Created by Lourenco MAGAIA, Neil MULLER and Ben HERBST
                 Stellenbosch, October 2004


   Given a video sequence, this track objects using optical flow.
                Processing! Please wait ...
   Number of pixels in the edges to skip: 1
  Tracking objects using Robust Optical Flow Method

from: 120, to: 199
max_level: 4, min_level: 1
in: ../data2/my_objects_, out: ../data2/object_out_
l1: 1.000000
l2: 1.000000
S1: 7.071068
S2: 0.707107
s1: 0.707107
s2: 0.070711
stages: 5
nx: 720
ny: 576
scale factor: 0.500000
filter: 0
omega: 1.995000
iters: 10
end: .pgm
skip: 15
allocating memory
initializing u,v,du,dv
```

```
reading first image Stage: 0
  s1: 7.071068
  s2: 0.707107

 nx 720
 nx 360
 nx 180
 nx 90
..........Max u, 0.181266, level 90
projecting (180 144) (90 72)
projecting (180 144) (90 72)
Max p_u (pre project), 0.181266, level 90
Max p2_u (post project), 0.362532, level 180
Max u, 0.000000, level 180
..........Max u, 0.296987, level 180
```

and the last part of the program should be

```
Stage: 4
  s1: 0.707107
  s2: 0.070711

 nx 720
 nx 360
 nx 180
 nx 90
..........Max u, 0.006677, level 90
projecting (180 144) (90 72)
projecting (180 144) (90 72)
Max p_u (pre project), 0.006677, level 90
Max p2_u (post project), 0.013355, level 180
Max u, 0.178167, level 180
..........Max u, 0.061001, level 180
projecting (360 288) (180 144)
projecting (360 288) (180 144)
Max p_u (pre project), 0.061001, level 180
Max p2_u (post project), 0.122002, level 360
Max u, 0.432464, level 360
..........Max u, 0.412260, level 360
projecting (720 576) (360 288)
projecting (720 576) (360 288)
```

```
Max p_u (pre project), 0.412260, level 360
Max p2_u (post project), 0.824521, level 720
Max u, 0.977597, level 720
.........Max u, 0.922622, level 720
720 576 -0.665527 0.922622 1.000000
720 576 -0.000000 3.613600 1.000000


   ######################################

   Tracking time  64.00 minutes.

   3D Reconstruction using: Unscented Kalman Filter
   Data from [Robust Optical Flow] with images of size 576x720
   A sequence of 80 consecutive frames selected --> from [1] to [80]


   [219] number of features per frame over [80] frames

   Motion[5.0e-07, 5.0e-06, 1.6e-05]
   Shape [5.0e-08, 5.0e-07, 1.6e-05]

####
   rms 2d error:      0.0010011330
   rms scaling error: -1.0000000000
          ######################################

                     Speed Analysis:

                     Object_1 -->   89.30 km/h.

                          ##o-o##

               Processing time  92.11 minutes.

          ######################################

>>
```

C.1.4 Output of the Results

The user has several ways to view the results. Using the different programs, he/she can save, print or plot the results:

- `main.m`: The main program which calls all others.
- `main_corners.m`: The sub-main program which calls programs that implement the corners tracker algorithm.
- `main_ukf_optic1.m`: The sub-main program which calls programs that perform *3D* reconstruction using the Unscented Kalman Filter.
- `buildhist0.m`: Build the region of interest given a sequence of images. For different sequences, the user should modify it in order to define the desired region.
- `buildback.m`: Program used to build a background model given a sequence of images. When the program finishes, it plots the background and saves it in the specified file, i.e. merriman01_background. This program uses histogram information. It is important to point out that this program can fail when the camera is vibrating.
- `buildback1.m`: This program can be used to build an image sequence containing only moving objects using frame by frame differences. The program is intended to be used even if the camera is vibrating.
- `back_moving.m`: Build an image sequence containing only moving objects using histogram modeling.
- `edges.m`: Find edges for a given video sequence and save them as data files, for example `my_objects_[x].edge`.
- `track.c` or `track_edge.c`: Track features for a given video sequence and save them in output directory as separate files, for example `feat_[x].mat`.
- `initialise_flow.m`: It re-initializes global variables, i.e. working directories, default constants, etc. This is necessary because each time the program performs the *3D* reconstruction it resets all variables.
- `initialise.m`: The initialise script in which most changes must be done, i.e. working directories, default constants, etc.
- `init_dir.m`: It initializes the working directories only.
- `generate_sample.m`: Another important script in which the observation data is filtered. It also displays the driveway and selected features for reconstruction.

- `setpath.m`: The setpath function, so that Matlab can easily find other programs, data directories and scripts.
- `plot_all1.m`: Call the other plotting programs and scripts/routines.
- `plot3_object.m`: Shows two graphs; the first plot shows the actual *3D* reconstructed image at every step; the second plot shows the moving objects with scattered observation at that step.
- `plot_Rot_Velocity.m`: Plots angular velocity over time.
- `plot_Trans_Velocity.m`: Plots linear velocity against time.
- `plot_coord.m`: Plots in the first subplot the speed against time and in the second subplot shows the change of coordinates against time in separate subplot. This enable us to view the convergence or stability of the system
- `plot_rot.m`: Plots the estimated rotation in terms of the quaternion values against time.
- `plot_trans.m`: Plots the estimated translation against time.
- `plot_2d.m`: Plots 2D observation and reconstructed motion against time.
- `plot_3d.m`: Plots estimated motion in *3D* against time.

C.2 Software Demonstration

All the programs are called within matlab, therefore there is no need to compile any program. But if the user wants to change some programs written in C, then the user should consult Section C.1 of this appendix for more details.

C.2.1 Programs Descriptions

The Directory *thesis/codes/DEMO* contains two sub-directories, namely *Track* and *Recon*. The program *demo.m* allows the user to select from all the examples discussed in the thesis. It will call the appropriate subprograms to track, reconstruct and visualize the results.

C.2.2 Programs in Recon Directory

The Directory *DEMO/Recon* contains many programs or scripts that execute the *3D* reconstruction using either *Singular Value Decomposition (SVD)* or *Unscented Kalman Filter (UKF)*. The directory contains the following files:

- `demo.m`: The main program which calls all others. It sets some default variables and asks the user to input the necessary parameter values:

- `fig_print` is intended for figure printing to file. If only if $fig\_print = 1$, then the figures will be printed to files (the default value is 0). In this case the user is advised to set up the figures names in printing mfile $ukf\_results\_demo.m$.
- `show_ind_rmse` can be used to print the $root\_mean\_square\_error$ of all estimated parameters if it is set up to one. Again the default value is set up to 0.
- `method` selects which experiment is intended to be processed. There are only four possible experiments to run: Cube under rotation, cube under rotation and translation, synthetic vehicle data and real data. The user is required to select the method. If the user provides incorrect parameter for three consecutive times, the program will end.
- `real_data_on` is another variable intended to determine which experiment to process (Simulation using real data set or Simulation using synthetic data). This is only valid for a pinhole camera model.
- `fs` is the font size used to print the legend on the figures.

By running this program the user has a choice of using *Singular Value Decomposition (SVD)* or *Unscented Kalman Filter (UKF)* for the *3D reconstruction algorithms*.
The mfile `demo.m` calls

- `title_demo.m` which displays the demonstration title
- `run_cube_demo.m` is the script for cube demonstration with rotation only. This calls `title_demo.m`, `make_obs.m` (generate observations), `do_recon` (do reconstruction) and it plots the results.
- `run_demo_trans.m` is the script for cube demonstration with rotation and translation. This calls `title_demo.m` and, depending on the experiment to be performed, it can call `make_obs_trans.m` (generates observations under rotation and translation), `do_recon_trans.m` (does reconstruction using SVD) or `rot_trans_demo.m` (does reconstruction using UKF) and `plot_cube.m` (plots cube results).
- `run_car_demo.m` calls routines that perform synthetic vehicles demonstration under rotation and translation.
  This calls `title_demo.m`, `vehicle_model.m`,
  `plot_veh_model.m` (plots results) and, depending on the experiment to be performed,
  it can call `make_obs_trans.m` (make observations under rotation and translation), `do_recon_trans.m` (do reconstruction using SVD) or `rot_trans_demo.m` (do reconstruction using UKF).

  - `run_real_demo.m` is the program intended for real data demonstration. It calls `rot_trans_demo.m` that performs *3D* reconstruction using *Unscented Kalman Filter* algorithm.

- `rot_trans_demo.m` is a script for UKF implementation pre-initialization. The main purpose of this program is to build the data observations using different methods. If the user calls this script directly and the data file (`real_data_on.mat`) does not exist, only the cube demonstration will be executed. `rot_trans_demo.m` calls `initialise_ukf.m`, `rotR_demo.m` (build rotation matrix), `plotcube.m`, `plot_veh_model.m` and loads experiments for real data demonstration. As the last stage it saves all necessary information to be used in `initialise_ukf.m`

  - `initialise_ukf.m` initializes the *Unscented Kalman Filter 3D* reconstruction algorithm. It begins with loading the data saved in `rot_trans_demo.m` under file `rot_cube_demo.mat`. It sets optimal *Gaussian* variables $\alpha = 1$, $\beta = 2$ and $\kappa = 0$ and then it calls `title_noise_ukf.m` where the user will set up other variables, such as Gaussian noise and covariances. We have provided some sensible default values for our experiments, but the user is free to choose any value.
  - `title_ukf.m` displays the title for *Unscented Kalman Filter* algorithm.
  - `ukf_demo.m` is the main script for *Unscented Kalman Filter 3D* reconstruction. It calls other functions such as, `ffun_n_demo.m` (observation model function), and filtered output `hfun_n_demo.m` or `hfun_demo.m` (estimation: process model function).
  - `ukf_results_demo.m` which plots the results.

- `ukf_results_demo.m`: This is the main program intended to plot all the demonstration results. It calls other plotting routines, such as `plotcube.m` and `plot_veh_model.m`. The results are saved in files, therefore we can just call it directly to see the results of the last executed experiment.

- `plotcube.m`: Plots a cube and the reconstructed cube. This is used with both Singular Value Decomposition and Unscented Kalman Filter results output.

- `ukf_veh_demo.m`: Plots a vehicle type and the reconstructed vehicle type. This can be used with both Singular Value Decomposition and Unscented Kalman Filter results.

C.2.2.1 Running a Demonstration Program

Run the demo script in Matlab in the directory *DEMO/Recon* and the program should begin as

```
        Orthographic and Pinhole Camera Models
                  3D Reconstruction Demo
                 Stellenbosch, January 2005

  Solving the Traffic Monitoring System Problem for moving objects

   1: Cube demonstration with rotation only
   2: Cube demonstration with rotation and translation
   3: Synthetic Vehicle demonstration with rotation and translation
   4: Real Data
   ..................................................
   Select:
```

If we select *Real Data* the next screen will appear,

```
        Orthographic and Pinhole Camera Models
                  3D Reconstruction Demo
                 Stellenbosch, January 2005


   Solving the Traffic Monitoring System Problem for moving objects

   1: Cube demonstration with rotation only
   2: Cube demonstration with rotation and translation
   3: Synthetic Vehicle demonstration with rotation and translation
   4: Real Data
   ..................................................
   Select: 4

   Executing Real Object.
   ......................................................

  Pressing enter will use the given default value

   We provide following real data set:

   1 - Merriman Avenue data tracked using ROF
   2 - National Higway N2 data tracked using ROF
   3 - National Road R102 data tracked using ROF
   4 - Merriman Avenue data and tracked using ROF
   5 - Merriman Avenue data and tracked using Corner Tracker
   Select real data set:
```

At this stage the user can input the required variables. Note that we provide some default values, but the user is free to input different ones. If the user does not use the defaults, the program will proceed as

```
  We need to select the UKF parameters.
  See Chapter 5 of my PhD Thesis or Kalman Filtering and Neural
  Networks by Simon Haykin, pages 228-246 for more information.

  Defaults may not work correctly for all examples !!
  Give Process noise default=1e-3, varQ = .001
  Give Measurement noise default=1e-3, varR = .001
  Give State Covariance default= 5e-2, varP = .05

  Running UKF on REAL DATA for 80 iterations.
################
```

But if we use the automatic selection, the program will proceed as

```
   Using default real data set = 3
   We need to select the UKF parameters.
   See Chapter 5 of my PhD Thesis or Kalman Filtering and Neural
   Networks by Simon Haykin, pages 228-246 for more information.

   Defaults may not work correctly for all examples !!

   Add State Covariance, default=5e-2, varP =
   Using default varP = 5.000000e-02
   Add Process noise, default=1e-3, varQ =
   Using default varQ = 1.000000e-03
   Add Measurement noise, default=1e-3, varR =
   Using default varR = 1.000000e-03

   Running UKF on REAL DATA for 80 iterations.
 ################
```

After the program finishes processing it will show a message similar to

```
  Plotting results, please wait...


  Plotting results are ok!
```

```
 ##########################
#     General Report     #
##########################

-.-.-.-.-.-.-.-.-.-.-.-.-
   Output in interval_1

Total of  1 vehicle(s) detected in the scene.

              Speed Analysis:

Maximum speed allowed in this area is  80.00 km/h.
Actual speed of the object is   69.10 km/h.

                 ##o-o##

          Thanks for using our demo!
                ...END...
```

At this point the user can visualize the ten graphs showing the parameters' estimations and object's structure (see chapter 7). Next we describe the *Track Directory*.

C.2.3 Programs in Track Directory

The Directory *DEMO/Track* contains many programs or scripts that execute the *UKF* as an estimator. The directory contains the following files:

demo.m is the main program which calls all others. It sets some default variables and asks the user to input the necessary parameter values. It sets some default variables, such as optimal *Gaussian* variables $\alpha = 1$, $\beta = 2$ and $\kappa = 0$. It is in this program where the user can change the above and other variables

- `fs` is a font size used to print legends on the image figure
- `fig_print` intended for figure printing to file. If $fig\_print = 1$, then the figures will be printed to files. In this case, the user is advised to set up the figures' names in printing $mfile\ plot\_results\_ukf.m$.
- `a` indicates which amplitude we want to use on a pre-setup curve (Simulation using non-real data).
- `real_data_on` indicates which experiment is intended to be processed. There are only two possible experiments to run (Simulation using real data set and Simulation using synthetic data).

By running this program the user has the choice of whether to run a *Simulation using real data set* or a *Simulation using synthetic data set* for estimating using *Unscented Kalman Filter*. The $mfile$ `demo.m` calls

- `title_demo.m` which displays the demonstration title only.
- `title_ukf.m` is another title script for displaying the initial message of *UKF* estimator.
- `ukf_ctrl.m`: This the main script for *Unscented Kalman Filter* implementation. The code was written by Simon J. Julier and Rudolph van der Merwe (rvdmerwe@ece.ogi.edu). It calls other function such as, `ukf_ffun_real.m` or `ukf_ffun.m` (observation model function), and filtered output `ukf_hfun_real.m` or `ukf_hfun.m` (estimation: process model function).
- `plot_results_ukf.m`: This is the main program intended to plot all the results.

C.2.3.1 Running a Demonstration Program

Run the demo script in Matlab in the directory *DEMO/Track* and the program should begin as

```
  Unscented Kalman Filter and Applications
            Object Tracking Demo
          Stellenbosch, January 2005

You can set up your own data or just use provided!
See lines 55-65 or 85-94 of <demo.m> to setup yours.

1: Simulation using real data set
2: Simulation using non-real data
.....................................
Select:
```

If we select the *Simulation using synthetic data* the next screen will appear asking the user to provide the number of frames, noise level (if not required, set this to zero) and covariance variables. The complete output should be similar to

```
  Unscented Kalman Filter and Applications
            Object Tracking Demo
          Stellenbosch, January 2005

You can set up your own data or just use provided!
See lines 112-133 or 135-144 of <demo.m> to setup yours.
```

```
 1: Simulation using real data set
 2: Simulation using synthetic data
 ....................................
 Select: 2

 We need to select the UKF parameters.
 See Chapter 5 of my PhD Thesis or Kalman Filtering and Neural
 Networks by Simon Haykin, pages 228-246 for more information.

 We add some noise (Gaussian white noise with the given variance)
 to the data

 By pressing enter will use the given default value

 Give number of frames to generate data, frames = 90
 Add data noise, default=0 noise = 0
 Add Process noise default=1e-3, varQ = .001
 Add Measurement noise default=1e-5, varR = .00001
 Add State Covariance default= 3/4, varP = 3/4

 Running UKF on Synthetic Data for 90 iterations: No noise added
##############################################
                      .......... ok ..........
>>
```

If the user opted to use all default settings, then the last output should be similar to

```
 We need to select the UKF parameters.
 See Chapter 5 of my PhD Thesis or Kalman Filtering and Neural
 Networks by Simon Haykin, pages 228-246 for more information.

 We add some noise (Gaussian white noise with the given variance)
 to the data

 By pressing enter will use the given default value

 Give number of frames to generate data, frames =
 Using default number of frames = 100
 Add data noise, default=0 noise =
 Using default data noise = 0
 Add Process noise default=1e-3, varQ =
```

```
 Using default varQ = 1.000000e-03
 Add Measurement noise default=1e-5, varR =
 Using default varR = 1.000000e-05
 Add State Covariance default= 3/4, varP =
 Using default varP = 7.500000e-01

 Running UKF on Synthetic Data for 100 iterations: No noise added
##################################
                      .......... ok ..........
>>
```

Using synthetic data with noise added, the output looks like this:

```
      Unscented Kalman Filter and Applications
             Object Tracking Demo
          Stellenbosch, January 2005

 You can set up your own data or just use provided!
 See lines 112-133 or 135-144 of <demo.m> to setup yours.

 1: Simulation using real data set
 2: Simulation using synthetic data
 ....................................
 Select: 2

 We need to select the UKF parameters.
 See Chapter 5 of my PhD Thesis or Kalman Filtering and Neural
 Networks by Simon Haykin, pages 228-246 for more information.

 We add some noise (Gaussian white noise with the given variance)
 to the data

 By pressing enter will use the given default value

 Give number of frames to generate data, frames = 90
 Add data noise, default=0 noise = .01
 Add Process noise default=1e-3, varQ =
 Using default varQ = 1.000000e-03
 Add Measurement noise default=1e-5, varR =
 Using default varR = 1.000000e-05
 Add State Covariance default= 3/4, varP =
```

```
 Using default varP = 7.500000e-01

 Running UKF on Synthetic Data for 90 iterations: Noise added
##########################################
                        .......... ok ..........

>>
```

If we run the experiment using real data the final code output would be similar to

```
      Unscented Kalman Filter and Applications
             Object Tracking Demo
          Stellenbosch, January 2005

 You can set up your own data or just use provided!
 See lines 112-133 or 135-144 of <demo.m> to setup yours.

 1: Simulation using real data set
 2: Simulation using synthetic data
 ....................................
 Select: 1
 Select real data set: <1 or 2> :2

 We need to select the UKF parameters.
 See Chapter 5 of my PhD Thesis or Kalman Filtering and Neural
 Networks by Simon Haykin, pages 228-246 for more information.

 By pressing enter will use the given default value
 Defaults may not work correctly for all examples !!

 Give Process noise default=1e-3, varQ = .001
 Give Measurement noise default=5e-2, varR = .05
 Give State Covariance default= 1e-3, varP = .001

 Running UKF on REAL DATA for 100 iterations: No noise added
##################################

                        .......... ok ..........
>>
```

Bibliography

[1] ALI, A. T. and DAGLESS, E. L., "Computer vision for automatic road traffic analysis." *In Int. Conference on Automation, Robotics and Computer Vision*, Sep 1990.

[2] ALIREZA, B.-H. and DAVID, S., "Optical Flow Calculation Using Robust Statistics." *Proc. Computer Vision and Pattern Recognition, CVPR-97, Puerto Rico*, Jun 1997, pp. 988–993.

[3] ASV, "ASV Vehicle Systems." `http://www.vehiclesystems.org.uk/`. 2002.

[4] AZARBAYEJANI, A. J., *Nonlinear Probabilistic Estimation of 3-D Geometry from Images*. PhD thesis, Massachusetts Institute of Technology, 1997.

[5] AZARBAYEJANI, A. J. and PENTLAND, A., "Recursive Estimation of Motion, Structure, and Focal Length." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jun 1995, Vol. 17, pp. 562–575.

[6] BADENAS, J. and PLA, F., "Segmentation Based on Region-Tracking in Image Sequences for Traffic Monitoring." *14th International Conference on Pattern Recognition (ICPR'98)*, Jun 1998, Vol. 2, p. 999.

[7] BAR-SHALOM, Y., RONG, L., and KIRUBARAJAN, T., *Estimation with Application to Tracking and Navigation*. John Wiley and Sons., New York, 2001.

[8] BARFOOT, T. D. and D'ELEUTERO, G. M. T., "Learning Distributed Control for an Object-Clustering Task." Tech. Rep. Technical Report, University of Toronto, Mar 2003.

[9] BARNETT, V. and LEWIS, T., *Outliers in Statistics Data, 3rd Edition*. John Wiley and Sons, Chichester, 1994.

[10] BARRON, J. L., FLEET, D. J., and BEAUCHEMIN, S. S., "Performance of Optical Flow Techniques." *International Journal of Computer Vision*, 1994, Vol. 12, pp. 43–77.

[11] BEAUCHEMIN, S. S. and BARRON, J. L., "On Discontinuous Optical Flow." *Computer and Artificial Intelligence*, 1999.

[12] BEYMER, B., MCLAUCHLAN, P. F., COIFMAN, B., and MALIK, J., "A real-time computer vision system for measuring traffic parameters." *In Int. Conference on Computer Vision and Pattern Recognition*, 1997.

[13] BEYMER, B., MCLAUCHLAN, P. F., and MALIK, J., "A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance." *Transport Research-C*, Dec 1998.

[14] BIRCHFIELD, S., "Derivation of *Kanade-Lucas-Tomasi* tracking equation." Proof of KLT-Equation, May 1997.

[15] BIRCHFIELD, S., "An implementation of the Kanade-Lucas-Tomasi feature tracker." `http://www.vision.stanford.edu/~birch/klt`. Oct 1998.

[16] BLACK, M. J., "Robust gradient method for determining optical flow." Tech. Rep. YALEU/DCS/RR-891, Yale University, Feb 1992.

[17] BLACK, M. J., *Robust Incremental Optical Flow*. PhD thesis, Faculty of Graduate School - Yale University, 1992.

[18] BLACK, M. J. and ANANDAN, P., "Constraints for the early detection of discontinuity from motion." *In National Conference on Artificial Intelligence, AAAI-90,Boston, MA*, May 1990, pp. 1060–1066.

[19] BLACK, M. J. and ANANDAN, P., "A model for detection of motion over time." *In Proc. Int. Conference on Computer Vision, ICCV-90, Osaka*, Dec 1990, Vol. 13, No. 90, pp. 33–37.

[20] BLACK, M. J. and ANANDAN, P., "Robust Dynamic Motion Estimation over Time." *Proc. Computer Vision and Pattern Recognition, Maui-Hawaii*, Jun 1991.

[21] BLACK, M. J. and ANANDAN, P., "Combining intensity and motion for incremental segmentation and tracking over long image sequence." *In Proc. of Second European Conference on Computer Vision, ECCV-92*, May 1992, Vol. 588, pp. 485–493.

[22] BOBICK, N., "Rotating Objects Using Quaternions." *Game Developer Magazine*, Feb 1998.

[23] BOGDANOV, A., WAN, E. A., CARLSSON, M., ZHANG, Y., KIEBURTZ, R., and BAPTISTA, A., "Model predictive neural control of a high-fidelity helicopter model." in *2001 AIAA Guidance, Navigation and Control Conference*, (Montreal, Quebec, Canada), Aug 2001.

[24] BOUTHEMY, P. and LALANDE, P., "Detection and tracking of moving objects based on a statistical regularization method in space and time." *In Proc. First European Conference on Computer Vision, ECCV-90, Antibes-France*, Apr 1990, pp. 307–311.

[25] BRANCO, C. and COSTEIRA, J. S., "A 3D Image Mosaicing System Using the Factorization Method.." *IEEE Intern. Symposium on Industrial Electronics*, 1998, Vol. 2, pp. 674–678.

[26] BROOKS, M., CHOJNACKI, W., and VAN DEN HENGEL, A., "Solving the Shape from Shading Problem on the CM5." *IEEE Conference on Computer Architectures for Machine Perception*, 1995.

[27] BURDEN, R. and FAIRES, J., *Numerical Analysis - sixth edition*. Brooks/Cole Publishing Company, 1997.

[28] CALWAY, A. and MAYOL-CUEVAS, W., "Real-time camera tracking using particle filtering." Research Project: Mobile and Wearable Computing and Computer Vision, 2005.

[29] CHANN, R., "Recursive estimation of 3-D motion and structure in image sequences based on measurement transformations." Master's thesis, Queen's University, Kingston, Ontario, Canada, Sep 1994.

[30] CHAU, T., LAU, B., and PARK, Y., "Vehicle Detection and Classification in Shadowy Traffic Images Using Wavelets and Neural Networks." *Transportation Sensors and Control: Collision Avoidance, Traffic Management and ITS*, Dec 1996, Vol. 2902, pp. 136–147.

[31] CHOUKROUN, D., BAR-ITZHACK, I., and OSHMAN, Y., "A Novel Quaternion Kalman Filter." in *In Proceedings of the 42th AIIA Guidance, Navigation, and Control Conference, Montery*, (Haifa 32000 Israel), Aug 2002.

[32] CILLIERS, C., "Design and Implementation of a Digital Modem on the DSP56001." Master's thesis, University of Stellenbosch, 1994.

[33] CITILOG, "Video Detection Systems." www.citilog.com. Jun 2005. 64, Rue du Dessous des Berges 75013, Paris, France.

[34] COIFMAN, B., "Improved Velocity Estimation Using Single Loop Detectors." *Transportation Research: Part A*, 2001, Vol. 35, pp. 863–880.

[35] COIFMAN, B. and BEYMER, D., "A real-time computer vision system for vehicle tracking and traffic surveillance." *Transportation and research: Part C*, Dec 1998, Vol. 6, No. 4, pp. 271–288.

[36] COIFMAN, B. and DHOORJATY, S., "Event Data Based Traffic Detector Validation Tests." *ASCE Journal of Transportation Engineering*, 2004, Vol. 130, pp. 313–321.

[37] COIFMAN, B. and ERGUETA, E., "Improved Vehicle Re-identification and Travel Time Measurement on Congested Freeways." *ASCE Journal of Transportation Engineering*, 2003, Vol. 129, pp. 475–483.

[38] COIFMAN, B. and YANG, Y., "Estimating Spatial Measures of Roadway Network Usage from Remotely Sensed Data." *Transportation Research Record 1870*, 2004, Vol. 35, pp. 133–138.

[39] CORPORATION, N. S., "Introduction to the Sampling Theory." *National Semiconductor Corporation*, Jan 1980. Application Note 236.

[40] COSTEIRA, J. S., *Multi-body factorization for Motion Analysis*. PhD thesis, Instituto Superior Técnico, 1995.

[41] CRUZ, B., ALI, A. T., and DAGLESS, E. L., "A temporal smoothing technique for real-time motion detection." *In CAIP*, Sep 1993.

[42] DAHRLQUIST, G. and BJORCK, A., *Numerical Methods*. Prentice Hall, Inc, 1974.

[43] DAILEY, D., CATHEY, F., and PUMRIN, S., "An Algorithm to Estimate Mean Traffic Speed Using Ucalibrated Cameras." *IEEE Transactions Intelligence Transportation Systems*, Jun 2000, Vol. 1, No. 2.

[44] DE MOOR, B., "Sustainability Effects of Traffic Management Systems." tech. rep., Belgian Public Planning Service Science Policy, Rue de la Science 8 - Wetenschapsstraat, B-1000 Brussels, January 2003.

[45] DEMMEL, J., "Applied Numerical Linear Algebra." *SIAM*, 1997.

[46] ERRIKSSON, A., "3-D Face Recognition." Master's thesis, Department of Electrical and Electronic Engineering, The University of Stellenbosch, South Africa, Dec 1999.

[47] FAUGERAS, O., *Three-Dimensional Computer Vision*, pp. 144–146. Cambridge, Massachusetts, London, England: The MIT Press, 1993.

[48] FAUGERAS, O., *Three-Dimensional Computer Vision - third edition*. The MIT Press, 1999.

[49] FERRYMAN, J. M., WORRALL, A. D., and MAYBANK, S. J., "Learning Enhanced 3D Models for Vehicle Tracking." *In Proceedings on British Machine Learning Conference*, 1998, pp. 871–882.

[50] FORSYTH, D., IOFFE, S., and HADDON, J., "Bayesian Structure from Motion." *In Proc. Int. Conference on Computer Vision (ICCV-99)*, 1999, pp. 660–665.

[51] FUKUNAGA, K., *Introduction to Statistical Pattern Recognition, second edition*. Academic Press, Inc, 1990.

[52] GOLUB, G. and VAN LOAN, C., *Matrix Computation - second edition*. The John Hopkins University Press, 1989.

[53] GONZALEZ, R. and WOODS, R., *Digital Image Processing*. Addison-Wiley Publishing Company, 1990.

[54] HAGER, G. and BELHUMEUR, P., "Efficient Region Tracking With Parametric Models of Geometry and Illumination." in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1025–1039, 1998.

[55] HAMPEL, F. R., RONCHETTI, E. M., ROUSSEEUW, P. J., and STAHEL, W. A., *Robust Statistics: The Approach Based on Influence Functions*. John Wiley and Sons., New York, 1986.

[56] HAN, M. and KANADE, T., "Scene Reconstruction from Multiple Uncalibrated Views." Tech. Rep. CMU-RI-TR-00-09, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Jan 2000.

[57] HARRIS, C. and STEPHENS, M., "A combined corner and edge detector." *In Proceedings 4th Alvey Vision Conference, Manchester"*, 1998, pp. 147–151.

[58] HARTLEY, R. and ZISSERMAN, A., *Multiple view in Computer Vision*. Cambridge University Press, 2000.

[59] HAYKIN, S., *Adaptive filter theory*, pp. 302–337. Upper Saddle River, New Jersey: Prentice Hall, Inc., 1996.

[60] HAYKIN, S., *Kalman Filtering and Neural Networks*. John Wiley and Sons, New York, 2001.

[61] HEEMAN, P. and DAMNATI, G., "Deriving Phrase-based Language Models." in *IEEE Workshop on Speech Recognition and Understanding*, (Santa Barbara, CA), pp. 41–48, Dec 1997.

[62] HOOSE, N., *Computer Image Processing in Traffic Engineering*. ohn Wiley and Sons, Inc., 1991.

[63] HOOSE, N., "Development and Implementation of the IMPACTS Traffic Image Processing System.." *World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems.*, 1995, Vol. 3.

[64] HOSOM, J. P. and COLE, R. A., "A Diphone-Based Digit Recognition System using Neural Networks." in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, (Munich, Germany), pp. 3369–3372, Apr 1997.

[65] HUBER, P., *Robust Statistics*. John Wiley and Sons, New York, 1981.

[66] INIGO, R., "Traffic monitoring and control using machine vision: A survey.." *IEEE Transactions on Industrial Electronics*, Aug 1985, Vol. 32, No. 3, pp. 177–185.

[67] JAHNE, B., *Digital Image Processing: Concepts, Algorithms and Scientific Applications-third edition*. Springer Verlag, 1995.

[68] JEBARA, T., AZARBAYEJANI, A. J., and PENTLAND, A., "3D Structure from 2D Motion." *IEEE Signal Processing Magazine, "3D And Stereoscopic Visual Communication"*, May 1999, Vol. 16, No. 3.

[69] JULIER, S., "The Scaled Unscented Transformation." *IEEE Journal Automatica*, Feb 2000, Vol. 36, pp. 1627–1638.

[70] JULIER, S. and UHLMANN, J., "A General Method for Approximating Nonlinear Transformations of Probability Distributions." Tech. Rep. RRG, Robotics Research Group, Dept. of Engineering Science, University of Oxford, Nov 1996.

[71] JULIER, S. and UHLMANN, J., "A New Extension of the Kalman Filter to Nonlinear Systems." *In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

[72] JULIER, S., UHLMANN, J., and DURRANT-WHYTE, H., "A New Approach for Filtering Nonlinear Systems." *Proceedings of the 1995 American Control Conference*, 1995, pp. 1628–1632.

[73] KALMAN, R., "A New Approach to Linear Filtering and Prediction Problems." *Transactions of the ASME - Journal of Basic Engineering*, 1960, Vol. 82, pp. 35–45.

[74] KAMBHATLA, N., *Local Models and Gaussian Mixture Models for Statistical Data Processing*. PhD thesis, Department of Computer Science and Engineering, Oregon Graduate Institute, 1996.

[75] KANADE, T. and TOMASI, C., "Shape and Motion Without Depth." *International Conference of Computer Vision*, 1990.

[76] KELLY, D., "Results of field trial of the IMPACTS image processing system for Traffic Monitoring." in *In 6th Int. Conference on Road Traffic Monitoring*, pp. 137–142, Apr 1992.

[77] KINGSBURY, N. and MARGAREY, J., "Wavelet Transform in Image Processing." Tech. Rep. DSTO-TR-1632, University of Cambridge, 1997.

[78] KOLLER, D., HUANG, T., and MALIK, J., "Towards robust automatic traffic scene analysis in real-time." *In Int. Conference on Pattern Recognition*, Oct 1994, pp. 126–131.

[79] KOLLER, D., WEBER, J., and MALIK, J., "Towards real-time visual based tracking in clustered traffic scenes." *In Intelligent Vehicles Symposium*, 1994, pp. 201–206.

[80] KOLLER, D., DANIILIDIS, K., and NAGEL, H., "Model-Based object tracking in traffic monocular image sequence of road traffic scenes." *In European Conference on Computer Vision*, 1993, Vol. 10, pp. 257–281.

[81] KOLLER, D., DANIILIDIS, K., THORHALLSON, T., and NAGEL, H., "Model-based object tracking in traffic scenes." *In European Conference Computer Vision*, May 1992, pp. 437–452.

[82] KOLLER, D., WEBER, J., and MALIK, J., "Robust multiple car tracking with occlusion reasoning." *In European Conference on Computer Vision*, 1994.

[83] LAMPORT, L., LaTeX—*A Document Preparation System*. Reading, Massachusetts: Addison-Wesley, 1994.

[84] LORIO, B., "Towards a non-intrusive traffic surveillance system using digital image processing." Master's thesis, Department of Civil Engineering, University of Stellenbosch, South Africa, Sep 2001.

[85] LOU, J. and TAN, T., "Model-Based Vehicle Tracking." *IEEE Transactions on Image Processing*, Oct 2005, Vol. 14, No. 10, pp. 1561–1569.

[86] LUCAS, B. and KANADE, T., "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, Apr 1981.

[87] LUONG, Q. T., DERICHE, R., FAUGERAS, O., and PAPADOPOULO, T., "On determining the fundamental matrix: analysis of different methods and experimental results." Tech. Rep. RR-1894, INRIA, Sophia Antipolis, France, 1993.

[88] LUONG, Q. T. and FAUGERAS, O., "The Fundamental Matrix: Theory, Algorithms, and Stability Analysis." *International Journal of Computer Vision*, 1995.

[89] MAGAIA, L. L. and FOURIE, B., "South african road law enforcement systems." Training notes on Law Enforcement in Stellenbosch, Sep 2005.

[90] MAGAIA, L. L. and HERBST, B. M., "The Singular Value Decomposition and Factorization Method." *In Proceedings of Pattern Recognition Association of South Africa, PRASA*, 2000.

[91] MAGAIA, L. L., MULLER, N. L., and HERBST, B. M., "A Video-Based Traffic Monitoring System." *In Proceedings of Pattern Recognition Association of South Africa, PRASA*, 2005.

[92] MALIK, J., WEBER, J., LUONG, Q. T., and KOLLER, D., "Smart cars and smart roads." *In Proceedings 6th British Machine Vision Conference*, 1995, pp. 367–381.

[93] MANOLIS, M. T. and JEFFREY, F. N., "On the Performance of Object Clustering Techniques." Tech. Rep. Technical Report 1090, University of Wisconsin-Madson, May 1992.

[94] MARGAREY, J., *Motion Estimation using Complex Wavelets*. PhD thesis, Department of Engineering, 1997.

[95] MATSUBARA, M., MASANORI, A., and DAY, D., "Development of a New Multi-Purpose Vehicle Detector and its Implementation in the Tokyo Metropolitan Traffic Control System." *The Proceeding of the 1994 Annual Meeting of IVHS America.*, 1994.

[96] MAYBECK, P., *Stochastic models, estimation, and control, Volume 1*. Academic Press, 1979.

[97] MICHALOPOULOS, P., "Vehicle detection through image processing. The Autoscope system.." *IEEE Transactions on Vehicular Technology*, 1991, Vol. 40, pp. 21–29.

[98] MILAN, S., HLAVAC, H., and BOYLE, B., "Image Processing." *Analysis and Machine Vision*, 1993, pp. 510–512.

[99] MOHINDER, S. G. and ANDREWS, A., *Kalman Filtering. Theory and Practice using Matlab*. John Wiley and Sons, Inc, 2001.

[100] MUKUNDAN, R., "Quaternions: From Classical Mechanics to Computer Graphics, and Beyond." in *In Processing of the 7th Asian Technology Conference in Mathematics*, (Christchurch, New Zeland), May 2001.

[101] MULLER, N. L., "Image Recognition Using the Eigenpicture technique." Master's thesis, Department of Mathematics and Applied Mathematics, The University of Cape Town, South Africa, Jun 1998.

[102] MULLER, N. L., *Facial Recognition, Eigenfaces and Synthetic Discriminant Functions*. PhD thesis, Department of Applied Mathematics, Stellenbosch, South Africa, Nov 2000.

[103] MULLER, N. L., MAGAIA, L. L., and HERBST, B. M., "The Singular Value Decomposition and Image Processing." *SIAM Review*, Sep 2001, Vol. 46, No. 3, pp. 518–545.

[104] NELSON, A. and WAN, E. A., "Neural Speech Enhancement Using Dual Extended Kalman Filtering." in *Proceedings of the International Conference on Neural Networks*, vol. 4, pp. 2171–2175, IEEE, Jun 1997.

[105] NELSON, A. and WAN, E. A., "A Two-Observation Kalman Framework for Maximum-Likelihood Modeling of Noisy Time Series." in *Proceedings of the International Joint Conference on Neural Networks*, IEEE,INNS, May 1998.

[106] NIHAN, N., LETH, M., and WONG, A., "Video Image Processing for freeway Monitoring and Control: Evaluation of the Mobilizer." Tech. Rep. TNW 95-03, University of Washington, Nov 1995.

[107] PAPENFUS, A., *Traffic Counting Strategy for Rural Roads*. PhD thesis, Faculty of Engineering, University of Pretoria, 1992.

[108] PARAGIO, N. and DERICHE, R., "Unifying Boundary and Region-Based Information for Geodesic Active." in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, vol. 2, p. 2300, 1999.

[109] PENIO, S., *Local feature Analysis: A Statistical Theory for Information, Representation and Transmission*. PhD thesis, Laboratory of Computational Neuroscience, 1998.

[110] PERVIN, E. and WEBB, J., "Quaternions for Computer Vision and Robotics." *Computer Vision and Pattern Recognition*, 1983, pp. 382–383.

[111] PETER, H., GAILE, G., and PETER, G., *Two- and Three-Dimensional Pattern of the Faces - Third Edition*. A K Peters, Natick, Massachusetts, 1999.

[112] PITMAN, B. and TENNE, D., "Tracking a Convoy of Ground Vehicles." tech. rep., The State University of New York at Buffalo, Jan 2002.

[113] POELMAN, C. and KANADE, T., "A Para-perspective Factorization Method for Shape and Motion Recovery." Tech. Rep. CMU-CS-93-219, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Dec 1993.

[114] POSKANZER, J., "Portable graymap file format." http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/PGM.txt. 1991.

[115] PUPILLI, M. and CALWAY, A., "Real-Time Camera Tracking Using a Particle Filter." *In Proceedings of British Machine Vision Conference*, Sep 2005, pp. 519–528.

[116] REINHARD, K., SCHLUENS, K., and KOSCHAN, A., *Computer Vision: Three-Dimensional Data from Images*. Springer-Verlag Singapore, 1998.

[117] RITTSCHER, J., KATO, J., JOGA, S., and BAKE, A., "A Probabilistic Background Model for Tracking." in *In European Conference on Computer Vision*, (Department of Engineering Science), 2000.

[118] ROBERT, G. and PATRICK, H., *Introduction to Random Signal and Applied Kalman Filtering*. John Wiley and Sons., New York, 1992.

[119] ROBERT, T., ALAN, L., and KANADE, T., "Introduction to the Special Section on Video Surveillance." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Aug 2000, Vol. 22, No. 8.

[120] SADT, "Moving South Africa (The Action Agenda): A 20 year Strategic Framework for Transport in South Africa." http://www.transport.gov.za/projects/msa/action-agenda-may99/actionagenda01.html. May 1999. South African Department of Transport.

[121] SAMARAS, D., METAXAS, D., FUA, D., and LECLERC, P., "Variable Albedo Surface Reconstruction from Stereo and Shape from Shading." *IEEE Computer Vision and Pattern Recognition Conference*, 2000.

[122] SAMUEL, B. and POPOLI, R., *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

[123] SANRAL, *Traffic Counting Yearbook 2004*. South African National Road Agency Limited., Pretoria, Mar 2005.

[124] SCHALKWYK, J., COLTON, D., and FANTY, M., "The CSLUsh Toolkit for Automatic Speech Recognition." Tech. Rep. CSLU–011–1995, Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology, P.O. Box 91000, Portland, Oregon 97291-1000 USA, Dec 1995.

[125] SCHMID, P. and BARNARD, E., "Robust, N-best Formant Tracking." in *Proceedings of the Fourth European Conference on Speech Communication and Technology*, (Madrid, Spain), ESCA, Sep 1995.

[126] SCLAROFF, S. and PENTLAND, A., "Modal Matching for Correspondence and Recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jun 1995, Vol. 17, No. 6, pp. 545–561.

[127] SETCHELL, C. J., *Application of Computer Vision to Road-Traffic Monitoring*. PhD thesis, Department of Electrical and Electronic Engineering, Sep 1997.

[128] SHAOGANG, G., STEPHEN, M., and ALEXANDRA, P., *Dynamic Vision: From Images to Face Recognition*. Imperial College Press, 2000.

[129] SOATTO, S. and PERONA, P., "Reducing Structure From Motion, a General Framework for Dynamic Vision part 1: Modeling." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Dec 1995.

[130] SOATTO, S. and PERONA, P., "Reducing Structure From Motion, a General Framework for Dynamic Vision part 2: Experimental Evaluation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Dec 1995.

[131] SOATTO, S. and PERONA, P., "Reducing Structure from Motion: A general framework for dynamic vision." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Sep 1998, Vol. 20, pp. 933–960.

[132] SORENSON, H. (Ed.), *Kalman Filtering: Theory and Application*. New York: IEEE Press, 1985.

[133] SOURABH, A. and ADELSON, H., "Analyzing and Recognizing Walking Figures in XYT." Tech. Rep. 223, MIT Media Laboratory, 1994.

[134] SROM, J., JEBARA, T., BASU, S., and PENTLAND, A., "Real Time Tracking and Modeling of Face: An EKF-Based Analysis by Synthesis Approach." Tech. Rep. Perceptual Computing Technical Report number 506, MIT Media Laboratory, 1999.

[135] STEWART, J. and LANGER, M., "Towards Accurate Recovery of Shape from Shading under Diffuse Lighting." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Sep 1997, Vol. 19, pp. 1020–1025.

[136] TAKABA, S., "Measurement of traffic flow using real-time processing of moving pictures." *In 32rd Conference on Vehicular Technology*, 1982, pp. 488–494.

[137] TARR, M. and BLACK, M. J., "A computational and evolutionary perspective on the role of representation in computer vision." Tech. Rep. YALEU/DCS/RR-899, Yale University, Oct 1991.

[138] TIERNEY, J., RADER, C., and GOLD, B., "Digital Frequency Synthesizers." *IEEE on Audio and Electroacoustics*, Mar 1971, Vol. AU-19, pp. 48–57.

[139] TOMASI, C. and KANADE, T., "Shape and Motion from Image Stream: a Factorization Method.." Tech. Rep. Technical Report CS-90-166, Carnegie Mellon University, 1991.

[140] TOMASI, C. and KANADE, T., "Shape and Motion from Image Stream: a Factorization Method - Detection and Tracking of Points Features." Tech. Rep. Technical Report CS-91-132, Carnegie Mellon University, 1991.

[141] TOMASI, C. and KANADE, T., "Shape and Motion from Image Stream under Orthography." *International Journal Of Computer Vision*, 1992, Vol. 9, pp. 137–154.

[142] TOSHIHIKO, M. and KANADE, T., "A Sequential Factorization Method for Recovering Shape and Motion from Image Streams." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Aug 1997, Vol. 19, pp. 858–867.

[143] TREFETHEN, L. and BAU, D., "Numerical Linear Algebra." *SIAM Press*, 1997.

[144] TRUCO, E. and VERRI, A., *Introductory techniques for 3D computer vision*. Prince Hall, New Jersey, 1998.

[145] VAN DER MERWE, R., DE FREITAS, D., and WAN, E. A., "The Unscented Particle Filter." in *Advances in Neural Information Processing Systems (NIPS13)* (T. K. LEEN, T. G. D. and TRESP, V. (Eds)), MIT Press, Dec 2000.

[146] VAN DER MERWE, R. and WAN, E. A., "The Square-Root Unscented Kalman Filter for State and Parameter-estimation." in *International Conference on Acoustics, Speech and Signal Processing*, (20000 NW Walker Road, Beaverton, Oregon), May 2002.

[147] VAN DER MERWE, R. and WAN, E. A., "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models." in *Proceedings of the Workshop on Advances in Machine Learning*, (Montreal, Canada.), Jun 2003.

[148] VAN NIEKERK, K., EDWARDS, C., and URBAN, P., "Special Traffic Surveys." Tech. Rep. CR112, Provincial Administration of the Cape of Good Hope, Mar 1994.

[149] VEMPALA, A. and BLUM, A., "Geometric Tools for Algorithms." Tech. Rep. CMU-CS-97-167, Carnegie Mellon University, Aug 1997.

[150] VENTER, C., "Structure form Motion estimation using a Nonlinear Kalman Filter." Master's thesis, Department of Electrical and Electronic Engineering, University of Stellenbosch, South Africa, Feb 2002.

[151] VICCI, L., "Quaternion and Rotation in 3-Space: The Algebra and its Geometric Interpretation." Tech. Rep. TR01-014, Microelectronic Systems Laboratory, University of North Carolina, Chapel Hill, North Carolina, USA, Aug 2000.

[152] WAN, E. A. and BOGDANOV, A., "Model predictive neural control with applications to a 6 DoF helicopter model." in *American Controls Conference*, (Arlington, VA), Jun 2001.

[153] WAN, E. A. and NELSON, A., *Kalman Filtering and Neural Networks*, Ch. Dual EKF Methods. Wiley Publishing, 2001.

[154] WAN, E. A. and VAN DER MERWE, R., "The Unscented Kalman Filter for Nonlinear Estimation." *In Proceedings of IEEE Symposium 2000 (AS-SPCC)*, Oct 2000.

[155] WAN, E. A. and VAN DER MERWE, R., *Kalman Filtering and Neural Networks*, Ch. Chapter 7 : The Unscented Kalman Filter, (50 pages). Wiley Publishing, 2001.

[156] WAN, E. A., VAN DER MERWE, R., and NELSON, A., "Dual Estimation and the Unscented Transformation." *Advances in Neural Information Processing Systems 12*, 2000, pp. 666–672.

[157] WARD, K. and HEEMAN, P., "Acknowledgments in Human-Computer Interaction." in *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, (Seattle), pp. 280–287, May 2000.

[158] WEI, G.-Q. and HIRZINGER, G., "Parametric Shape-from-Shading by Radial Basis Functions." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Apr 1997, Vol. 19, No. 4.

[159] WEI, W., LEEN, T., and BARNARD, E., "A Fast Histogram-Based Post-processor that Improves Posterior Probability Estimates." *Neural Computation*, 1996, Vol. 11.

[160] WELCH, G. and BISHOP, G., "An Introduction to the Kalman Filter." `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`. Dec 1997.

[161] WETTELAND, T. and LUNDEBYE, S., "Third African Road Safety Congress." www.worldbank.org/transport/roads/saf_docs/finance.pdf. Apr 1997. Financing of Road Safety Actions, Pretoria.

[162] WHEELER, M. and IKEUCHI, K., "Iterative Estimation of Rotation and Translation using the Quaternion." Tech. Rep. CMU-CS-95-215, Computer Science Department, Carnegie Mellon University, Dec 1995.

[163] XENOPHON, P. and PETER, B., "Estimation of motion boundary location and optical flow using dynamic programing." Tech. Rep. 9607, Yale University New Haven, Jun 1996.

[164] YILMAR, A., LI, X., and SHAH, M., "Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Camera." in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1531–1536, 2004.

[165] YU, C., HARBURY, A., and MUIR, S., "A Fiber Optic Intersection Traffic Control System Based on a Competitive Cumulative Momentum Model." Tech. Rep. DTRS93-G-0018, North Carolina Agricultural and Technical State University, Apr 2003.

[166] YU, L. and DYER, C., "Observer Motion Estimation and Control from Optical Flow." in *International Conference on Image Processing*, (Computer Science Department), 2001.

[167] ZHAO, L., "Recursive estimation of 3-D Motion Trajectories from image sequences using measurements of feature point positions and optical flow." tech. rep., Queen's University, Kingston, Ontario, Canada, May 1998.