Binocular Ego-Motion Estimation for Automotive Applications

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik
der Goethe Universität
in Frankfurt am Main

von

Hernán Badino

aus Oncativo, Córdoba, Argentinien

Frankfurt 2008

vom Fachbereich Informatik der Goethe Universität als Dissertation angenommen.

Dekan: Prof. Dr.-Ing. Detlef Krömker.

Gutachter: Prof. Dr.-Ing. Rudolf Mester and Prof. Dr.-Ing. Reinhard Koch.

Datum der Disputation: 20.10.2008.

Acknowledgments

First of all I would like to thank my advisor, Dr. Uwe Franke for his unconditional support and valuable advice since even before the start of this thesis. He provided me with the necessary freedom to carry out this work and a open-minded and friendly atmosphere in an optimal work environment.

Thanks to Prof. Dr. Rudolf Mester for supervising my dissertation. He helped me with fruitful discussions and supported me in managing the administrative issues.

Thanks to Prof. Dr. Reinhard Koch for examining my dissertation.

Very special thanks to Dr. Stefan Gehrig for the very productive discussions and advices and for proof-reading this work.

I would also like to thank Tobi Vaudrey for the proof-reading this thesis.

Thanks to Clemens Rabe for the unbelievable software engineering support and for the implementation of pieces of this thesis.

I would also like to express my gratitude to Stefan Hahn and Hans-Georg Metzler at Daimler Research for maintaining the project in which this thesis was developed.

Thanks to Dr. Fridtjof Stein, Dr. Jens Klappstein and Dr. Carsten Knöppel for proof-readings my papers.

I would like to thank my parents for supporting me and my academic education.

At last, but not least, I would like to thank my wife Vina for her eternal support and motivation.

©by Hernán Badino

Contents

Deutsche Zusammenfassung der Dissertation **v**

Abstract **xiii**

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives of the Dissertation 2

1.3 Contributions of the Dissertation 4

1.4 Dissertation Overview 5

2 Image Geometry and the Correspondence Problem **7**

2.1 Introduction 7

2.2 Image Formation and Camera Geometry 7

2.2.1 Image 7

2.2.2 Thin Lenses and Pinhole Camera 8

2.2.2.1 Thin Lens Model 8

2.2.2.2 Ideal Pinhole Camera 9

2.2.2.3 Frontal Pinhole Camera 10

2.2.2.4 Field of View 11

2.2.2.5 Camera and Image Coordinate System 11

2.3 Geometry of Two Views 12

2.3.1 Epipolar Geometry 13

2.3.2 Standard Stereo Configuration 14

2.3.3 Calibration and Rectification 15

2.4 Image Primitives and Correspondence 17

2.4.1 Translational Motion Model 17

2.4.2 Affine and Projective Motion Models 21

2.5 Literature 21

3 Overview of the Proposed Approach **22**

3.1 Introduction 22
3.2 Problem Statement 22
3.3 Sensors for Ego-motion Computation 24
3.4 Proposed Approach 24
  3.4.1 The *Chicken-And-Egg* Problem 26
  3.4.2 The Positive Feedback Effect 26

4 Kalman Filter-based Estimation of 3D Position and 3D Velocity 28
4.1 Introduction 28
4.2 Literature Review on 3D Object Tracking 28
  4.2.1 Literature Based on Kalman Filters 29
  4.2.2 Alternative Methods for Object Tracking. 31
4.3 The Kalman Filter Model 32
  4.3.1 Stochastic Models and Kalman Filters 33
  4.3.2 Continuous Motion Model of 3D Position and 3D Velocity . 34
    4.3.2.1 Camera Motion Model 34
    4.3.2.2 Object Motion Model 35
    4.3.2.3 Continuous System Model 35
  4.3.3 Discrete System Model 36
    4.3.3.1 Transition Matrix A 36
    4.3.3.2 System Input Matrix B 37
    4.3.3.3 System Covariance Matrix Q 37
    4.3.3.4 Summary of the Discrete System Model 38
  4.3.4 Measurement Model 39
  4.3.5 The Extended Kalman Filter Equations 41
4.4 Initialization of the Filter and the Cramér-Rao Lower Bound 42
  4.4.1 Initialization of the Filter 42
  4.4.2 Comparison with Optimal Unbiased Estimator 44
4.5 Simulation Results 44
4.6 Summary 53

5 The Absolute Orientation Problem 54
5.1 Introduction 54
5.2 Literature Review 54
5.3 Preliminaries 55
  5.3.1 Introduction to Least Squares 55
  5.3.2 The Absolute Orientation Problem 57
5.4 Weighted Least Squares Formulation 58

5.4.1 Solution by Singular Value Decomposition 60
5.4.2 Solution by Polar Decomposition 60
5.4.3 Solution by Rotation Quaternions 61
5.5 Matrix Weighted Total Least Squares Formulation 61
5.6 Summary 64

6 Modeling Error in Stereo Triangulation 65
6.1 Introduction 65
6.2 Hexahedral Model 65
6.3 Egg-Shaped Ellipsoidal Model 67
6.4 Ellipsoidal Model 67
6.5 Biased Estimation of 3D Position 69
6.6 Summary 74

7 Robust Real-Time 6D Ego-Motion Estimation 76
7.1 Introduction 76
7.1.1 Organization of the Chapter 77
7.2 Literature Review on Ego-Motion Estimation 77
7.2.1 Monocular methods 78
7.2.2 Multi-ocular methods 80
7.2.2.1 Methods based on Stereo and Optical Flow 80
7.2.2.2 Methods based on Stereo and Normal Flow 83
7.2.3 Fusion of Multiple Sensors for Ego-Motion and Positioning Estimation 83
7.2.4 Summary of the Literature Review 84
7.3 Overview of the Algorithm 84
7.3.1 Motion Representation with Matrices 85
7.4 Smoothness Motion Constraint 87
7.4.1 Introduction 87
7.4.2 SMC for Weighted Least Squares 88
7.4.3 SMC for Total Least Squares 89
7.4.4 Discussion of the Scalar and Matrix SMC 90
7.4.5 Generation of Simulated Data 91
7.4.6 Simulation Results 93
7.5 Multi-Frame Estimation 95
7.5.1 Introduction 95
7.5.2 Integration of Multiple-Frames 102
7.5.3 Simulation Results for MFE 106

7.6 Integration of Filtered Data 107
7.7 Integration with Inertial Sensors 111
7.8 Summary 114

8 Experimental Results 116
8.1 Introduction 116
    8.1.1 Optical Flow and Stereo Implementation 116
8.2 Traffic Scenarios 117
    8.2.1 Sequence *Curves* 118
    8.2.2 Sequence *Ring* 121
    8.2.3 Computation Times 123
        8.2.3.1 Absolute Orientation Computation 123
        8.2.3.2 Full cycle times 123
8.3 Off-Road and Country Scenarios 123
8.4 Indoor Environment 125
8.5 Pose-Estimation for Crash Test Analysis 129

9 Conclusions and Outlook 134
9.1 Summary 134
9.2 Conclusions 135
9.3 Future Work 135
9.4 Outlook 136

A Quaternions 137

B Abbreviations and Symbols 141

Bibliography 143

Deutsche Zusammenfassung der Dissertation „Binokulare Eigenbewegungsschätzung für Fahrerassistenzanwendungen"

Einführung

Autofahren kann gefährlich sein. Die Fahrleistung wird durch die physischen und psychischen Grenzen des Fahrers und durch externe Faktoren wie das Wetter beeinflusst. Fahrerassistenzsysteme erhöhen den Fahrkomfort und unterstützen den Fahrer, um die Anzahl an Unfällen zu verringern. Fahrerassistenzsysteme unterstützen den Fahrer durch Warnungen mit optischen oder akustischen Signalen bis hin zur Übernahme der Kontrolle über das Auto durch das System.

Eine der Hauptvoraussetzungen für die meisten Fahrerassistenzsysteme ist die akkurate Kenntnis der Bewegung des eigenen Fahrzeugs. Heutzutage verfügt man über verschiedene Sensoren, um die Bewegung des Fahrzeugs zu messen, wie zum Beispiel GPS und Tachometer. Doch Auflösung und Genauigkeit dieser Systeme sind nicht ausreichend für viele Echtzeitanwendungen. Die Berechnung der Eigenbewegung aus Stereobildsequenzen für Fahrerassistenzsysteme, z.B. zur autonomen Navigation oder Kollisionsvermeidung, bildet den Kern dieser Arbeit.

Diese Dissertation präsentiert ein System zur Echtzeitbewertung einer Szene, inklusive Detektion und Bewertung von unabhängig bewegten Objekten sowie der akkuraten Schätzung der sechs Freiheitsgrade der Eigenbewegung. Diese grundlegenden Bestandteile sind erforderlich, um viele intelligente Automobilanwendungen zu entwickeln, die den Fahrer in unterschiedlichen Verkehrssituationen unterstützen. Das System arbeitet ausschließlich mit einer Stereokameraplattform als Sensor.

Lösungsansatz

Das vorgestellte System gliedert sich in drei wesentliche Bausteine.

Die „Registrierung von Bildmerkmalen" erhält eine Folge rektifizierter Bilder als Eingabe und liefert daraus eine Liste von verfolgten Bildmerkmalen mit ihrer entsprechenden 3D-Position.

Der Block „Eigenbewegungsschätzung" besteht aus vier Hauptschritten in einer Schleife: Bewegungsvorhersage, Anwendung der Glattheitsbedingung für die Bewegung (GBB), absolute Orientierungsberechnung und Bewegungsintegration. Die GBB ist eine mächtige Bedingung für die Ablehnung von Ausreißern und für die Zuordnung von Gewichten zu den gemessenen 3D-Punkten. Die absolute Orientierung wird mit der Methode der kleinsten Quadrate in geschlossener Form geschätzt. Jede Iteration stellt eine neue Bewegungshypothese zur Verfügung, die zu der aktuellen Bewegungsschätzung integriert wird. Wir nennen diese Schätzung Multiframeschätzung im Gegensatz zur Zweiframeschätzung, die nur die aktuellen und vorherigen Bildpaare für die Berechnung der Eigenbewegung betrachtet.

Der dritte Block besteht aus der iterativen Schätzung von 3D-Position und 3D-Geschwindigkeit von Weltpunkten. Hier wird eine Methode basierend auf einem Kalman Filter verwendet, das Stereo, Featuretracking und Eigenbewegungsdaten fusioniert.

Iterative Schätzung von 3D-Position und 3D-Geschwindigkeit

Ein stochastisches Modell für die rekursive Schätzung der 3D-Position und 3D-Geschwindigkeit von Weltpunkten wird präsentiert. Das Kalman Filter ist ein mathematisches Werkzeug, das rekursiv den Zustand eines dynamischen Systems mit verrauschten Messdaten schätzt. Unter bestimmten Annahmen sind Kalman Filter optimale Schätzer, in dem Sinne, dass sie die Unsicherheit der Schätzung minimieren. Viele Methoden, die auf einem Kalman Filter basieren, sind vorgeschlagen worden, um die Bewegung von Objekten zu schätzen. Eine Übersicht über einige der bedeutendsten Veröffentlichungen wird aufgeführt.

Diese Dissertation schlägt ein Kalman Filter-Modell vor, um die 3D-Position und 3D-Geschwindigkeit von Weltpunkten zu schätzen. Messungen der Position eines Weltpunkts werden durch das Stereokamerasystem gewonnen. Die Differenzierung der Position des geschätzten Punkts erlaubt die zusätzliche Schätzung seiner Geschwindigkeit. Kalman Filter bieten eine einfache Methode Position und Geschwindigkeit zu schätzen und damit auch unabhängig bewegte Objekte zu erkennen. Die Bewegung eines statischen Weltpunkts in einem bewegten Kamerakoordinatensystem wird durch die Systemgleichungen modelliert. Die Systemgleichungen beziehen die Starrkörperbewegung der Kamera in Bezug auf die statische Umgebung ein. Die Messungen werden durch das Messmodell gewonnen, das Stereo- und Bewegungsdaten fusioniert. Ohne jegliche vorherige Informationen werden zwei Messungen gebraucht, um den Filter zu initializieren. Simulationsergebnisse validieren das Modell. Der Vergleich mit der unteren Schranke von Cramer-Rao zeigt, dass die Methode effizient ist. Die Verringerung der Positionsunsicherheit im Laufe der Zeit wird mit einer Monte-Carlo Simulation nachgewiesen.

In dem Systemgleichungsmodell müssen die Parameter der Eigenbewegung gesetzt werden. Bei Simulationen sind diese Parameter bekannt, die in realen Sequenzen geschätzt werden müssen. Dies erfolgt durch die Berechnung der absoluten Orientierung zwischen den verrauschten 3D-Punktwolken der Umgebung.

Das absolute Orientierungsproblem und die Fehlermodellierung

Um die Eigenbewegung und die Szenenstruktur zu berechnen, wird eine Analyse des Rauschens und der Fehlerfortpflanzung im Bildaufbereitungsprozess benötigt. Deshalb werden in dieser Dissertation die Rauscheigenschaften der durch Stereotriangulation erhaltenen 3D-Punkte analysiert. Dies führt zu der Entdeckung eines systematischen Fehlers in der Schätzung der 3D-Position, der sich mit einer Neuformulierung der Projektionsgleichung korrigieren lässt. Die Eigenbewegungsschätzung wird gewonnen, indem die Rotation und Translation zwischen Punktwolken geschätzt wird. Dieses Problem ist als „absolute Orientierung" bekannt und viele Lösungen auf Basis der Methode der kleinsten Quadrate sind in der Literatur vorgeschlagen worden. Diese Arbeit rezensiert die verfügbaren geschlossenen Lösungen zum Problem.

Absolute Orientierung

Der Berechnung der absoluten Orientierung in geschlossener Form ist oft in der Literatur angenähert worden. Viele Artikel sind veröffentlicht worden ohne die Kenntnisse von vorherigen Untersuchungen, so dass dieselben Methoden mehrere Male entdeckt und wieder entdeckt wurden. Dies geschah sowohl mit der quaternionenbasierten Lösung als auch mit der Methode basierend auf der Singulärwertzerlegung. Bis heute sind vier Methoden bekannt, um die gewichteten Least Squares in geschlossener Form zu lösen. Die zwei oben erwähnten, eine Methode basierend auf der Polarzerlegung und eine vierte Methode basierend auf der Doppel-Quaternionen Methode.

Der Total Least Squares Ansatz für das absolute Orientierungsproblem ist iterativ. Eine Lösung in geschlossener Form wird gefunden, wenn eine Annäherung an die gesuchte Rotationsmatrix verfügbar ist. Dennoch wird die Lösung leicht verschlechtert, da die Rotationsmatrix als die Projektion einer uneingeschränkten Lösung in der Parametermannigfaltigkeit gefunden wird.

Fehlermodellierung von 3D-Punkten

Die absolute Orientierung wird zwischen 3D-Punktwolken berechnet. Der Fehler eines 3D-Punkts, erhalten durch Stereo, wird durch den Fehler des projizierten Punkts auf das Bild und der Triangulationsgleichung charakterisiert. Verschiedene Modelle sind gemäß den Fehlereigenschaften der projizierten 3D-Punkte verfügbar.

Bei der Betrachtung der Bildquantisierung als Hauptursache für Fehler in der Stereotriangulation (durch begrenzte Auflösung) approximiert das hexaedrische Modell den Fehler als gleichmäßig verteilt in einem Volumen, das als der Schnitt zweier Pyramiden gesehen werden kann.

Wenn die Fehlerverteilung der Bildmerkmale in der Bildebene als gaußverteilt angenommen werden kann, stellt das eiförmige Modell eine bessere Annäherung an die echte Fehlerverteilung im dreidimensionalen euklidischen Raum dar. In dem eiförmigen Modell ist die geometrische Interpretation der Fehlerverteilung im euklidischen Raum ein eiförmiger Ellipsoid. Ein solcher hat zylindrische Symmetrie, aber ist asymmetrisch in einer Ebene senkrecht zur Längsachse.

Beide vorherigen Modelle machen die Fehlerfortpflanzung in späteren Stufen ziemlich schwierig. Deshalb wird die Fortpflanzung der Bildkovarianzmatrix zum dreidimensionalen euklidischen Raum berechnet, was das Ellipsoidmodell darstellt. Da die Fortpflanzung mit einer ersten Ordnungsannäherung der Triangulationsgleichung erreicht wird, führt dieses Modell zu einem systematischen Fehler in der Bewertung der 3D-Position. Dieser wird durch eine Neudefinition der Triangulationsfunktion reduziert. Die neue Triangulationsgleichung korrigiert die gemessene Disparität. Hierfür ist die reale Disparitätsvarianz erforderlich. Die Simulationsergebnisse zeigen, dass eine bedeutende Verringerung des Fehlers in der geschätzten 3D-Punktposition möglich ist, selbst wenn die echte Disparitätsvarianz grob geschätzt wird.

Eigenbewegungsschätzung

Das Problem der visuellen Schätzung der Eigenbewegung besteht in der Extraktion der Parameter der Kamerabewegung zwischen zwei Zeitpunkten. Einige Arbeiten zur Berechnung der Eigenbewegung sind in den letzten vier Jahrzehnten veröffentlicht worden. Alle Methoden können als zu einer von zwei Hauptgruppen gehörend klassifiziert werden: monokulare Methoden und multiokulare Methoden. Der Hauptunterschied zwischen beiden Gruppen ist die Relativität der Ergebnisse. Multiokulare Methoden berechnen die Starrkörpertransformation der Kameraplattform zwischen zwei Zeitpunkten. Monokulare lösen stattdessen das relative Orientierungsproblem. Der fehlenden Skalierungsfaktor kann wiederhergestellt werden, indem Annahmen über die Bewegung der Kamera oder über die Struktur der Szene gemacht werden. Beide Gruppen werden nach der Art und Weise untergliedert, wie sie den Zeitbestandteil integrieren: Methoden basierend auf optischer Fluss, Methoden basierend auf normalen Fluss, direkte Methoden und Methoden basierend auf Landmarken. Diese Arbeit rezensiert einige Hauptbeiträge zur Eigenbewegungschätzung für jede Kategorie.

Die in dieser Dissertation präsentierte Methode arbeitet als ein Prädiktor-Korrektor Algorithmus, der frühere Messungen in jede neue Iteration integriert.

Jede Iteration führt vier Hauptschritte aus: 1. Bewegungsvorhersage, 2. Anwendung der Glattheitsbedingung für die Bewegung (GBB), 3. Berechnung der absoluten Orientierung (Korrektur) und 4. Integration der neuen Bewegung zu der derzeitigen Schätzung. Die Bewegungsvorhersage hilft für die Anwendung der Glattheitsbedingung (GBB). Die GBB ist ein Kriterium, das angewandt wird, um Ausreißer, wie fremdbewegte Punkte und falsche Korrespondenzen zu verwerfen. Die GBB gewichtet Daten um den Beitrag von verrauschten Messungen zu verringern. Das Ergebnis des GBB sind zwei 3D-Punktwolken mit entsprechenden Gewichten.

Die Korrektur der vorausgesagten Eigenbewegung wird durchgeführt, indem die absolute Orientierung zwischen den Wolken von Punkten berechnet wird. Der vierte Schritt ist die Integration der zuvor erworbenen Bewegungsinformation zur gegenwärtigen Schätzung. Dieser Schritt erfolgt durch eine Zerlegung gefolgt von einer Interpolation der Bewegungsparameter.

Glattheitsbedingung für die Bewegung

Es gibt zwei Rauschvorgänge, die die Messungen beeinflussen. Das extrinsische Rauschen wird hauptsächlich durch den Messprozess des Sensors erzeugt. Das intrinsische Rauschen hat die stärkere Wirkung in Korrespondenzproblemen und wird hauptsächlich durch falsche aber auch durch richtige Zuordnungen, die aber mit dem modellierten System nicht beschrieben werden können, erzeugt. Falsche Korrespondenzen entstehen in Stereo- und Trackingalgorithmen und werden durch die zeitlich (optischer Fluss) oder die räumlich (Stereo) falsche Zuordnung von Punkten zwischen zwei Ansichten verursacht. Somit ist die Modellierung des entsprechenden extrinsischen Rauschens und die gleichzeitige Erkennung der Ausreißer von größter Bedeutung für einen robusten Algorithmus.

Diese Dissertation schlägt ein wirkungsvolles Kriterium vor, um Ausreißer (bewegte Punkte und falsche Korrespondenzen) in der Punktwolke zu entdecken. Es hilft auch das Gewicht verrauschter Messungen zu reduzieren.

Zwei Versionen des GBB werden vorgeschlagen. Eine Version für die Weighted Least Squares (WLS) und eine Version für Total Least Squares (TLS) Methode. Die WLS-Version der GBB bestimmt das Gewicht, das den Beitrag eines Punktpaares als Ganzes ohne Diskriminierung ihrer Komponenten festlegt. Ein Gewicht von Null wird angewandt, wenn das Paar keine kohärente Bewegung zeigt. Andernfalls gewichtet die Methode das Paar von Punkten gemäß der Entfernung zwischen Vorhersage und Messung. Die TLS-Version der GBB erlaubt zusätzlich die Gewichtung jedes Punktes unabhängig einzusetzen.

Simulationen werden mit gaußschem und slashschem Rauschen ausgeführt. Die Ergebnisse zeigen die Überlegenheit der GBB-Version über die Standardgewichtungsmethoden. Die Stabilität der Ergebnisse hinsichtlich Ausreißern wurde analysiert. Es zeigt sich, dass der „break down point" größer als 50% ist.

Multiframeschätzung

Zweiframeschätzung, d. h. die Schätzung der Bewegungsparameter zwischen dem gegenwärtigen und dem vorherigen Bildpaar, ist der Standardfall in den meisten Ansätzen. Das Hauptproblem dieser Annäherung ist die schnelle Fehlerakkumulation. Der Fehler in der Eigenposition wächst superlinear im Laufe der Zeit. Diese Arbeit schlägt eine Verringerung der Fehlerakkumulation durch die Integration mehrerer Frames in der aktuellen Bewegungsschätzung vor.

Der Algorithmus führt vier Hauptschritte aus: 1. Bewegungsvorhersage, 2. Anwendung des GBB, 3. Bewegungskorrektur und 4. Bewegungsintegration. Wenn diese vier Schritte iterativ ausgeführt werden, wird ein Prädiktor-Korrektor-Verfahren gewonnen. Die erste Iteration wird zwischen der aktuellen und vorherigen Wolke von Punkten durchgeführt. Jede weitere Iteration integriert eine zusätzliche Punktwolke eines vorherigen Zeitpunkts. Diese Methode reduziert die Fehlerakkumulation bei der Integration von mehreren Schätzungen in einer einzigen globalen Schätzung. Die Integration von Bewegungsschätzwerten wird mit sphärischer linearer Interpolation für die Rotationsmatrizen und linearer Interpolation bei der Verschiebungsvektoren durchgeführt. Die Faktoren für die Interpolationen werden vom Residuum der absoluten Orientierungsschätzung berechnet.

Simulationsergebnisse zeigen, dass, obwohl der Fehler noch superlinear im Laufe der Zeit zunimmt, die Größe des Fehlers um mehrere Größenordnungen reduziert wird.

Fusion von visueller Odometrie und Inertialsensoren

Die in dieser Dissertation vorgeschlagene Methode arbeitet ausschließlich mit den Bildern, geliefert durch das Stereosystem. Doch es gibt Situationen, in denen das Stereosystem in seiner Leistungsfähigkeit eingeschränkt ist, z. B. ganze oder teilweise Verdeckung durch den Scheibenwischer oder direkte Einstrahlung von Sonnenlicht.

Wenn das Fahrzeug mit zusätzlichen Systemen für die Messung der Bewegung ausgestattet wird, wie ein Tachometer, Gierraten- und GPS-Sensor, können die redundanten Informationen zur Erhöhung der Robustheit und Verbesserung der Schätzung verwendet werden. Zu diesem Zweck wird eine Kovarianzmatrix der visuellen Schätzung abgeleitet. Geschätzte Zustände mit entsprechenden Kovarianzmatrizen werden in einer einzelnen Endschätzung fusioniert.

Experimentelle Ergebnisse

Experimentelle Ergebnisse werden mit langen Sequenzen von Bildern erzielt.

Die ersten beiden Testsequenzen wurden aufgenommen, während das Fahrzeug Strecken mit mehreren Kurven bei einer Geschwindigkeit von $0\ km/h$ und $60\ km/h$ zurücklegte. Die Framerate für beide Sequenzen beträgt 10 Frames pro Sekunde. Die Basisbreite des Stereokamerasystems ist $0,35$ Meter und die Bilder haben einen Standard-VGA-Auflösung (640×480 Pixels). Jede Sequenz umfasst eine Entfernung von mindestens $1,25\,km$. Die Sequenzen enthalten mehr als 25 fremdbewegte Objekte, hauptsächlich entgegenkommende Autos und Busse. Ein Kalman Filter wird für jedes neue Merkmal initialisiert und das Filter wird im Laufe der Zeit aktualisiert. Die Eigenbewegung wurde mit und ohne Verwendung der Inertialsensorik berechnet. Der herausgezogene Pfad wurde über die Luftansichten der Straßen geplottet. Nur geringe Abweichungen werden zwischen dem Weg der Luftansichten und dem geschätzten Pfad beobachtet. Die durchschnittliche Zeit für die Berechnung der Eigenbewegung betrug $8,5\ ms$ und für das Kalman Filter $9,8\ ms$.

Zusätzliche Tests, einschließlich einer 3D-Rekonstruktion einer Waldszene und der Berechnung der freien Kamerabewegung in einem Indoor-Szenario, wurden durchgeführt. Die Methode zeigt gute Ergebnisse in allen Fällen.

Die Methode wurde auch an die Schätzung der Lage von kleinen Objekten angepasst. Eine Anpassung ist erforderlich, da bei der Schätzung der Lage eines Objektes die Kamera statisch bleibt während sich das Zielobjekt mit den sechs Freiheitsgraden bewegt. Die Genauigkeit des Algorithmus wurde mit synthetischen Bildern von Objekten unterschiedlicher Größe und mit einer Winkelauflösung von 2° bis 8° getestet. Der maximale Fehler, einschließlich der integrierten Fehler der Lage über 100 Frames, bleibt innerhalb von 10°. Der durchschnittliche Fehler bleibt weit unter 1°. Der Algorithmus liefert zudem akzeptable Ergebnisse bei der Schätzung der Lage kleiner Objekte, wie Köpfe und Beine von realen Crash-Test-Dummies.

Schlussfolgerungen

In dieser Arbeit wurde ein robustes, echtzeitfähiges Verfahren vorgestellt, das es erlaubt, alle 6 Freiheitsgrade eines bewegte Kamerasystems ausschliesslich anhand von Stereobildsequenzen zu schätzen. Die Inertialsensoren des Fahrzeugs können verwendet werden, um die Robustheit der Schätzung zu erhöhen.

In dieser Dissertation wurde auch die Anpassungsfähigkeit der Methode auf verschiedenen Szenarios und Anwendungen gezeigt. Die experimentellen Ergebnisse haben die mögliche Anwendbarkeit dieser Methode als SLAM Methode gezeigt, indem eine 3D-Rekonstruktion für eine ganze Szene gemacht wurde. Innenanwendungen sind auch mit guten Resultaten untersucht worden. Schließlich wurde die Methode angepasst, um die präzise Lage von kleinen Objekten zu schätzen.

Abstract

Driving can be dangerous. Humans become inattentive when performing a monotonous task like driving. Also the risk implied while multi-tasking, like using the cellular phone while driving, can break the concentration of the driver and increase the risk of accidents. Others factors like exhaustion, nervousness and excitement affect the performance of the driver and the response time. Consequently, car manufacturers have developed systems in the last decades which assist the driver under various circumstances. These systems are called *driver assistance systems*.

Driver assistance systems are meant to support the task of driving, and the field of action varies from alerting the driver, with acoustical or optical warnings, to taking control of the car, such as keeping the vehicle in the traffic lane until the driver resumes control. For such a purpose, the vehicle is equipped with on-board sensors which allow the perception of the environment and/or the state of the vehicle. Cameras are sensors which extract useful information about the visual appearance of the environment. Additionally, a binocular system allows the extraction of 3D information.

One of the main requirements for most camera-based driver assistance systems is the accurate knowledge of the motion of the vehicle. Some sources of information, like velocimeters and GPS, are of common use in vehicles today. Nevertheless, the resolution and accuracy usually achieved with these systems are not enough for many real-time applications. The computation of ego-motion from sequences of stereo images for the implementation of driving intelligent systems, like autonomous navigation or collision avoidance, constitutes the core of this thesis.

This dissertation proposes a framework for the simultaneous computation of the 6 degrees of freedom of ego-motion (rotation and translation in 3D Euclidean space), the estimation of the scene structure and the detection and estimation of independently moving objects. The input is exclusively provided by a binocular system and the framework does not call for any data acquisition strategy, i.e. the stereo images are just processed as they are provided. Stereo allows one to establish correspondences between left and right images, estimating 3D points of the environment via triangulation. Likewise, feature tracking establishes correspondences between the images acquired at different time instances. When both are used together for a large number of points, the result is a set of clouds of 3D points with point-to-point correspondences between clouds.

The apparent motion of the 3D points between consecutive frames is caused by a variety of reasons. The most dominant motion for most of the points in the clouds is caused by the ego-motion of the vehicle; as the vehicle moves and images are acquired, the relative position of the world points with respect to the vehicle changes. Motion is also caused by objects moving in the environment. They move independently of the vehicle motion, so the observed motion for these points is the sum of the ego-vehicle motion and the independent motion of the object. A third reason, and of paramount importance in vision applications, is caused by correspondence problems, i.e. the incorrect spatial or temporal assignment of the point-to-point correspondence. Furthermore, all the points in the clouds are actually noisy measurements of the real unknown 3D points of the environment.

Solving ego-motion and scene structure from the clouds of points requires some previous analysis of the noise involved in the imaging process, and how it propagates as the data is processed. Therefore, this dissertation analyzes the noise properties of the 3D points obtained through stereo triangulation. This leads to the detection of a bias in the estimation of 3D position, which is corrected with a reformulation of the projection equation. Ego-motion is obtained by finding the rotation and translation between the two clouds of points. This problem is known as *absolute orientation*, and many solutions based on least squares have been proposed in the literature. This thesis reviews the available closed form solutions to the problem.

The proposed framework is divided in three main blocks: 1) stereo and feature tracking computation, 2) ego-motion estimation and 3) estimation of 3D point position and 3D velocity. The first block solves the correspondence problem providing the clouds of points as output. No special implementation of this block is required in this thesis.

The ego-motion block computes the motion of the cameras by finding the absolute orientation between the clouds of static points in the environment. Since the cloud of points might contain independently moving objects and outliers generated by false correspondences, the direct computation of the least squares might lead to an erroneous solution. The first contribution of this thesis is an effective rejection rule that detects outliers based on the distance between predicted and measured quantities, and reduces the effects of noisy measurement by assigning appropriate weights to the data. This method is called Smoothness Motion Constraint (SMC).

The ego-motion of the camera between two frames is obtained finding the absolute orientation between consecutive clouds of weighted 3D points. The complete ego-motion since initialization is achieved concatenating the individual motion estimates. This leads to a super-linear propagation of the error, since noise is integrated.

A second contribution of this dissertation is a predictor/corrector iterative method, which integrates the clouds of 3D points of multiple time instances for the computation of ego-motion. The presented method considerably reduces the accumulation of errors in the estimated ego-position of the camera.

Another contribution of this dissertation is a method which recursively estimates the 3D world position of a point and its velocity; by fusing stereo, feature tracking and the estimated ego-motion in a Kalman Filter system. An improved estimation of point position is obtained this way, which is used in the subsequent system cycle resulting in an improved computation of ego-motion.

The general contribution of this dissertation is a single framework for the real time computation of scene structure, independently moving objects and ego-motion for automotive applications.

Chapter 1

Introduction

1.1 Motivation

Drivers often perform multiple tasks while driving. Listening to music or news on the radio, eating or using the cellular phone are typical activities performed while driving. The risk implied by attending multiple tasks can break the concentration of the driver, diverting their attention off the road and increasing the risk of accidents. Furthermore, humans become inattentive when performing a monotonous task like driving. These factors and others like exhaustion, nervousness and excitement affect the performance of the driver and the response time.

The driving performance is not only affected by the limitations of the driver but also by external factors like the weather. For example, there are difficulties when:

- braking and keeping the car steerable at the same time;
- pressing the brake pedal sufficiently when braking in an emergency situation;
- avoid skidding in a curve at high velocity; and
- perceiving the distance to other vehicles.

Consequently car manufacturers have developed systems in the last decades which assist the driver under various circumstances. Examples of this are given by the Anti-lock Brake System (ABS) and the Brake Assistance, the Electronic Stability Program (ESP) and the Adaptive Cruise Control (ACC). These systems are examples of *driver assistance systems*.

Driver assistance systems support the task of driving, and the field of action varies from alerting the driver with acoustical or optical warnings to taking the control of the car, such as keeping the vehicle in the traffic lane until the driver resumes control. For such a purpose, the vehicle is equipped with on-board sensors, which allow the perception of the environment and/or the state of the vehicle.

There are two classes of sensors: passive and active. Active sensors emit some form of energy and then measure the impact as a way of understanding the environment (e.g. millimeter-wave radars, sonar sensors and lasers radars) while passive sensors receive energy already in the environment. A camera is a passive sensor which allows the extraction of useful data about the appearance of the environment. A combination of multiple cameras allows the instantaneous extraction of 3-D information. Therefore, multiple cameras are often used to measure the structure of the environment at a specific instant in time.

The data obtained with cameras must be processed in order to generate information. *Computer Vision* is the branch of science that develops methods and techniques for the extraction of information from images. Visual motion is an essential cue in the computer vision community, which studies the space variation in the image over time. From the psychological point of view, visual motion serves a wide variety of crucial roles: "way-finding (optic flow), perception of shape from motion, depth segregation, judgments of coincidence (time to collision, time to filling a tea cup), judgments of motion direction and speed, and perception of animate, biological activity" [SWB04]. The implementation of algorithms which carry out these perception tasks is essential for the development of intelligent vehicle assistance systems, specially for the detection and interpretation of the driving environment.

One of the main requirements for most camera-based driver assistance systems is the accurate knowledge of the motion of the camera. Knowing the motion of the camera means knowing the current state of the camera and the vehicle w.r.t. its environment, i.e. the position, orientation and velocity. The accurate knowledge of the motion of the camera allows the establishment of geometrical constraints which highly reduces the complexity of many computer vision problems (e.g. the correspondence problem). The computation of ego-motion from sequences of images (also called *visual odometry information*) for the implementation of driving intelligent systems, like obstacle detection, constitutes the core of this dissertation.

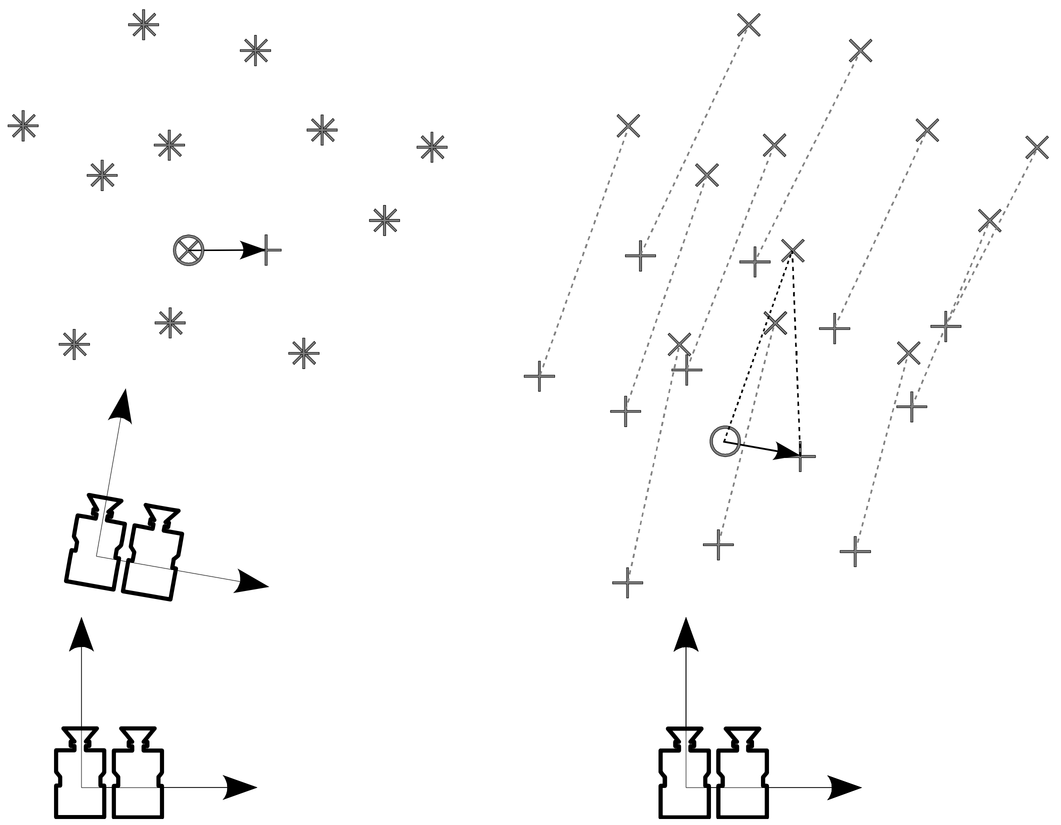
1.2 Objectives of the Dissertation

The development of intelligent systems is a very complex task which requires a clear *understanding* of traffic situations. Determining the scene structure, computing the ego-motion and estimating the independently moving objects of the environment are low level tasks required for the interpretation of the traffic environment. The determination of the scene structure at a specific instant in time is achieved with stereo cameras. A stereo camera system consists of two or more cameras synchronized to acquire images simultaneously. A triangulation between image points allows the computation of 3D point positions. When this operation is performed for many points in the image, an instantaneous structure of the scene is available. An analysis of the cloud of points can result in the identification of potential obstacles. Nevertheless, making a decision about the data provided by an instantaneous shot of the scene is usually dangerous, since very important motion information is missing. Erroneous data in the cloud of 3D points can also lead to making the wrong decision.

The 3D structure of the scene changes as the ego-vehicle[1] and other traffic participants move. For instance, if the vehicle drives straight ahead, the new structure of

---

[1] The prefix *ego* is used to refer to the reference point of the viewer. In this specific case *ego-vehicle* is the vehicle where the camera is installed in.



(a) Motion observed from the environment. (b) Motion as observed from the camera.

Figure 1.1: Motion of the cloud of points. The left figure shows a cloud of points in the environment and the corresponding motion of the camera platform. When the same motion is viewed from a camera reference frame, the cloud of points does not coincide. Points corresponding to the static scene are described with a rotation and a translation, which corresponds to the camera motion. Independently moving points are not described in the same way. The new position deviates from the expectation (denoted with a circle). The difference between expected position and observed position is the proper motion of the point.

the static scene will be shifted towards the vehicle; if the vehicle was driving on a curve, the new structure will also be rotated (see Figure 1.1). Registering the change in the structure of the scene can be realized in various ways. This is addressed later in Section 4.2. One way of integrating information over time is assigning a point-to-point correspondence between images of consecutive times. The point-to-point correspondence is known as *feature tracking*. Thus, the data obtained with stereo and feature tracking consist of two clouds of 3D points with point-to-point correspondences. The obtained motion of each point in the clouds is a combination of the following two reasons: 1) the ego-vehicle is in motion, and 2) other participants are moving.

If a point is static, its observed motion is described with the inverse motion of the camera. Thus, knowing the ego-motion of the camera allows the establishment of static points. If the motion of the point is not described with the inverse motion of the camera, since it deviates from the theoretical motion, the point corresponds to an independently moving object. The detected deviation is the proper motion of the point, which is estimated (see Figure 1.1(b))

This dissertation has two objectives. The first, and main objective, is the computation of the motion of the camera from the clouds of 3D points obtained with stereo vision; where the point-to-point correspondence is obtained by feature tracking. The second objective is the estimation of the position of the points in the environment and their motion. This way, ego-motion, scene structure and independently moving objects are estimated in a single framework, providing useful information to more complex automotive visual perception systems, such as autonomous navigation or collision avoidance.

1.3 Contributions of the Dissertation

The scenario, as described in the previous section, is far form being realistic, since the noise involved when measuring quantities from images was ignored. Stereo and feature tracking are affected by noise. The uncertainty of 3D point positions introduced by noise, can be reduced by integrating multiple measurements of the same point over time. Feature tracking provides the image coordinates of the projection of a world point, while stereo provides its corresponding disparity. When a point is static, its observed motion from a camera reference frame is defined by the inverse camera motion. Any deviation with the expected motion is caused by noise or independent motion. The first contribution of this dissertation is a method which recursively estimates the 3D world position of a point and its velocity; by fusing stereo, feature tracking and the 6 degrees of freedom of the ego-motion in a Kalman Filter system.

The motion of the camera, between current and previous times, is implicitly given by the cloud of noisy static points obtained with stereo. Ego-motion is obtained by finding the rotation and translation between the two clouds of points. However, the noisy measurements are not the only problem an estimator must deal with. Outliers represent also a problem and they are caused by two main reasons:

- *Incorrect matches*: caused by the temporal (optical flow displacement) or spatial (stereo disparity) incorrect establishment of the point-to-point correspondence.
- *Correct matches which cannot be described by the model*: this is the case when a point assumed to be static, actually belongs to an independently moving object.

Thus, the appropriate modeling of noise and the opportunely detection of outliers is of paramount importance when a robust algorithm is desired. The second contribution of this dissertation is a simple but effective rejection rule that detects outliers based on the distance between predicted and measured quantities and reduces the effects of noisy measurement by assigning appropriate weights to the data. We call this method the Smoothness Motion Constraint (SMC).

The third contribution of this dissertation is a predictor/corrector iterative method which integrates the clouds of 3D points of multiple times for the computation of ego-motion. This reduces considerably the accumulation of errors in the estimated ego-position of the camera, leading to a better estimation of point position and velocity with Kalman Filters.

The general contribution of this dissertation is a single framework for the computation of scene structure, independently moving objects and ego-motion in real time for automotive applications.

1.4 Dissertation Overview

The remainder of this work is organized as follows. Chapter 2 is dedicated to basic introductory concepts of the image, the image geometry and the correspondence problem, which are required in the remaining chapters of this dissertation. The topics addressed are the image formation, the geometrical properties of binocular systems, the calibration and rectification, the image primitives and the correspondence of image primitives.

Chapter 3 formulates the problem statement in detail. The chapter reviews some alternative sensors for the computation of ego-motion and proposes a framework for the computation of the ego-motion, the scene structure and independently moving points. Some characteristics of the proposed solution are also outlined in the chapter.

The Kalman Filter system which estimates the position and velocity of world points in 3D Euclidean space is fully described in Chapter 4. An improvement in the estimation is possible thanks to the integration of stereo and feature tracking measurements over time. The role of ego-motion in the Kalman Filter is made clear in system model equations. The initialization of the filter is addressed and the computation of the Cramér-Rao lower bound for the Kalman Filter is given. Various simulation results validate the filter and show its efficiency.

Chapters 5 and 6 deal with two topics; the computation of the absolute orientation between two clouds of 3D points and the modeling of stereo error. The absolute orientation problem was already addressed multiple times in the literature, and here a complete review of these publications is given. Chapter 5 focuses on the closed form solutions to the absolute orientation problem and reviews principally the weighted least squares solutions. A total least squares closed form solutions which has had little attention in the literature is also reviewed. In Chapter 6 an analysis of the error in the estimation of 3D position with stereo is also carried out. The propagation of the covariance matrices leads to the ellipsoidal modeling of 3D position. In this model, constant probability contours of the distribution of the points describe ellipsoids about the nominal mean, that approximate the true error. The true error distribution is more egg-shaped, and therefore a bias in the estimation of 3D position takes place. The reduction of the estimation bias is achieved by a Taylor series expansion of the triangulation equation.

The ego-motion estimation algorithm is presented in Chapter 7. The chapter starts with a literature review of ego-motion estimation methods. An overview of the proposed algorithm is given, and an outlier rejection rule called Smoothness Motion Constraint is presented for the weighted and total least squares approaches. Simulation results showing the effectiveness of the method under different situations are carried out. The multi-frame estimation is presented and simulations results showing the reduction of the integration error are carried out. The use of filtered data obtained from a Kalman Filter is also addressed, and a strategy for avoiding falling into a positive feedback loop is described. The possibility of fusing inertial sensor information is also outlined.

Experimental results showing the performance of the proposed approach are shown in Chapter 8. The method is not only evaluated on traffic environments, but also in off-road, indoor and industrial scenarios. The various application possibilities of the method are described. The last chapter summarizes and concludes this dissertation.

Chapter 2

Image Geometry and the Correspondence Problem

2.1 Introduction

This chapter reviews some topics of image geometry and the correspondence problem, which will be required in the following chapters of this dissertation. Only basic concepts are introduced here and the reader is invited to consult the bibliography listed at the end of the chapter for further details about the addressed topics.

2.2 Image Formation and Camera Geometry

2.2.1 Image

A digital image is represented by a two dimensional array or matrix. The elements of the matrix are called pixels [1] and the value assigned to each element of the matrix is its associated gray level. Formally,

$$I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+; (u, v) \mapsto I(u, v) \tag{2.1}$$

an image is a map I defined on a compact region Ω of a two-dimensional surface, taking values in the positive real numbers [YSJS04]. In the case of digital image, both the domain Ω and the range $\mathbb{R}_+$ are discretized. For instance, $\Omega = [0, 639] \times [0, 479] \subset \mathbb{N}_0^2$ and $\mathbb{R}_+$ is approximated by an interval of integers $[0, 255] \subset \mathbb{N}_0$. The image configuration above mentioned is used in the remainder of this work, when not stated otherwise.

The value of each point of the image is typically called *image intensity*, *brightness* or *irradiance* and describes the energy falling onto a small patch of the imaging sensor. The irradiance value depends among others on the exposure time of the sensor, the shape of the object or objects in the region of space being measured, the material

---

[1] Also called *pel*. Both, *pixel* and *pel* are commonly used abbreviations of *picture element*

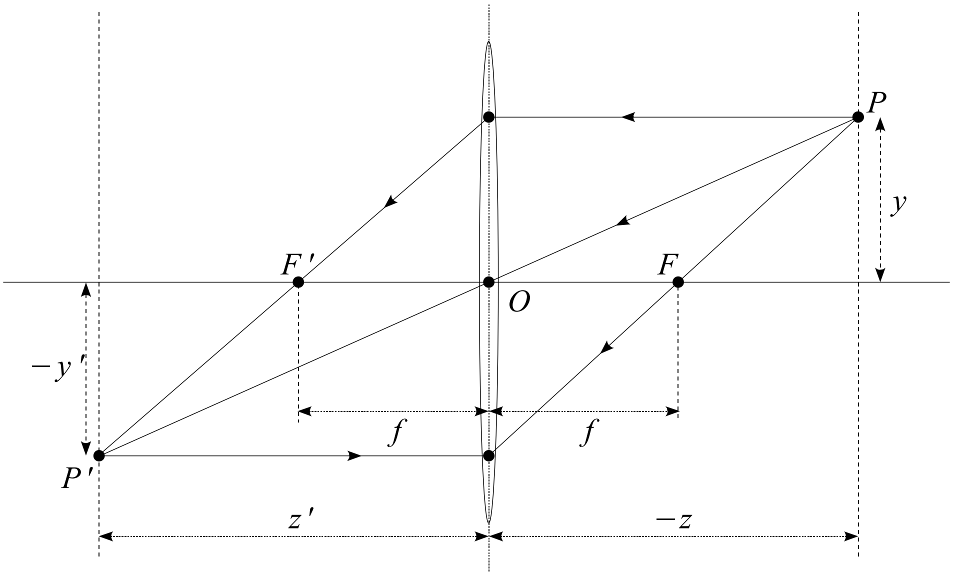


Figure 2.1: Thin lens.

of the objects, the illumination and the optics of the imaging device. The measuring of light corresponds to the radiometry[2] which is a science *per se*. In the following, only *Lambertian surfaces* are assumed. The radiance of a Lambertian surface only depends on how the surface faces the light source, but not on the direction from which it is viewed. This assumption allows the derivation of expressions for the establishment of correspondences between multiple images of the same object. This is shown in the next sections.

2.2.2 Thin Lenses and Pinhole Camera

2.2.2.1 Thin Lens Model

Real images are obtained with optical systems, such as a camera devices. Camera devices are composed of a set of lenses in order to direct light in a controlled manner. This section describes the imaging through thin lenses.

A thin lens is a spherical refractive surface, symmetrical across the vertical and horizontal planes (see Figure 2.1). The horizontal axis which passes exactly through the center of the lens is the *optical axis*. The plane perpendicular to the optical axis, which bisects the symmetrical lens in two, is the *focal plane*. The *optical center* O is defined as the intersection between the optical axis and the focal plane. Light rays incident towards either face of the lens and traveling parallel to the principal axis converge to a point on the optical axis called *focus* or *focal point*. The distance f between the focal point and the optical center is the *focal length* of the lens. In Figure 2.1 F and F' are both focal points and are equidistant from the optical center. An important property of thin lenses is that rays passing through the optical center

---

[2] Also called *photometry* if the interest lies only on light detected by the human eye (wavelength range from about 360 to 830 nm).

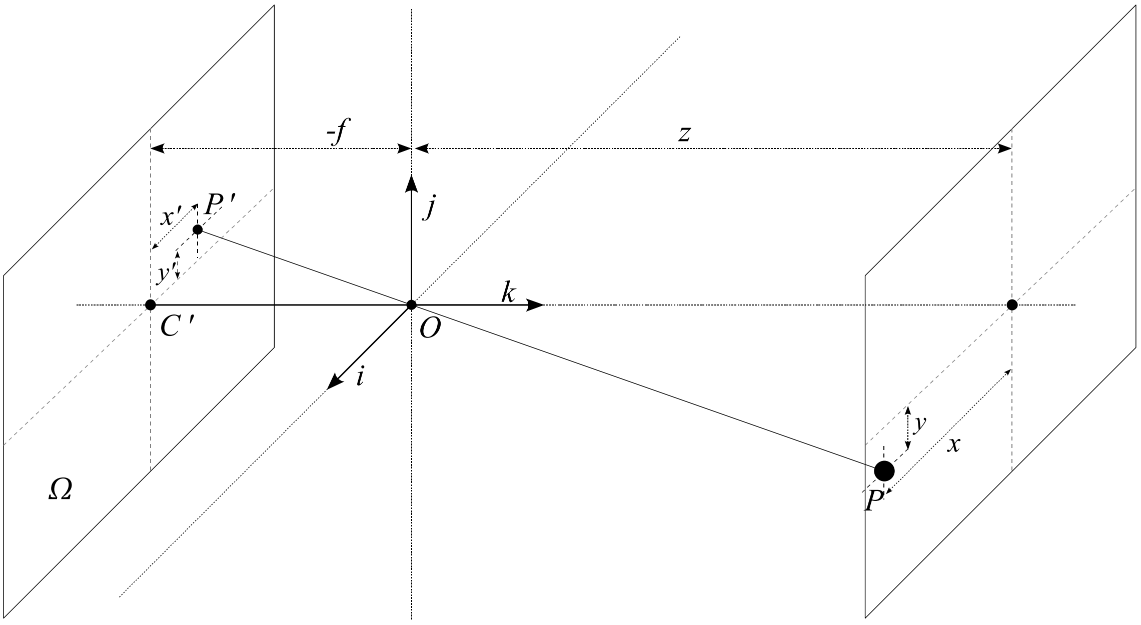


Figure 2.2: Perspective camera projection

are not deflected. Consider a point $P \in \mathbb{E}^3$ at a distance z from the focal plane. Let $\overrightarrow{PO}$ denote the ray passing through P and the optical center. Now consider a ray parallel to the optical axis passing through P. The parallel ray is refracted by the lens and intersects $\overrightarrow{PO}$ at P'[3], which is at a distance z' from the focal plane. Thus, it can be argued that every ray from P intersects at P' on the other side of the lens. In particular the ray from P' parallel to the optical axis pass through P . With the above geometry formation, the *fundamental equation of the thin lens* is obtained:

$$\frac{1}{z} + \frac{1}{z'} = \frac{1}{f} \tag{2.2}$$

2.2.2.2 Ideal Pinhole Camera

Letting the aperture of the lens decrease to zero, all rays are forced to go through the center of the lens and therefore are not refracted. All the irradiance corresponding to P' is given by the points lying on a line passing through the center O (see Figure 2.2). Let us consider the coordinate system (O, i, j, k) with center O, depth component k and (i, j) forming a basis for a vector plane parallel to the image plane Ω at a distance f from the origin. The line passing through the origin and perpendicular to the image plane is the *optical axis*, which pierces the images plane at the *image center* C'. Let P be a point with coordinates (x, y, z) and let be P' its image with coordinate $(x', y', -f)$. Since P, O and P' are collinear then $\overrightarrow{OP'} = \lambda\overrightarrow{OP}$ for some

---

[3] We call P' the *image* of P

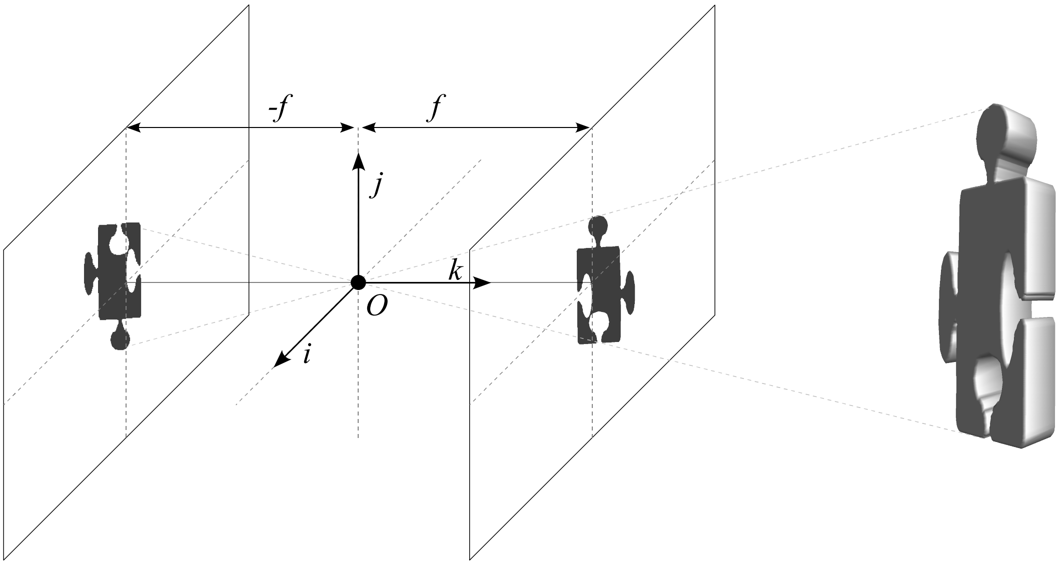


Figure 2.3: Frontal Pinhole Camera

λ, therefore:

$$\begin{array}{rcl} x' & = & \lambda x \\ y' & = & \lambda y \quad \Leftarrow: \quad \lambda = \dfrac{x'}{x} = \dfrac{y'}{y} = \dfrac{-f}{z} \\ -f & = & \lambda z \end{array} \tag{2.3}$$

this obtains the *ideal perspective projection* equations:

$$x' = -f\frac{x}{z} \qquad y' = -f\frac{y}{z} \tag{2.4}$$

This model is known as *ideal pinhole camera*. It is an idealization of the thin lens model, since as the aperture decreases, the energy going through the lens becomes zero. The thin lens model is also an idealization of real lenses. For example, diffraction and reflection are assumed to be negligible in the thin lens model. Other characteristics of real lenses are spherical and chromatic aberration, radial distortion and vignetting. Therefore, the ideal pinhole camera is a geometric approximation of a well-focused imaging system.

2.2.2.3 Frontal Pinhole Camera

Since the image plane is at position $-f$ from the optical center O, the image of the scene obtained is inverted. In order to simplify drawings, the image plane is moved to a positive distance f from O as shown in Figure 2.3. In the remainder of this dissertation this frontal representation will be used. All geometric and algebraic arguments presented hold true when the image plane is actually *behind* the

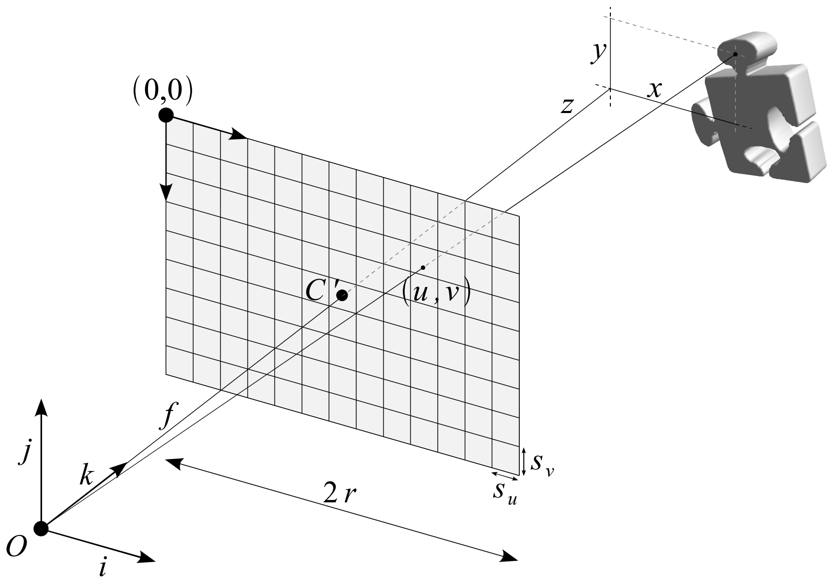


Figure 2.4: Camera and image coordinate systems.

corresponding pinholes. The new perspective equations are thus given by:

$$x' = f\frac{x}{z} \qquad y' = f\frac{y}{z} \tag{2.5}$$

where (x', y') are in a *retinal plane coordinates frame*.

2.2.2.4 Field of View

In practice, the area of the sensor of the camera device is limited and therefore, not every world point will have an image in the sensor area. The *field of view* (FOV) of the camera is the portion of the scene space that actually is projected on the image plane. The FOV varies with the focal length f and area of the image plane. When the sensor is rectangular, a horizontal and vertical FOV is usually defined. The FOV is usually specified in angles and can be obtained by

$$\theta = 2\arctan(r/f) \tag{2.6}$$

where θ is the FOV angle and $2r$ is the spatial extension of the sensor (see Figure 2.4).

2.2.2.5 Camera and Image Coordinate System

Equations 2.5 relate the 3D position of a point and its projection on the retinal plane, using the coordinate system specified for the camera. On the other side, a digital image is composed of pixels, where $(0, 0)$ is the coordinates of the pixel of the upper-left corner of the image (see Figure 2.4). The following equations relate the retinal plane coordinate frame with the image coordinate frame:

$$(u - u_0)s_u = f\frac{x}{z} \qquad (v_0 - v)s_v = f\frac{y}{z}$$

where (s_u, s_v) is the width and height of the pixel in the camera sensor and (u_0, v_0) is the image position in pixels corresponding to the image center C'. Expressing the focal length in pixel width and height, i.e. $f_u = \frac{f}{s_u}$ and $f_v = \frac{f}{s_v}$ respectively, the projection of a world point P in the image plane is given by

$$u = u_0 + f_u\frac{x}{z} \qquad v = v_0 - f_v\frac{y}{z}$$

In a homogeneous coordinate system the following representation is also used:

$$\lambda \underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{{}^{\cdot}\boldsymbol{p}'} = \underbrace{\begin{bmatrix} f_u & f_\theta & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{K}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Upsilon_0} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{{}^{\cdot}\boldsymbol{P}} \qquad (2.7)$$

$$\lambda\, {}^{\cdot}\boldsymbol{p}' = \boldsymbol{K}\Upsilon_0\, {}^{\cdot}\boldsymbol{P}$$

where $\boldsymbol{K}$ is known as *intrinsic parameter matrix* or *calibration matrix* and Υ_0 as *projection matrix*, and ${}^{\cdot}\boldsymbol{P}$ is the vector homogeneous coordinate of point $\boldsymbol{P}$. Observe that the second diagonal element in the projection matrix is negative because the vertical dimension has opposite direction in the image coordinate system [4]. The scalar f_θ in the matrix $\boldsymbol{K}$ is equivalent to $\frac{f_u}{s_\theta}$ where $s_\theta = \cot\theta$ is called *skew factor* and θ is the angle between the image axes (because of manufacturing error). Nevertheless, in current hardware θ is very close to 90° and therefore the skew factor is very close to zero.

2.3 Geometry of Two Views

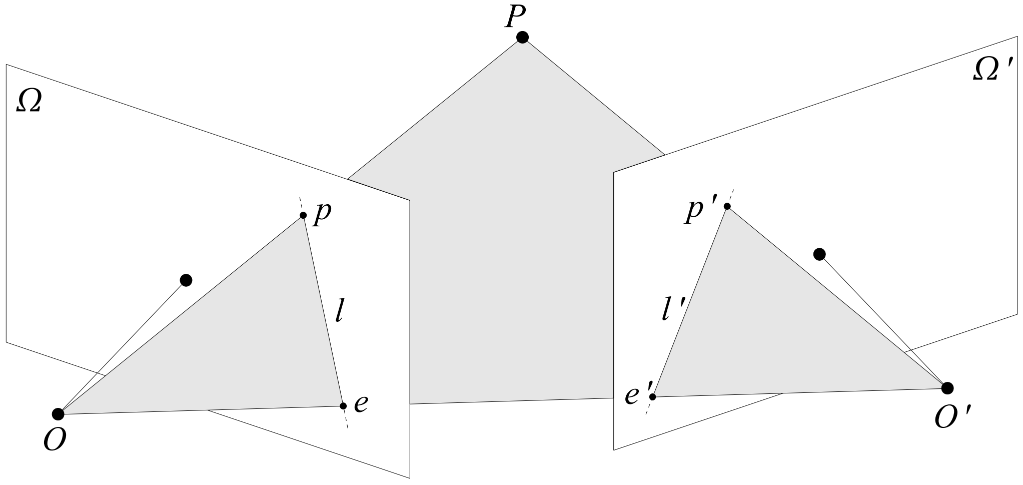
A perspective projection is the mapping of a three-dimensional space into a two-dimensional space. Formally,

$$\pi : \mathbb{R}^3 \longrightarrow \mathbb{R}^2; \quad P \mapsto P'. \qquad (2.8)$$

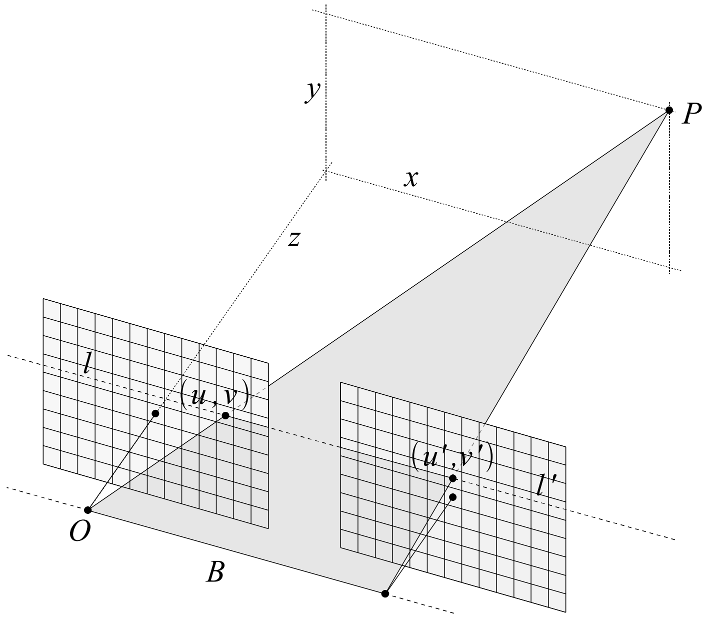
A characteristic of the projection is that the scale factor λ of Equation 2.3 is lost. In planar perspective projection, the unknown scale factor corresponds to the depth z of the projected point. Nevertheless, if two or more images of the same point taken at different known positions are available, the unknown scale factor can be recovered through triangulation. Before deriving the triangulation equations, this section introduces some basic geometric properties when two images of the same scene are

---

[4] The camera coordinate system in Figure 2.4 is left-handed. The remainder of this work will use a left-handed coordinate system as shown in Figure 2.4.



(a) epipolar geometry with two image planes.



(b) standard stereo configuration.

Figure 2.5: Two-View geometry.

available. The properties derived here allow the simplification of the triangulation equations, and the reduction of the search space for the correspondence problem.

2.3.1 Epipolar Geometry

Figure 2.5(a) shows the imaging process of a point P into two views. In the figure, O and O' are the focal points of each camera, and p and p' are the images of P. The plane defined by POO' is called the *epipolar plane*, and the lines l and l' obtained by the intersection of the epipolar plane with the image planes are called *epipolar* *lines*. The *epipolar constraint* expresses the coplanarity of the vectors $\overrightarrow{Op}$, $\overrightarrow{O'p'}$ and $\overrightarrow{OO'}$ and therefore

$$\overrightarrow{Op}(\overrightarrow{OO'} \times \overrightarrow{O'p'}) = 0 \tag{2.9}$$

If $\boldsymbol{R}$ is the rotation matrix relating the relative orientation of the second camera with the first camera, and $\boldsymbol{t}$ the translation coordinate vector separating $\overrightarrow{OO'}$ Equation 2.9 is equivalent to:

$$\boldsymbol{p}^T(\boldsymbol{t} \times \boldsymbol{R}\boldsymbol{p}') = 0$$

where $\boldsymbol{p}$ and $\boldsymbol{p'}$ are the homogeneous image coordinate vectors of the points p and p'.

$$\boldsymbol{p}^T \boldsymbol{E}\boldsymbol{p}' = 0 \tag{2.10}$$

where $\boldsymbol{E}$ is called *essential matrix* and is equal to $\boldsymbol{t}_\times \boldsymbol{R}$ where $\boldsymbol{a}_\times$ is the matrix such that $\boldsymbol{a}_\times \boldsymbol{x} = \boldsymbol{a} \times \boldsymbol{x}$, with

$$\boldsymbol{a}_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.11}$$

allowing us to express the cross product of two vectors as the product of a skew-symmetric matrix and a vector. Equation 2.10 shows that point p lies on the epipolar line defined by the vector $\boldsymbol{E}\boldsymbol{p}'$. The images e and e' are called *epipoles*. They are the projections of the optical center in the camera image plane of the other camera, and therefore $\boldsymbol{E}^T\boldsymbol{e} = \boldsymbol{e}'^T\boldsymbol{E} = \boldsymbol{0}$.

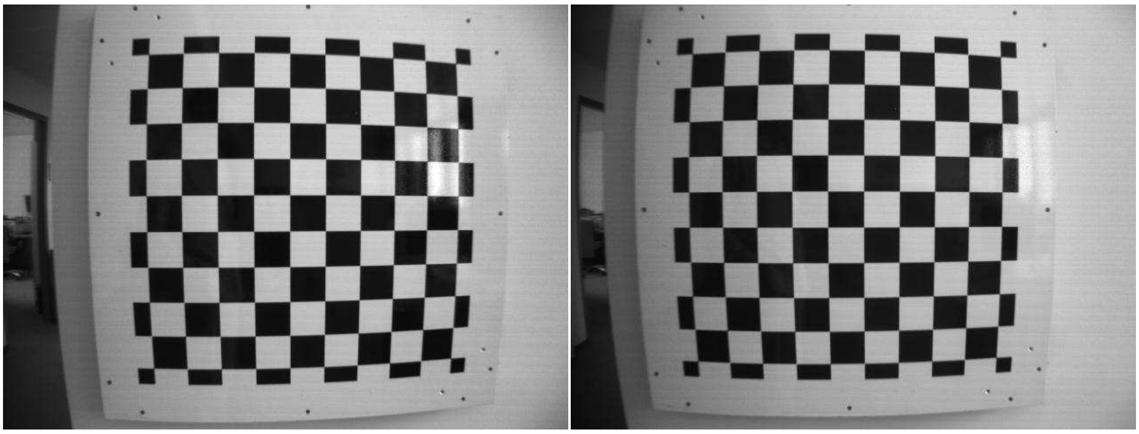
2.3.2 Standard Stereo Configuration

When the relative pose between both cameras is only a lateral translation, i.e. $\boldsymbol{R} = \boldsymbol{I}_{3x3}$ and $\boldsymbol{t} = (B, 0, 0)$ the epipoles lie at a lateral position of infinity and the epipolar lines are aligned with the rows of the images. The distance B is called the *baseline*. Given an image point p_l with coordinates (u_l, v_l) in the *left* image, its corresponding right image p_r at (u_r, v_r) is found in exactly the same image scanline (see Figure 2.5(b)), i.e. $v_l = v_r$, since the epipolar lines are now collinear. The distance $d = u_l - u_r$ is called *disparity*. The reconstruction of the 3D position (X, Y, Z) of P is then obtained by triangulation:

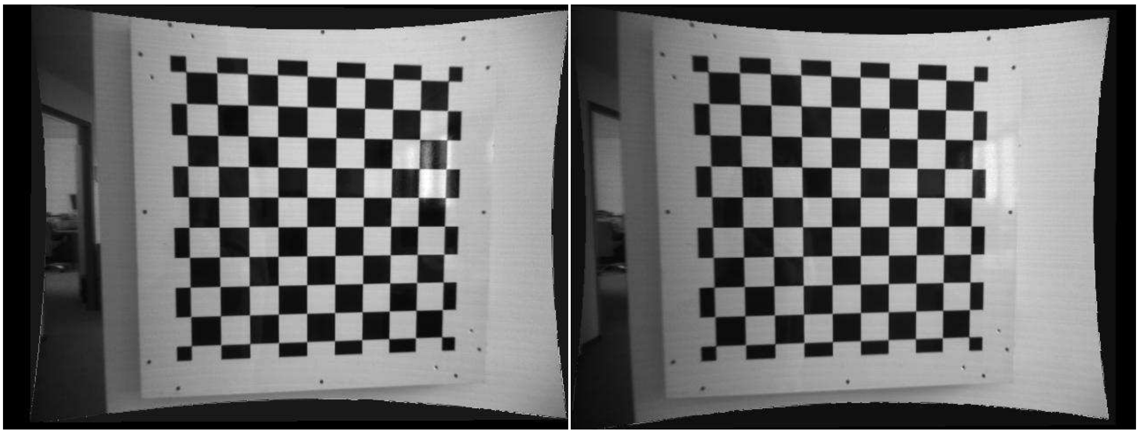
$$\begin{aligned} X &= \frac{B}{d} u' \\ Y &= \frac{B}{d} v' s_{vu} \\ Z &= \frac{B}{d} f_u \end{aligned} \tag{2.12}$$

where $s_{vu} = s_v / s_u$, $u' = (u_l - u_0)$ and $v' = (v_0 - v_l)$. Reorganizing the coordinates in vectors $(X, Y, Z)^T$ and $(u', v', d)^T$ the the triangulation function $\boldsymbol{g}$ is defined as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{g}(u_l, v_r, d)^T = \frac{B}{d} \begin{bmatrix} (u_l - u_0) \\ (v_0 - v_l) s_{vu} \\ f_u \end{bmatrix} \tag{2.13}$$



(a) original raw stereo pair images obtained from the cameras.



(b) rectified stereo pair.

Figure 2.6: Rectification of image pairs.

2.3.3 Calibration and Rectification

A standard stereo configuration is usually desired when implementing stereo matching algorithms. In a *standard stereo configuration* the epipolar line of any world point captured by both cameras is parallel to the scanlines of the left and right images. This means that the cameras must be arranged parallel to each other. Nevertheless, a physical lateral arrangement of the cameras is not enough to obtain accurate 3D information. This is because:

- The accurate physical positioning and orientation of the cameras in a left/right configuration is very difficult to achieve.

- The perspective camera is just an approximation to the optimal pinhole model. The optical system introduces non-linear distortions in the image which penalize the epipolar constraint, i.e. the epipolar lines are not found along the image scanlines but are distorted into curves.

- The real value of the parameters of the camera, such as focal length and pixel size are just approximate values and might deviate from the technical specifications of the manufacturer of the cameras.

In order to obtain the standard stereo configuration all these parameters must be known. Calibration is the process of measuring the internal parameters of the camera. The process of remapping the raw image, in order to obtain an undistorted image which meets the camera parameters obtained in the calibration phase, is called rectification. In multi-camera systems additional steps are considered:

- the calibration process also includes the measurement of the relative camera poses; and
- the rectification process also includes the remapping of the images, to impose the collinearity of the epipolar lines with the scanlines of the images.

Since the parameters of a stereo platform generally do not change over time, calibration is an off-line process in which computation time is not a constraint. Rectification, on the other hand, must be performed with every acquired image, and therefore, time is a factor to consider in real-time applications.

The literature on calibration and rectification is quite extensive, and many methods have been proposed. A discussion and review of calibration and rectification methods for multi-camera systems can be found in [WW03]. The calibration method used in the remainder of this dissertation is the solution of Bouguet [Bou00], based on the publications by Heikkilä and Silvén [HS97] and Zhang [Zha99]. The method requires images of a planar calibration rig of known geometry as shown in Figure 2.6. Some parts of the algorithm require an iterative optimization. With a two-camera setup, a total of 16 parameters are estimated by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\|\boldsymbol{m}_{ij}-\breve{\boldsymbol{m}}(\boldsymbol{K},\boldsymbol{k},\boldsymbol{R}_i,\boldsymbol{t}_i,\boldsymbol{M}_j)\|^2, \tag{2.14}$$

where $\boldsymbol{m}_{ij}$ is a measured feature point j of the calibration rig in image i, $\boldsymbol{M}_j$ is the corresponding known 2D world point of the calibration rig, and $\breve{\boldsymbol{m}}(\boldsymbol{K},\boldsymbol{k},\boldsymbol{R}_i,\boldsymbol{t}_i,M_j)$ is the distorted projection of point $\boldsymbol{M}_j$ into image i. The matrix $\boldsymbol{K}$ is the same of Equation 2.7, $\boldsymbol{R}_i$ and $\boldsymbol{t}_i$ are the 3×3 rotation matrix and three-dimensional translation vector of the camera with respect to the calibration rig in image i, and $\boldsymbol{k} = (k_1, k_2, k_3, k_4, k_5)^T$ is a five-dimensional vector of distortion coefficients. The first three coefficients count for radial distortion while k_4 and k_5 are the tangential distortion coefficients. A real undistorted normalized image point $(\bar{u}, \bar{v})^T$ is distorted to the normalized point $(u, v)^T$ according to the following equation:

$$\begin{aligned} u &= \bar{u}\left[1+k_1r_p+k_2r_p^2+k_3r_p^3\right]+2k_4\bar{u}\bar{v}+k_5\left(r_p+2\bar{u}^2\right) && (2.15)\\ v &= \bar{v}\left[1+k_1r_p+k_2r_p^2+k_3r_p^3\right]+2k_4\bar{u}\bar{v}+k_5\left(r_p+2\bar{v}^2\right), && (2.16)\end{aligned}$$

where $r_p = \bar{u}^2 + \bar{v}^2$.

Equation 2.14 implies a non-linear minimization problem, which is solved by Levenberg-Marquardt optimization [Lev44].

The rectification requires only a remapping of the image, i.e. each pixel in the image is displaced to a new position in the image space. An example of image rectification for a stereo pair is shown in Figure 2.6.

2.4 Image Primitives and Correspondence

One of the main problems of Computer Vision is the establishment of correspondences between images. The correspondence problem consists of finding a given pattern in multiple images. In most applications the real interest lies in finding the pixel coordinates of the *same world point* in multiple images, i.e. the correspondence of image points which are projections of the same point in space. Once the correspondence problem is solved, more specific geometric information about the world can be obtained. For example, given the image position of the same world point in the left and right rectified images of a calibrated standard stereo camera, the corresponding 3D position of the world point is obtained using Equation 2.12.

The main problem in correspondence is how to infer the motion of 3D geometric structures given only the measurement of their reflected or emitted light intensity. It can be assumed that a point in the left and right images having the same intensity value correspond to the projection of the same world point. Finding solution based on this observation will fail for a number of reasons. The first reason is that given the small range of intensity value of the image (typically encoded in 8 bits, i.e. $[0, 255]$), each intensity value of the sought point is expected to appear multiple times within the image (assuming an equal distribution of intensity values in a VGA image, the same brightness value is expected to appear 1200 times).

A second problem is the noise affecting the image brightness. The effects of noise can be reduced by attaching a support region around the point of interest. But an exact solution is still very unlikely to be found. It is therefore more convenient to define some discrepancy function and search for those image regions which minimize it. We address this point later in this section.

2.4.1 Translational Motion Model

The simplest image correspondence model is obtained assuming that the whole support region of the reference point moves constantly. This is valid actually only for those world regions which are flat, parallel to the image plane and which move parallel to it. But even when these assumptions are not fulfilled, almost any motion of any structure form is well approximated through this model, if the camera motion is small.

Let us suppose that both images are taken from infinitesimally close vantage points. Let us rewrite $I_1(\boldsymbol{p_1})$ and $I_2(\boldsymbol{p_2})$ as $I(\boldsymbol{p}(t), t)$ and $I(\boldsymbol{p}(t + dt), t + dt)$ respectively, i.e. the continuous version of the discrete equations. Let us assume that the brightness remains constant over time, i.e. $I(\boldsymbol{p}(t), t)$ is constant for all t. Now since dt is an infinitesimal increment we can rewrite $\boldsymbol{p}(t + dt) = \boldsymbol{p}(t) + \boldsymbol{v}\, dt$ where $\boldsymbol{v}$ is a velocity vector. Then we have

$$I(\boldsymbol{p}(t), t) = I(\boldsymbol{p}(t) + \boldsymbol{v}\, dt, t + dt). \tag{2.17}$$

Applying Taylor series expansion around $\boldsymbol{p}(t)$ to the r.h.s. and neglecting the sec-

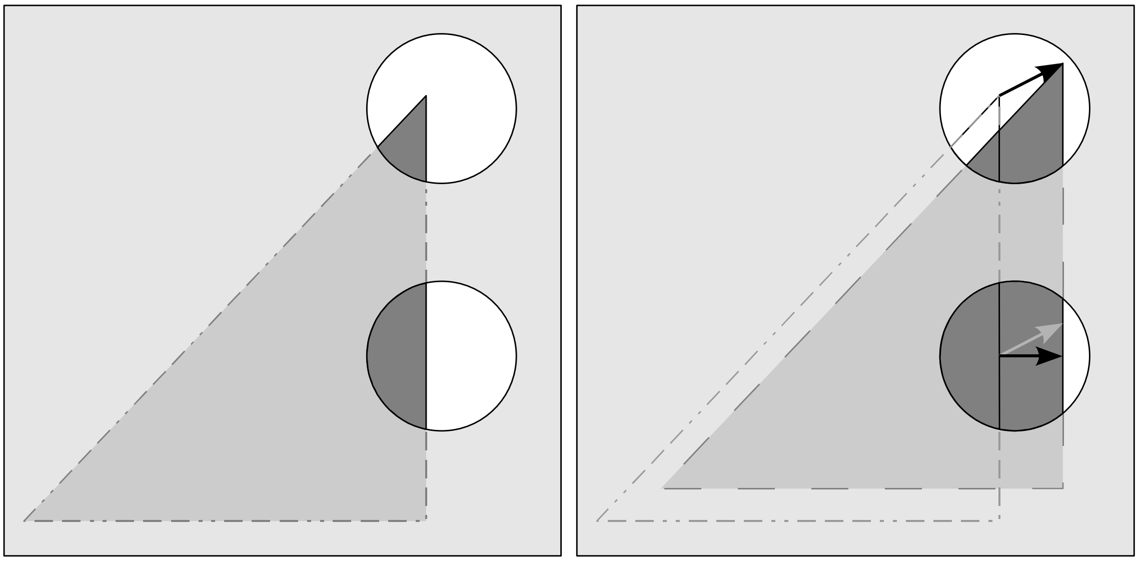


Figure 2.7: The aperture problem. Two parts of a figure are observed through apertures. Although the triangle moves diagonally, only a horizontal motion can be observed from the lower aperture. When enough 2D information is available, the 2D motion can be measured (upper aperture).

ond and higher-order terms we obtain the *brightness constancy constraint*

$$\boldsymbol{\nabla I}^T \boldsymbol{v} + I_t = 0 \tag{2.18}$$

where

$$\boldsymbol{\nabla I} = \begin{bmatrix} \frac{\partial I}{\partial u}(\boldsymbol{p}, t) \\ \\ \frac{\partial I}{\partial v}(\boldsymbol{p}, t) \end{bmatrix}, \quad I_t = \frac{\partial I}{\partial t}(\boldsymbol{p}, t). \tag{2.19}$$

The vector $\boldsymbol{\nabla I}$ is the frame spatial derivative or image gradient and the scalar I_t is the temporal derivative of $I(\boldsymbol{p}, t)$. Observing that the velocity vector $\boldsymbol{v} = (du/dt, dv/dt)$, Equation 2.18 can also be written as:

$$\frac{\partial I}{\partial u}\frac{du}{dt} + \frac{\partial I}{\partial v}\frac{dv}{dt} + \frac{\partial I}{\partial t} = 0 \tag{2.20}$$

Equations 2.18 and 2.20 involve two unknowns and one constraint, therefore there are infinitely many solutions for $\boldsymbol{v}$ that satisfy the equation. This is called the *aperture problem* in the literature. The name comes from the geometrical interpretation which exemplifies this fact and which is shown in Figure 2.7. If the solution is found in the direction of the image gradient $\boldsymbol{\nabla I}$, the resulting vector is called *normal flow*. Geometrically, the normal flow is the minimum norm vector that satisfies the brightness constancy constraint and represents the projection of the real motion vector onto the gradient direction. It is given by

$$\boldsymbol{v}_n = -\frac{I_t \boldsymbol{\nabla I}}{\|\boldsymbol{\nabla I}\|^2} \tag{2.21}$$

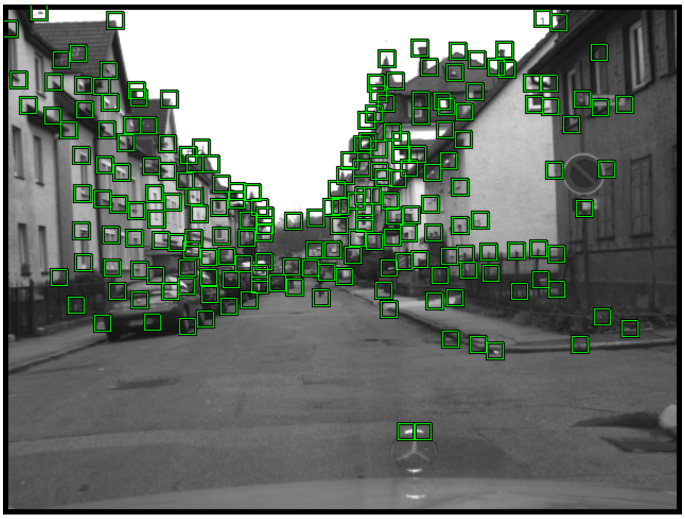


Figure 2.8: Features selected for tracking. Observe that every selected window presents a high textureness and regions of constant intensity values are avoided.

For the computation of the real motion vector, more constraints are required. A support region around the point provides enough information to find a solution if i) the region contains enough "information" and, ii) the motion of the region is constant. The second condition is required in order to provide enough constraints on $\boldsymbol{v}$. The first condition implies that the region must contain enough "texture". As it was also remarked before it is quite unprovable that the brightness remains constant over time because of image noise. Equation 2.17 can be rewritten to consider noise as

$$I(\boldsymbol{p}(t), t) = I(\boldsymbol{p}(t) + \boldsymbol{v}\, dt, t + dt) + \eta \tag{2.22}$$

where η is a noise term. A solution is then found minimizing the sum of the squared residuals, which leads to

$$E(\boldsymbol{v}) = \sum_{\tilde{\boldsymbol{p}} \in W(\boldsymbol{p})} \left(\boldsymbol{\nabla I}^T(\tilde{\boldsymbol{p}})\, \boldsymbol{v} + I_t(\tilde{\boldsymbol{p}}) \right)^2 \tag{2.23}$$

where $W(\boldsymbol{p})$ is the support region around $\boldsymbol{p}$. The minimum can be found in the least-square sense by finding the zero derivatives of $E(\boldsymbol{v})$ w.r.t. $\boldsymbol{v}$, i.e.

$$\begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_u I_v & \sum I_v^2 \end{bmatrix} \boldsymbol{v} + \begin{bmatrix} \sum I_u I_t \\ \sum I_v I_t \end{bmatrix} = 0 \tag{2.24}$$

or in matrix form,

$$\boldsymbol{G}\boldsymbol{v} + \boldsymbol{e} = 0. \tag{2.25}$$

If the matrix $\boldsymbol{G}$ is invertible a solution can be found for $\boldsymbol{v}$

$$\boldsymbol{v} = -\boldsymbol{G}^{-1}\boldsymbol{e}. \tag{2.26}$$

If the vector $\boldsymbol{p}$, for which $\boldsymbol{v}$ was computed is some fixed integer position in the image, then the motion vector is called *optical flow*. If instead $\boldsymbol{p}$ is computed repeatedly as a particle which moves trough the image domain, the motion is called *feature tracking*. When $\boldsymbol{G}$ is singular, then no solution can be found by 2.8. This happens when the intensity variations in the support region varies in only one dimension (i.e. $I_u = 0$ or $I_v = 0$) or there is no variation at all (i.e. $I_u = 0$ and $I_v = 0$). In fact, Equation 2.26 can be solved reliably if the matrix $\boldsymbol{G}$ is well-conditioned and above the image noise level. The matrix $\boldsymbol{G}$ is above the image noise level, if both eigenvalues are large. The conditioning requirement means that the eigenvalues cannot differ by several orders of magnitude. Both requirements are normally implemented by just checking if the smallest eigenvalue is larger than a predefined threshold. This is normally enough to check for both requirements since the maximal eigenvalue is actually upper bounded because of the limited intensity range of the image. Therefore, the difference between eigenvalues is also finite. Observe that the matrix $\boldsymbol{G}$ gives a measure of the textureness contained in the support region. This method is used by the KLT tracker [LK81] [ST94] [TK91], which is used in the experimental results of this dissertation for the computation of feature tracking. Another possibility is to threshold the quantity $det(\boldsymbol{G}) + k\ tr(\boldsymbol{G})$, where k is some small value. This variation is known as Harris corner detector [HS88].

Figure 2.8 shows an example of feature selection. The matrix $\boldsymbol{G}$ is computed for every point of the image forming a descending sorted list according to the smallest eigenvalue of the matrix. The top 200 features which satisfy a minimal distance constraint are chosen as features and are shown in the figure.

The computation of image velocities with the method described above is usually expensive. It requires the computation of spatial and time derivatives as well as some matrix operations. An alternative is to define some function, which measures the discrepancy between support regions, and then find the displacement which minimizes it. A typical dissimilarity measure is the *Sum of Squared Differences* (SSD) criterion. Considering Equation 2.22, the sum of the squared of the residua is minimized this way, and therefore,

$$SSD = \sum_{\tilde{\boldsymbol{p}} \in W(\boldsymbol{p})} \left(I(\boldsymbol{p}(t), t) - I(\boldsymbol{p}(t) + \boldsymbol{d}, t + dt) \right)^2 \tag{2.27}$$

where $\boldsymbol{d} = \boldsymbol{v}\,dt$ is the displacement. The result is found as the displacement $\boldsymbol{d}$ that minimizes the SSD. Observe that Equations 2.27 and 2.26 are related since $\boldsymbol{v}\,dt = (-\boldsymbol{G}^{-1}\boldsymbol{e})\,dt$ is a first order approximation of $\boldsymbol{d}$. The SSD is the correlation function of the stereo algorithm used in the experimental results in Chapter 8.

Other dissimilarity criteria are also used in the literature. The *Sum of Absolute Differences* (SAD) is an alternative to the SSD. Locally scale versions as well as zero-mean normalized version of SAD and SSD are also commonly used. The *Zero-mean Normalized Cross Correlation coefficient* (ZNCC) and the pseudo normalized correlation [Mor80] are also examples of similarity functions. All these functions are listed and evaluated for performance and robustness in [AG92] and [Bad02].

2.4.2 Affine and Projective Motion Models

More advanced motion models have been proposed. It is actually rather improbable that all the points corresponding to the support region have the same transformation, which is a requirement for finding a solution. A better model would be to consider some deformation of the support region. From Equation 2.17 a general transformation model is expressed as

$$I(\tilde{\boldsymbol{p}}(t), t) = I(\boldsymbol{h}(\tilde{\boldsymbol{p}}(t), \boldsymbol{\alpha}), t + dt) \; \forall \tilde{\boldsymbol{p}} \in W(\boldsymbol{p}) \tag{2.28}$$

where h is the function of motion model and $\boldsymbol{\alpha}$ captures all the parameters corresponding to the model (e.g. in the translational case $\boldsymbol{h}(\boldsymbol{p}, \boldsymbol{\alpha}) = \boldsymbol{p} + \boldsymbol{d}$ and $\boldsymbol{\alpha} = \{\boldsymbol{d}\}$).

In the *affine transformation model*, the motion of every point in the support region depends linearly on its location with respect to the reference point $\boldsymbol{p}$. So $\boldsymbol{h}(\boldsymbol{p}, \boldsymbol{\alpha}) = \boldsymbol{A}\boldsymbol{p} + \boldsymbol{d}$ (and $\boldsymbol{\alpha} = \{\boldsymbol{A}, \boldsymbol{d}\}$) where $\boldsymbol{A}$ is a 2×2 deformation matrix. This model approximates a motion of a planar patch with arbitrary translation, arbitrary rotation about the optical axis, and small rotation about any other axis. The affine transformation model is used by Shi and Tomasi [ST94] for monitoring the quality of a track for non-consecutive frames.

An arbitrary rotation and translation of planar surface is modeled by the *projective motion model*, which applies an affine model to the homogeneous coordinates of the points. In this case, $\boldsymbol{h}(\boldsymbol{p}, \boldsymbol{\alpha}) = \boldsymbol{H} \cdot \boldsymbol{p}$ where $\boldsymbol{H}$ is a 3×3 matrix defined up to a scale factor. The projective motion model is also called a *homography motion model*, since $\boldsymbol{H}$ describes the homography between $\boldsymbol{p}(t)$ and $\boldsymbol{p}(t + dt)$.

Observe that the complexity in the derivation of the solution for the affine and projective motion models increases and the corresponding implemented algorithms are much slower (in comparison to the translational model). The readers are referred to the bibliography cited in the next section for further details.

2.5 Literature

Most books of Computer Vision describe in detail all or at least parts of the topics addressed in this chapter. As examples, I cite to "An Invitation to 3-D Vision" of Yi Ma *et al.* [YSJS04], "Computer Vision, A Modern Approach" of Forsyth and Ponce [FP03] and "The geometry of multiple images" [FL01] of Faugeras and Luong, which are the main bibliography sources used for this chapter.

Chapter 3

Overview of the Proposed Approach

3.1 Introduction

This chapter summarizes an approach for the estimation of *ego-motion* and for the *simultaneous estimation of position and velocity of single points*, providing low-level information to more complex visual perception systems, such as autonomous navigation or collision avoidance. Treating such a process as a low-level task should not be surprising since, from the biological point of view, motion detection is a direct experience uniquely specified by the visual system.[1] The further segmentation and integration of the information provided by this approach is here referred to as high-level vision, which must introduce some additional knowledge and intelligence to carry out a cognitive process (e.g. for statements such as "a bicyclist is approaching from the left and a collision is imminent").

In the next section, a statement of the problem is formulated. Section 3.3 reviews some alternative sensors for the computation of ego-motion and in Section 3.4 the proposed approached is described.

3.2 Problem Statement

The motion of an object observed by a stationary observer produces a spatial-temporal change in the light distribution on some regions of the retinal image. If we want to detect the motion of an object and estimate its position and velocity, the focus of interest must be given on those image regions. The rest of the retinal image is assumed to be the environment or background and remains static. Nevertheless, when the observer is in motion, the projected background into the retinal image changes as well. The problem becomes more complex to solve because the system must discriminate between the motion caused by the observer and the motion caused by an independent moving object.

---

[1] Neurons in the middle temporal visual area integrate motion signals over large regions of visual space and respond to motion in their preferred direction, e.g. these neurons register motion information *per se*. More advanced perception activities are distributed over many areas of the brain, each extracting somewhat different information from the retinal image [SWB04].

We define *independent 3D motion* as the rigid or articulated temporal change of the position of an object with respect to an environment which is considered static. Nevertheless, and citing Daniilidis and Nagel [DN93], "what we can measure in the image are apparent shifts or velocities of gray-value structures which are approximations to the geometrically defined displacements or velocities of the projections of three-dimensional features", i.e. what we can measure is motion relative to the camera and not to the background. Nevertheless, if the camera motion is known, it can be extracted from the observed motion to reveal independently moving objects. The real camera motion is not available, but can be estimated from the images.

After these preliminary remarks the the statement of the problem can be formulated:

> *Given as input a sequence of images of a freely moving calibrated stereo camera platform, compute in real-time the ego-motion of the camera as well as the 3D position and 3D velocity of world points of the environment.*

The real-time requirement is a factor which highly depends on the hardware used and on the implementation of the algorithm. Nevertheless we define *real-time applications* in this dissertation as those applications which are able to provide new information at a frequency of at least $10\,Hz$ by using off-the-shelf hardware (e.g. Pentium M, 2 GHz). There are some additional requirements which are normally not mentioned because of their obviousness, but which I would like to address.

- The images composing the sequence must be of a fair quality, i.e. enough information is expected to be obtained from the images. The current implementation of the method includes actually an extrapolation (prediction) module in case not sufficient information can be obtained from the images. But computing the ego-motion of the camera from a sequence of blank images is a problem we do not pretend to solve. In this dissertation the images have a "fair quality" when the stereo and the tracking algorithms can compute reliable information for at least 50 points of the image.

- The intersection of the FOV corresponding to consecutive images must not be empty, otherwise no tracking is possible. In fact, a smooth motion between images is desired since the correspondence problem is easier to solve and it allows more accurate predictions of motion.

- The transformation of the static background w.r.t. the camera must be that of a rigid body motion.

When any of these requirements is not fulfilled, it is rather improbable that the visual estimation of motion will be successful. When implementing a real robust system, this problem can be overcome by using motion data from other sources, as for example inertial sensors.

3.3 Sensors for Ego-motion Computation

The ego-motion of a camera mounted on a vehicle can be obtained from other sources than images. The Global Positioning System (GPS) seems to be a valid alternative to the visual system. The accuracy of current GPS technology like WAAS [Adm06] and DGPS [Adm05] is up to 1.5 to 2 meters. As this dissertation shows, the accuracy achieved for short distances using the visual approach is several hundreds times better than GPS. Furthermore, GPS is a positioning system, and therefore orientation must be computed by differentiation over time, the latter being quite sensitive to measurement errors. GPS is, this way, more appropriate for navigation purposes.

Another alternative solution is given by the inertial sensors of the vehicle such as velocimeters and steering angle sensors. There are a number of drawbacks when using inertial sensors for the computation of ego-motion. Usually, not all components of motion are covered with sensors. Commonly only a velocimeter and a steering angle sensor are available. The motion obtained with these two sensors is incomplete since pitch and roll rotation as well as vertical translation are unknown. Another disadvantage is that the motion computed with inertial sensors is the vehicle motion and not the camera motion. In order to be able to estimate the camera motion, the transformation between both coordinate systems must be known, i.e. an additional calibration procedure is required. The inertial sensors themselves require a calibration procedure. When computing visual ego-motion the motion is already computed from the camera point of view. The independence of inertial sensors makes the approach also appropriate for other applications areas where no inertial sensors are available (in the experimental results of Chapter 8 are shown some examples). Nevertheless, when available, the data provided by the inertial sensors are not worthless but can be fused with the visual odometry information for a more robust motion estimation as will be shown in Section 7.7.

3.4 Proposed Approach

A block diagram of the proposed approach is shown in Figure 3.1. The system is composed of three main parts: registration of image features, ego-motion computation and Kalman filtering.

The first block, "Registration of Image Features" receives as input the rectified images and provides as output a list of tracked image features with their corresponding 3D position. Feature tracking is computed using the current and previous left images for those points already being tracked by the system. Disparities between the left and right images are only computed for the image features with tracking information. Triangulation is then performed and a list with the tracked points for the current frame is generated. The list is added to a table were the last m list of tracked 3D points are stored.

The Ego-Motion block receives as input the list of tracked points (and eventually the information provided by the inertial sensors of the vehicle) and provides as output the translation and rotation of the camera platform between current and previous times. The output is added to a list of motion steps which is used in subsequent cycles. A motion step stores the rotation and translation of the camera between two consecutive frames. For the visual computation of ego-motion four mains steps are carried out in an iterative way: motion prediction, application of the Smoothness Motion Constraint (SMC), absolute orientation computation and motion integration. An ego-motion prediction is required in order to apply the Smoothness Motion Constraint (SMC). The SMC is a powerful constraint for the rejection of outliers and for the assignment of weights to the measured 3D points. Once the SMC has been applied, the absolute orientation problem between the clouds of points is solved in the least squares sense. These three steps are carried out not only between the current and previous clouds of 3D points but iteratively between the current cloud and every cloud obtained in a predefined interval of time. Each iteration provides this way a motion hypothesis which must be integrated into the final motion estimation. The iteration stops when the number of points in the clouds is not sufficient in order to compute the absolute orientation, or the predefined maximal amount of integration cycles is achieved. We call this approach Multi-Frame Estimation (MFE) in contrast to the Two-Frame Estimation (TFE) which considers only the current and previous frames for the computation of ego-motion.

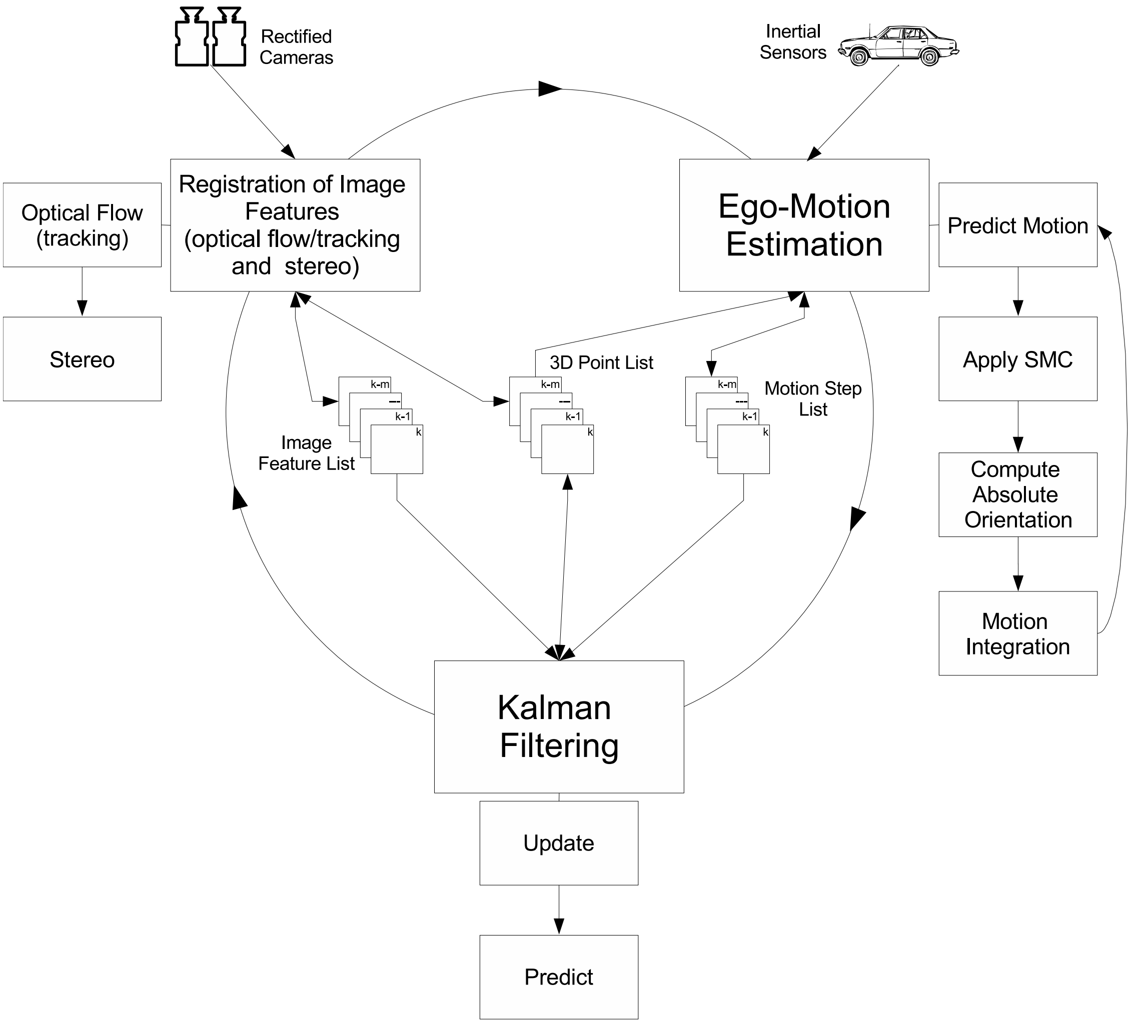


Figure 3.1: Block Diagram of the Proposed Approach

In order to better estimate the 3D position of the points of the environment, an iterative refinement of 3D position is achieved by means of Kalman Filters (KF). The dynamic of a static world point is given by the rigid body motion of the background, which at the same time corresponds to the inverse camera motion. Any deviation from this dynamic implies an independent motion, which is estimated by the differentiation of the point position over time. For this purpose, Kalman filters provide an iterative refinement of 3D point position and 3D point velocity. This information is used to update the list of 3D points, which is used in the next system cycle for a better computation of ego-motion. The Kalman filter provides also a prediction of the next state estimate, which are used by the optical flow and stereo algorithms in the next system cycle.

3.4.1 The *Chicken-And-Egg* Problem

The computation of ego-motion requires a list of static 3D points of the environment observed at different time instances. A tracked point is labeled as *static* if its observed motion is only because of measurement noise. The Kalman filter model estimates the 3D position as well as the 3D velocity of the tracked points, so it is tempting to ask the KF for the estimated velocity of a point before using it as a measurement for the computation of ego-motion. Nevertheless, Kalman filters requires first the ego-motion parameters before updating the state of a point and being able to tell its current velocity. Problems involving systems with reciprocal requirements like this are called *chicken-and-egg problems*. A solution to this kind of problems is given by starting with a very rough estimate and iteratively giving reciprocal feedback of the other system outputs until a convergence is found. We avoid the reciprocity by applying the Smoothness Motion Constraint (SMC) as commented in the previous section and shown in detail in Chapter 7.

3.4.2 The Positive Feedback Effect

Suppose that for some reason ego-motion was wrongly estimated with the current list of 3D points. This can happen if for example, not enough static points could be tracked or stereo failed in estimating an accurate 3D position for the tracked features. In this case, the KF block will receive as input an incorrect estimation of ego-motion. KF will propagate this error to the filtered position and velocity of the points. Position and velocity will be incorrectly estimated and the 3D Point list will be updated with wrong estimates. This means that in the next system cycle, the ego-motion estimation will also fail even if the stereo feature tracking are correctly computed, because the filtered 3D positions stored in the list are incorrect. The situation becomes worse with every cycle and the system is not able to work properly any more. This is an example of the *positive feedback effect*. In the proposed approach, the positive feedback effect can be avoided by defining a *firewall*. A firewall defines a limit for the integration of information. For example, the firewall can be so defined to avoid that the filtered 3D position of young feature points[2] be stored in the 3D Point List.

---

[2] The age of a feature point is the number of times that the feature could be correctly tracked since its first appearance. The age is given in frames.

A relative old feature point will have an already converged state in the KF model and a more stable behavior in front of wrong estimates. This effect will be analyzed later in Chapters 7 and 8.

Chapter 4

Kalman Filter-based Estimation of 3D Position and 3D Velocity

4.1 Introduction

The objective of this chapter is the estimation of the position and velocity of world points in 3D Euclidean space, and the recursive improvement of these estimates over time. A continuous improvement of the estimates is motivated by the noisy nature of the measurements and, therefore, Kalman filters happen to be appropriate to address this problem.

This chapter is organized as follows. Section 4.2 reviews the literature on tracking with Kalman filters and alternative approaches for independent motion estimation. In Section 4.3, the Kalman filter model is presented, and the system and measurement equations are derived. Section 4.4 deals with the initialization of the filter and with the computation of the Cramér-Rao lower bound. Section 4.5 carries out some simulations for validating the filter. The last section summarizes the chapter. Some parts of this chapter have been published in [Rab05] [FRBG05] and [BFRG06].

4.2 Literature Review on 3D Object Tracking

In order to estimate the velocity of a world point, the system must observe its change of position over time. The point position is obtained with multi-ocular platforms, which allow the instantaneous extraction of 3D position through triangulation. The time component is obtained by finding correspondences in consecutive frames. The correspondences are found between image points (i.e. optical flow or normal flow) or at the level of objects, which requires the previous segmentation of stereo points and the further tracking of objects. This last strategy is commonly approached by an orthographical projection of the 3D points into an evidence-grid-like structure, and by grouping the projections according to their vicinity [MM96]. This method has difficulties in segmenting distant objects and in separating distinct objects which are close together.

Normal flow methods have some advantages with respect to optical flow methods, since the correspondence problem is reduced. Argyros and Orphanoudakis [AO97] have proposed a method based on normal flow fields and Least Median of Squares (LMedS), which estimates simultaneously ego-motion and independent 3D motion. Morency and Darrell [MD02] have also proposed a method for pose estimation based on normal flow and the Iterative Closest Point algorithm [BM92]. Nevertheless, normal flow is less informative compared to optical flow, since it reflects only the motion in the direction of the image gradient.

Methods based on optical flow have been widely proposed. One of the first attempts to fuse stereo and optical flow information was studied by Waxman and Duncan in [WD86], exploiting the relationship between 3D motion and image velocities with stereo constraints. Kellman and Kaiser [KK95], Heinrich [Hei02] and Mills [Mil97] also make use of such geometric constraints to detect independent motion. Demirdjian and Horaud [DH00] propose a method for the estimation of the ego-motion and the segmentation of moving objects. Demirdjian and Darrel [DD01] estimate rigid motion transformation mapping two reconstructions of a rigid scene in the disparity space (which they called d-motion).

4.2.1 Literature Based on Kalman Filters

Kalman filters for object tracking are used everywhere in the computer vision literature. Some of the most significant methods are shortly described here.

Dang *et al.* [DHS02] fuse optical flow and stereo disparity using Kalman filters for object tracking. The detection and segmentation of the object must be provided beforehand. A test based on the Mahalanobis distance is performed, in order to eliminate those points with incoherent motion and which possibly do not belong to the object being represented by the rest of the observations.

Suppes *et al.* [SSH01] also estimate Kalman filtered landmark positions obtained with stereo. The projection of the probability density function (p.d.f.) of the points on a depth map allows the accurate detection of stationary obstacles. Phantom objects are also less probable to appear this way, since the lifetime of a false correspondence is normally very short and, therefore, its covariance matrix is large. If the covariance matrix of a point is large, its registration on the depth map has little impact. The ego-motion required is obtained from the inertial sensors of the robot.

Sibley *et al.* [SMS05] use Kalman filters for modeling the dynamics of distant points measured with stereo. The dynamics of the point is left unspecified and is assumed to be given. An analysis of the bias in the triangulated 3D points is carried out and a correction using a second order approximation of the triangulation function is proposed.

Lee and Kay [LK91] estimate object motion using Kalman filters in stereo image sequences. The position and orientation as well as the translational and rotational velocities of the object are estimated. The paper first presents the camera geometry and then derives a simplified linear expression relating the measurement noise of a feature point in a stereo image and its position error induced in 3D space. A differential rotation matrix is defined and a least squares expression is found. A linear measurement equation for the Kalman filter is found using quaternions for representing rotation and differential rotation.

Rives *et al.* [RBE87] present one of the first Structure-from-Motion algorithms using Kalman filters and normal flow using monocular images. The paper derives the equations of image velocity field given the motion of the camera and then eliminates rotation from the equations in order to simplify the analysis. A solution for the depth of the tracked point, given the velocity field, is obtained that way. The robot displacement is refined minimizing a cost function relating the normal flow and the motion parameters from the inertial sensors.

Matthies *et al.* [MKS89] propose an iconic disparity map estimation using Kalman filters assuming a purely lateral translational monocular camera. An analysis of the accuracy of the estimated distance of 3D points regarding the direction of the camera motion is carried out. The results that are obtained are the relative precision of stereo and depth-from-motion. The authors emphasize the importance of taking into account the off-diagonal elements of the state covariance matrix in order to model smoothness in the disparity map. A feature based model is also presented and compared with the iconic model. The feature based approach has a faster convergence rate, because it keeps the disparity and the sub-pixel-position of the feature as state elements, while the iconic model only keeps the disparity. A comparison with stereo shows the interesting result that processing the intermediate frames (as the camera moves laterally) does not improve the precision, if compared to computing stereo on the first and last frames of the images sequence.

Zhang and Faugeras [ZF91] present a complete framework for the segmentation of objects and the computation of 3D motion using a trinocular stereo camera system. Correspondences for image lines are found in space and time. The line segment is accordingly represented with a mid-point and a direction vector. An appropriate covariance matrix is considered this way. Kalman filters are used to estimate angular velocity, translational velocity and translational acceleration of the detected line segments. The Mahalanobis distance between predicted line segment and measured line segments are used to select possible matches. A bucketing technique is also used to reduce, even more, the number of hypotheses. Every remaining hypothesis is then tracked in order to observe its dynamics. The Mahalanobis distance is once again used to eliminate features incorrectly tracked (the authors assign a "support of existence" to every tracked line segment). The grouping of line segments into objects is also performed with the Mahalanobis distance between the motions of two line segments. The covariance matrix of a detected objects is computed and used to iteratively check if other line segments also belong to the object.

Altunbasak *et al.* [ATB95] estimate 3D point motion with a maximum likelihood approach and Kalman filters in stereo sequences. Kalman filters are used to model point position, translation velocity, translation acceleration, rotation, angular velocity and precession. Stereo and motion are fused in this way by maximizing the probability that the estimated motion and disparity conform to the observed frames. The conditional probability distribution is modeled as a Gibbs distribution. The algorithm then iterates between the Maximum Likelihood step and the Kalman filter step until the maximum likelihood cost can no longer be reduced.

Yao and Chellappa [YC94] present a method for tracking features using Kalman filters in an image sequence. The state model considers image position, image velocity and rotation of the features. The Mahalanobis distance is used to choose potential feature points in the next image. The zero-mean normalized cross-correlation function (ZNCC) is applied for matching feature points. New image features are only tracked if a minimum distance criterion is fulfilled.

Hung and Tang *et al.* [YCS$^+$95] [CYSZ99] detect and track multiple moving objects computing stereo and optical flow in left and right images. A mutually-supported consistency constraint is used to reduce errors in the feature matching. Random Sample Consensus (RANSAC) [FB81] is then used to find clusters of points with similar motion, where similarity is defined as the inverse Euclidean distance between point position and predicted point position. Kalman filters are used on each detected cluster in order to track each object in the scene. Angular velocity, angular acceleration, point of rotation, translational velocity and translational acceleration constitute the state vector. When tracking a feature, the prediction for the corresponding cluster is used to predict the 2D image position and supports the correspondence.

Kalman filters are widely used in Self Localization And Mapping (SLAM) applications [DNC$^+$01]. Jung and Lacroix [JL03] describe a method for building digital elevation maps using stereo images. The Kalman filter is used to simultaneously refine estimates of ego-motion and 3D landmark position of world points. Only a sub-set of the dense output provided by the stereo algorithm is used as landmarks for the computation of ego-motion. The rest of the stereo output is used to build maps of the environment. The state vector includes the six motion parameters of the camera and the 3D position of every tracked point.

Matthies and Shafer [MS87] estimate landmark positions in a camera-centered coordinate system using Kalman filters. Ego-motion is computed from the 3D points obtained with stereo. The covariance matrix of each stereo point is used to compute a motion covariance matrix, which is next propagated to the covariance matrices of the landmarks. The update of the global robot position is carried out; concatenating the transformation matrices, and estimating the uncertainty of the global position by propagating the covariance matrices of the incremental motions into a covariance of the global position.

4.2.2 Alternative Methods for Object Tracking.

Kalman filters are not the only tool for combining stereo and motion components. Some alternative methods are briefly described here.

Liu and Skerjanc [LS93] present a method for finding stereo and motion correspondences using a coarse-to-fine strategy. Dynamic programming is used with a cost function including interline penalty, motion penalty and pyramid penalty components. Dynamic programming is applied in each level of the pyramid. The authors also point out some geometric relationships between motion and disparity.

Jenkin and Tsotsos [JT86] present a method for handling multiple matching hypothesis generated in a stereo image sequence. The paper describes some smoothness assumptions and defines constraint based on such assumptions. Features are tracked in 3D and multiple hypotheses are generated in a tree-like structure. Every node is assigned a label and some label-combinations are defined as to be incoherent in order to eliminate possible false correspondences.

Ho and Pong [HP96] present a method for matching features in two consecutive stereo images. Four match processes are integrated as a network. In the first step, features are extracted from the images. Second, multiple matching hypothesis are established for every feature. Third, to each potential match an initial probability is assigned. In the last step, the probabilities are updated iteratively by a relaxation labeling process.

Altunbasak *et al.* [ATB94] propose a framework for simultaneous motion and disparity estimation. Motion is estimated with 6 d.o.f. The Bayesian framework is presented and probabilities are modeled with Gibbs distributions. The method iterates between computing the MAP estimate of the disparity and segmentation fields, conditioned on the present motion parameter estimates, and the ML estimates of the motion parameters via simulated annealing.

Agrawal *et al.* [AKI05] present a complete framework for detecting independently moving objects. The main steps of the method are: 1. compute ego-motion, 2. warp previous image to current image according to the ego-motion, 3. compute sum of absolute differences of warped image with current image, 4. extract blobs from difference image, 5. track blobs in time. RANSAC is used to produce multiple hypotheses of motion. Every hypothesis is assigned a score depending on the error of the projection of the points, based on the current motion hypothesis. The motion hypothesis with larger vote is used as the starting point for a non-linear minimization using Levenberg-Marquardt optimization [Lev44]. The function to minimize is the projection error. The authors use the method of Demirdjian [DD01] called "d-motion" to compute the projection errors. Blob extraction and tracking are the last steps of this procedure.

Talukder and Matthies [TM04] use a similar method for the detection and tracking of independently moving objects. Independent motion is found by first computing the ego-motion of the camera with respect to the static scene, and then observing the difference between predicted and measured optical flow and disparity. These differences are thresholded in order to build a binary map. Moving objects are detected from the map as binary blobs of moving pixels. The segmentation of moving objects is performed with a simple algorithm based mainly on heuristics. This method requires dense optical flow and dense stereo computation.

Woelk and Koch [WK04] detect independently moving objects from the optical flow computed on a monocular camera and known ego-motion. The ego-motion of the camera is first improved by refining the initially provided essential matrix. Particle filters are used to select the regions in the image were optical flow must be computed, leading to a fixed number of optical flow calculations and reducing this way the computational burden.

4.3 The Kalman Filter Model

This section presents a model which allows to estimate the relative motion of world points relative to the observer. This is done by using the ego-motion of the camera platform to compensate for the motion of the world points. If a world point is static, its observed motion is described as the inverse of the camera motion. Otherwise, the point has an independent 3D motion, which the filter estimates. In the following subsections a vehicle-based coordinate system is assumed, i.e. the origin of the coordinate system moves with the observer.

4.3.1 Stochastic Models and Kalman Filters

Stochastic models are mathematical tools for problems involving random variables. Modeling a problem usually requires the design of a merit-function that measures the agreement between the output of a system and the model. The objective is to find the set of parameters of the merit-function for which the observed output can be achieved. When the parameters are time-dependent, they are called *the state of the system*, and the goal extends to finding their variation over time. Kalman filters [Kal60] are recursive estimators of the state of a system, using noisy data measured from the environment. Under certain assumptions Kalman filters are optimal estimators, in the sense that the estimation error is statistically minimized. Such assumptions include a Gaussian error distribution, the independence of the process and measurement noises, and the existence of a linear relation between the measurements and the state of the system.

The Kalman filter addresses the problem of estimating the state $x \in \mathbb{R}^n$ of a process that is governed by a stochastic difference equation of the form:

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{\rho}_k \tag{4.1}$$

with measurement $z \in \mathbb{R}^m$ where

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{\nu}_k. \tag{4.2}$$

$\boldsymbol{A}_k$ is a matrix relating the previous state $\boldsymbol{x}_{k-1}$ with the current state $\boldsymbol{x}_k$, $\boldsymbol{B}_k$ is an optional control input of the state of the system, and $\boldsymbol{\rho}_k$ represents the process noise. $\boldsymbol{H}_k$ relates the current state with the measurement $\boldsymbol{z}_k$, and $\boldsymbol{\nu}_k$ represents the measurement noise. The noise terms in both equations are assumed to be independent and with normal probability distributions

$$\begin{aligned} p(\boldsymbol{\rho}_k) &\sim N(0, \boldsymbol{\mathcal{Q}}) \\ p(\boldsymbol{\nu}_k) &\sim N(0, \boldsymbol{\mathcal{R}}) \end{aligned}$$

$\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{R}}$ being the process noise and measurement noise covariance matrices, respectively.

The Kalman filter is the optimal recursive linear estimator of $\boldsymbol{x}_k$ [May79]. However, the linearity assumed in the model and measurement equations are not satisfied for most real problems. For example, in the next section the measurements are not a linear relations of the state and therefore Kalman filters cannot be used directly. The solution is provided by the Extended Kalman filter (EKF), which linearizes the functions about the current mean and covariance. The linearized filter is not optimal any more, since there are non-linear filters which perform better than the KF [GA93]. Nevertheless, between all linear estimators the KF is still the best estimator. In the next section, the required model for estimating 3D position and 3D point velocity from stereo image measurements is derived. The extended Kalman filter equations are presented later in Section 4.3.5. Details about KFs and EKFs can be found in [GA93] [BSL93] [Lof90] and [May79].

4.3.2 Continuous Motion Model of 3D Position and 3D Velocity

In this section a continuous motion model for the description of the 3D point motion is derived. The camera motion model as well as the object motion model are also presented. With this basis, the system model equation is obtained. In the next section the continuous motion model will be discretized.

4.3.2.1 Camera Motion Model

Let us assume the camera moves continuously in a world coordinate system with translational velocity vector

$$\boldsymbol{v}_C = \left[\begin{array}{ccc} v_x(t) & v_y(t) & v_z(t) \end{array}\right]^T \tag{4.3}$$

and angular velocity pseudo-vector

$$\dot{\boldsymbol{\omega}}_C = \left[\begin{array}{ccc} \dot{\omega}_x(t) & \dot{\omega}_y(t) & \dot{\omega}_z(t) \end{array}\right]^T \tag{4.4}$$

The translational velocity is measured in meters per second. The angular velocity pseudo-vector specifies the angular speed and axis about which the camera is rotating. The angular speed is obtained as the magnitude of the pseudo-vector. The axis of rotation is obtained from its direction. The angular velocity is measured in radians per second.

The computation of the translation and rotation of the camera after a time $\Delta\mathrm{t}$ is obtained through integration over $\Delta\mathrm{t}$, i.e.

$$\boldsymbol{t}_C = \int_0^{\Delta\mathrm{t}} \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} dt \quad \text{and} \quad \boldsymbol{\omega}_C = \int_0^{\Delta\mathrm{t}} \begin{bmatrix} \dot{\omega}_x(t) \\ \dot{\omega}_y(t) \\ \dot{\omega}_z(t) \end{bmatrix} dt \tag{4.5}$$

Assuming that $\Delta\mathrm{t}$ is small and $\boldsymbol{v}_C$ and $\boldsymbol{\omega}_C$ remain constant over the interval $\Delta\mathrm{t}$, it is obtained

$$\boldsymbol{t}_C = \begin{bmatrix} v_x\Delta\mathrm{t} \\ v_y\Delta\mathrm{t} \\ v_z\Delta\mathrm{t} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\omega}_C = \begin{bmatrix} \dot{\omega}_x\Delta\mathrm{t} \\ \dot{\omega}_y\Delta\mathrm{t} \\ \dot{\omega}_z\Delta\mathrm{t} \end{bmatrix} \tag{4.6}$$

The above equations represent a translation and a rotation of the camera over the interval $\Delta\mathrm{t}$. The translation $\boldsymbol{t}_C$ is measured in meters and the rotation in radians. The direction of the pseudo-vector $\boldsymbol{\omega}_C$ defines the axis of rotation, and its magnitude defines the angle of rotation. A rotation matrix can be obtained as:

$$\boldsymbol{R}_C = \left[\begin{array}{ccc} \cos\alpha + v_1^2(1-\cos\alpha) & v_1 v_2(1-\cos\alpha) - v_3\sin\alpha & v_1 v_3(1-\cos\alpha) + v_2(\sin\alpha) \\ v_2 v_1(1-\cos\alpha) + v_3\sin\alpha & \cos\alpha + v_2^2(1-\cos\alpha) & v_2 v_3(1-\cos\alpha) - v_1(\sin\alpha) \\ v_3 v_1(1-\cos\alpha) - v_2\sin\alpha & v_3 v_2(1-\cos\alpha) + v_1(\sin\alpha) & \cos\alpha + v_3^2(1-\cos\alpha) \end{array} \right] \tag{4.7}$$

where $\alpha = \|\boldsymbol{\omega}_C\|$ is the angle of rotation and $(v_1, v_2, v_3)^T = \boldsymbol{\omega}_C/\alpha$ the unit direction vector.

4.3.2.2 Object Motion Model

This section proposed a continuous object motion model following a similar derivation as done by Lux [Lux00].

Let $(X, Y, Z)^T$ represent the coordinate vector of a world point relative to the camera coordinate system at time t_{k-1}, and $(\dot{X}, \dot{Y}, \dot{Z})^T$ represent its associated velocity vector. An object is defined as the vector $\boldsymbol{x}_R = (X, Y, Z, \dot{X}, \dot{Y}, \dot{Z})^T$

The motion of the point can be expressed with a differential equation of the form

$$\dot{\boldsymbol{x}}_R = \boldsymbol{A}_R \boldsymbol{x}_R \tag{4.8}$$

where

$$\boldsymbol{A}_R = \left[\begin{array}{cc} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \end{array} \right] \tag{4.9}$$

In order to be able to express the object state at time t_k in a camera coordinate system, a rotation and translation to the new camera position must be performed. This is achieved by applying the following transformation

$$\boldsymbol{x} = \boldsymbol{R}\boldsymbol{x}_R + \boldsymbol{t} \tag{4.10}$$

where

$$\boldsymbol{R} = \left[\begin{array}{cc} \boldsymbol{R}_C^T & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{R}_C^T \end{array} \right] \quad \text{and} \quad \boldsymbol{t} = \left[\begin{array}{c} -\boldsymbol{R}_C^T \boldsymbol{t}_C \\ \boldsymbol{0}_{3\times1} \end{array} \right] \tag{4.11}$$

$\boldsymbol{t}_C$ and $\boldsymbol{R}_C$ correspond to the camera translation and rotation from Equations 4.6 and 4.7 respectively.

4.3.2.3 Continuous System Model

In order to obtain a differential equation of the form of Equation 4.1, Equation 4.10 is differentiated for the time component obtaining:

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \dot{\boldsymbol{R}}\boldsymbol{x}_R + \boldsymbol{R}\dot{\boldsymbol{x}}_R + \dot{\boldsymbol{t}} \\
&= \dot{\boldsymbol{R}}\boldsymbol{x}_R + \boldsymbol{R}\boldsymbol{A}_R\boldsymbol{x}_R + \dot{\boldsymbol{t}} \\
&= (\dot{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{A}_R)\boldsymbol{x}_R + \dot{\boldsymbol{t}} \\
&= (\dot{\boldsymbol{R}} + \boldsymbol{R}\boldsymbol{A}_R)\boldsymbol{R}^T(\boldsymbol{x} - \boldsymbol{t}) + \dot{\boldsymbol{t}}
\end{aligned}$$

$$\dot{\boldsymbol{x}} = (\dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{A}_R)\boldsymbol{x} + (\dot{\boldsymbol{t}} - (\dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{A}_R)\boldsymbol{t})$$

which leads to

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B} + \boldsymbol{\rho} \tag{4.12}$$

with

$$\boldsymbol{A} = \dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{A}_R \tag{4.13}$$

and

$$\boldsymbol{B} = \dot{\boldsymbol{t}} - (\dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{A}_R)\boldsymbol{t}. \tag{4.14}$$

where the noise term $\boldsymbol{\rho}$ was added in order to model the uncertainty of the system.

Inserting Equations 4.11 into 4.13 and 4.14 results in

$$\boldsymbol{A} = \begin{bmatrix} \dot{\boldsymbol{R}}_C^T\boldsymbol{R}_C & \boldsymbol{I}_{3\times3} \\ 0_{3\times3} & \dot{\boldsymbol{R}}_C^T\boldsymbol{R}_C \end{bmatrix} \tag{4.15}$$

and

$$\boldsymbol{B} = \begin{bmatrix} -\boldsymbol{R}_C^T\dot{\boldsymbol{t}}_C \\ 0_{3\times1} \end{bmatrix} \tag{4.16}$$

Equations 4.12 to 4.14 represent the continuous system model. In order to apply Kalman filters, the system must be first discretized. This is done in the next section.

4.3.3 Discrete System Model

In this section, the continuous model derived in the previous section is discretized. This means that discrete versions of the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{Q}$ must be obtained. For details about the discretization procedure see [Föl94] and [BS94].

4.3.3.1 Transition Matrix A

The discrete version of the matrix $\boldsymbol{A}$ from Equation 4.13 is obtained applying the exponential series expansion [Föl94]

$$e^{\boldsymbol{A}\Delta\mathrm{t}} = \phi(\Delta\mathrm{t}, 0) = \sum_{i=0}^{\infty} \frac{\boldsymbol{A}^i\Delta\mathrm{t}^i}{i!} = \boldsymbol{I} + \boldsymbol{A}\Delta\mathrm{t} + \frac{\boldsymbol{A}^2\Delta\mathrm{t}^2}{2!} + \frac{\boldsymbol{A}^3\Delta\mathrm{t}^3}{3!} + \cdots \tag{4.17}$$

where $\Delta \mathrm{t} = \mathrm{t}_k - \mathrm{t}_{k-1}$. Applying the series expansion to Equation 4.13 results in

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{R}_k & \Delta \mathrm{t}_k \boldsymbol{R}_k \\ \boldsymbol{0}_{\boldsymbol{3}\times\boldsymbol{3}} & \boldsymbol{R}_k \end{bmatrix} \tag{4.18}$$

where $\boldsymbol{R}_k$ corresponds to the inverse rotation matrix of the camera, i.e. $\boldsymbol{R}_k = \boldsymbol{R}_C^T$.

4.3.3.2 System Input Matrix B

The discrete version of the $\boldsymbol{B}$ matrix is obtained by solving (see e.g. [Sim06], page 27)

$$\boldsymbol{B}_k = e^{\boldsymbol{A}\Delta \mathrm{t}} \int_0^{\Delta \mathrm{t}} e^{-\boldsymbol{A}\alpha} d\alpha \boldsymbol{B}(\Delta \mathrm{t}) \tag{4.19}$$

Substituting Equation 4.18 and Equation 4.16 into 4.19 and solving the integral results in

$$\boldsymbol{B}_k = \begin{bmatrix} \boldsymbol{t}_k \\ \boldsymbol{0}_{3\times 1} \end{bmatrix} \tag{4.20}$$

where $\boldsymbol{t}_k$ is the inverse translation of the camera between times t_{k-1} and t_k, i.e. $\boldsymbol{t}_k = -\boldsymbol{R}_C^T \boldsymbol{t}_C$

4.3.3.3 System Covariance Matrix Q

The driving noise term $\boldsymbol{\rho}$ of Equation 4.12 is assumed to have a known covariance matrix $\boldsymbol{\mathcal{Q}}$. The discrete covariance matrix $\boldsymbol{\mathcal{Q}}_k$ is obtained solving the following integral [BS94]

$$\boldsymbol{\mathcal{Q}}_k = \int_0^{\Delta \mathrm{t}_k} \boldsymbol{A}_k \boldsymbol{\mathcal{Q}} \boldsymbol{A}_k^T dt \tag{4.21}$$

Let us define separate covariances for position and velocity

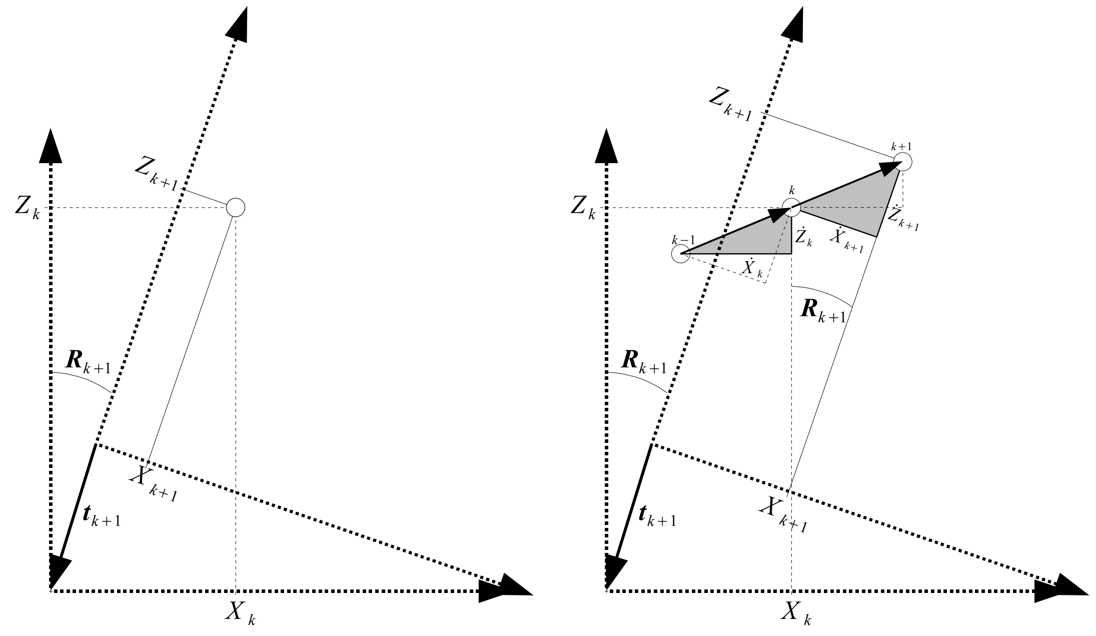
$$\boldsymbol{\mathcal{Q}} = \begin{bmatrix} \boldsymbol{\mathcal{Q}}_{\boldsymbol{P}} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{\mathcal{Q}}_{\boldsymbol{V}} \end{bmatrix} \tag{4.22}$$

with

$$\boldsymbol{\mathcal{Q}}_{\boldsymbol{P}} = \begin{bmatrix} \sigma_P^2 & 0 & 0 \\ 0 & \sigma_P^2 & 0 \\ 0 & 0 & \sigma_P^2 \end{bmatrix} \quad \boldsymbol{\mathcal{Q}}_{\boldsymbol{V}} = \begin{bmatrix} \sigma_V^2 & 0 & 0 \\ 0 & \sigma_V^2 & 0 \\ 0 & 0 & \sigma_V^2 \end{bmatrix} \tag{4.23}$$

Solving 4.21 results in

$$\boldsymbol{\mathcal{Q}}_k = \begin{bmatrix} \Delta \mathrm{t}_k \boldsymbol{\mathcal{Q}}_{\boldsymbol{P}} + \frac{1}{3}\Delta \mathrm{t}_k^3 \boldsymbol{\mathcal{Q}}_{\boldsymbol{V}} & \frac{1}{2}\Delta \mathrm{t}_k^2 \boldsymbol{\mathcal{Q}}_{\boldsymbol{V}} \\ \frac{1}{2}\Delta \mathrm{t}_k^2 \boldsymbol{\mathcal{Q}}_{\boldsymbol{V}} & \Delta \mathrm{t}_k \boldsymbol{\mathcal{Q}}_{\boldsymbol{V}} \end{bmatrix} \tag{4.24}$$



(a) Static Point. (b) Moving Point.

Figure 4.1: Motion of a point observed from a camera coordinate system. The left figure shows an example of the motion of a static point observed from a moving camera: the change in the observed position of the point is only a function of the camera motion. In the right figure the point moves with constant velocity. The new position of the point is not only a function of camera motion but also a function of the point motion. The observed velocity of the point rotates inversely with the camera rotation.

4.3.3.4 Summary of the Discrete System Model

Let $\boldsymbol{p}'_{k-1} = (X, Y, Z)^T$ represent the coordinate vector of a world point observed by the system at time t_{k-1} and $\boldsymbol{v}'_{k-1} = (\dot{X}, \dot{Y}, \dot{Z})^T$ represent its associated velocity vector. The camera platform moves in its environment with a given instantaneous translational and angular velocity, changing its relative position to the point. After a time $\Delta\mathrm{t}_k$ the new position of the point from the camera point of view is given by

$$\boldsymbol{p}'_k = \boldsymbol{R}_k \boldsymbol{p}'_{k-1} + \boldsymbol{t}_k + \Delta\mathrm{t}_k \boldsymbol{R}_k \boldsymbol{v}'_{k-1} \tag{4.25}$$

where $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ are the inverse rotation matrix and inverse translation vector of the camera, i.e. the motion of the static scene with respect to the camera. The velocity vector $\boldsymbol{v}'_k$, in the vehicle coordinate system, changes its direction according to:

$$\boldsymbol{v}'_k = \boldsymbol{R}_k \boldsymbol{v}'_{k-1} \tag{4.26}$$

Figure 4.1 shows examples for static and moving points. Combining position and velocity in the state vector

$$\boldsymbol{x_k} = (\boldsymbol{p}'_k, \boldsymbol{v}'_k)^T \tag{4.27}$$

leads to the discrete linear system model equation:

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k + \boldsymbol{\rho}_k \tag{4.28}$$

with the state transition matrix

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{R}_k & \Delta\mathrm{t}_k \boldsymbol{R}_k \\ \boldsymbol{0_{3\times3}} & \boldsymbol{R}_k \end{bmatrix} \tag{4.29}$$

and input matrix

$$\boldsymbol{B}_k = \begin{bmatrix} \boldsymbol{t}_k \\ \boldsymbol{0_{3\times1}} \end{bmatrix}$$

The noise term $\boldsymbol{\rho}$ is assumed to be Gaussian white noise with covariance matrix $\boldsymbol{\mathcal{Q}}$. The discrete covariance matrix is

$$\boldsymbol{\mathcal{Q}}_k = \begin{bmatrix} \Delta\mathrm{t}_k \boldsymbol{\mathcal{Q}_P} + \frac{1}{3}\Delta\mathrm{t}_k^3 \boldsymbol{\mathcal{Q}_V} & \frac{1}{2}\Delta\mathrm{t}_k^2 \boldsymbol{\mathcal{Q}_V} \\ \frac{1}{2}\Delta\mathrm{t}_k^2 \boldsymbol{\mathcal{Q}_V} & \Delta\mathrm{t}_k \boldsymbol{\mathcal{Q}_V} \end{bmatrix} \tag{4.30}$$

4.3.4 Measurement Model

This section derives the equations corresponding to the measurement of the system. As stated before in Equation 4.2, the measurement $\boldsymbol{z}$ must be related to the state of the system through a matrix $\boldsymbol{H}_k$, which will be obtained in this section.

The measurement model captures the information delivered by the stereo and optical flow systems. The non-linear measurement equation $\boldsymbol{h}$ for a point $\boldsymbol{x}_k = (X, Y, Z, \dot{X}, \dot{Y}, \dot{Z})^T$, given in the camera coordinate system is

$$\boldsymbol{z} = \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \boldsymbol{h}(\boldsymbol{x}_k) = \begin{bmatrix} u_0 \\ v_0 \\ 0 \end{bmatrix} + \frac{1}{Z} \begin{bmatrix} X f_u \\ -Y f_v \\ B f_u \end{bmatrix} + \boldsymbol{\nu} \tag{4.31}$$

where (u, v) corresponds to the new image position of the point (obtained with the optical flow) and d is the disparity computed with stereo. The scalars f_u, f_v, u_0 and v_0 are parameters as introduced in Section 2.2.2.5. The noise term $\boldsymbol{\nu}$ is assumed to be Gaussian white noise with covariance matrix

$$\boldsymbol{\mathcal{R}} = \begin{bmatrix} \sigma_u^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_d^2 \end{bmatrix}. \tag{4.32}$$

Figure 4.3.4 shows an example of the successive measurements.

The non-linearity imposed by the depth component in Equation 4.31 requires the linearization of $\boldsymbol{h}(\boldsymbol{x_k})$. This is achieved by computing the Jacobian matrix of partial

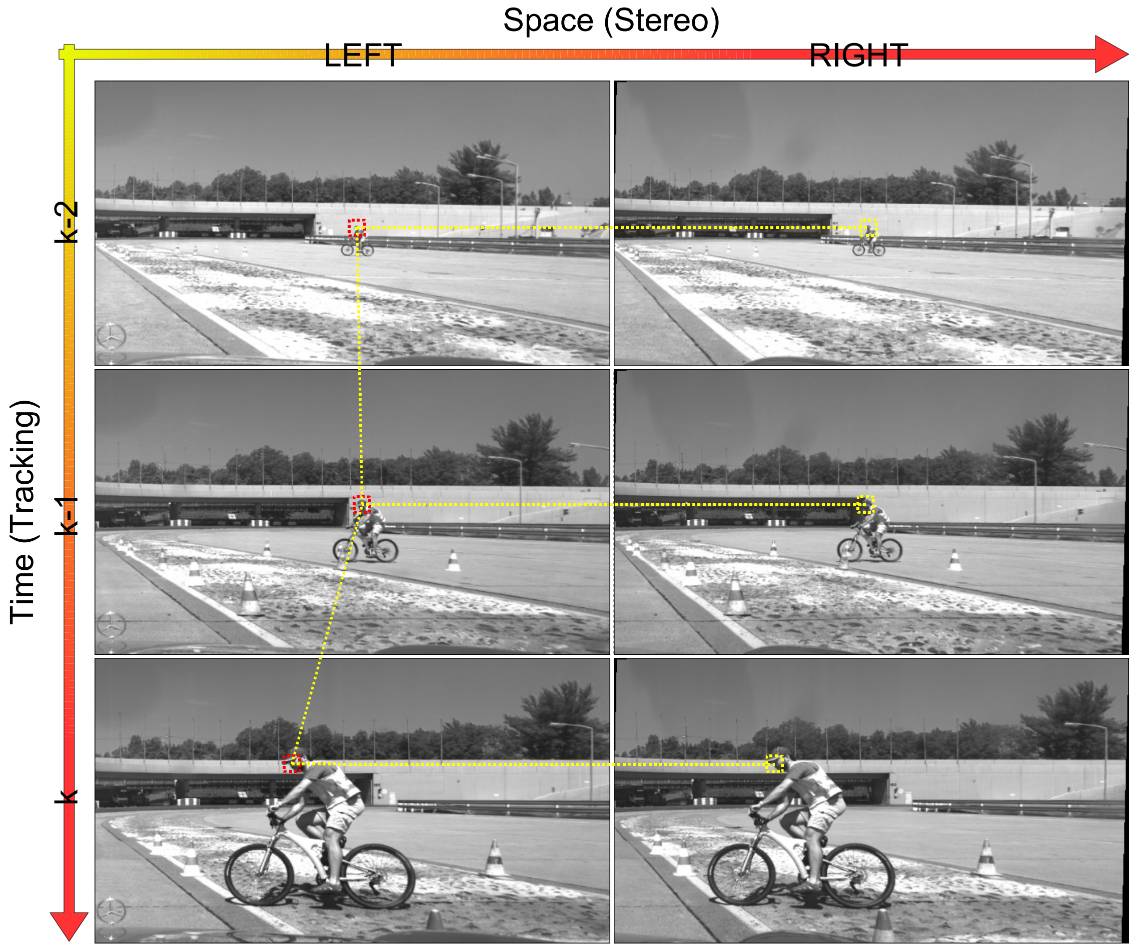


Figure 4.2: Measurement in space and time. The image position (tracking) and disparity (stereo) of a world point is measured for every new image.

derivatives of h with respect to $\boldsymbol{x}$, i.e. :

$$\boldsymbol{H}_{[i,j]} = \frac{\partial \boldsymbol{h}_{[i]}}{\partial \boldsymbol{x}_{[j]}}(\boldsymbol{x}_k, 0) \tag{4.33}$$

The upper-left 3×3 sub-matrix of $\boldsymbol{H}$ results in

$$\boldsymbol{H}_{\boldsymbol{p},\boldsymbol{k}} = \begin{bmatrix} \frac{f_u}{Z} & 0 & -f_u\frac{X}{Z^2} \\ 0 & -\frac{f_v}{Z} & f_v\frac{Y}{Z^2} \\ 0 & 0 & -f_u\frac{B}{Z^2} \end{bmatrix} \tag{4.34}$$

The linearized $\boldsymbol{H}_k$ matrix is

$$\boldsymbol{H}_k = \left[\ \boldsymbol{H}_{\boldsymbol{p},\boldsymbol{k}} \quad \boldsymbol{0}_{\boldsymbol{3}\times\boldsymbol{3}}\ \right]. \tag{4.35}$$

4.3.5 The Extended Kalman Filter Equations

At a given time, the Kalman filter estimates the state of the system projecting the current state and error covariance estimate to obtain *a priori* estimate for the next time step. At another time the Kalman filter receives input in the form of measurements, incorporating them in the *a priori* estimate leading to a *posterior* estimate. The a *a priori* estimate equations are normally called "prediction" while the incorporation of measurements in order to obtain an improved *a posteriori* estimate is called "correction". In this sense the Kalman filter is a predictor-corrector algorithm. In the following a *super minus* over the variable such as $\hat{\boldsymbol{x}}_k^-$ will indicate *a priori* estimate (i.e. prediction), while the lack of such as $\boldsymbol{P}_k$ it will indicate that the variable is a *a posteriori* estimate (i.e. correction).

The state and error covariance prediction equations are:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}_k \tag{4.36}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A^T}_k + \boldsymbol{\mathcal{Q}_k} \tag{4.37}$$

The above equations project the state $\hat{\boldsymbol{x}}$ and its covariance estimate $\boldsymbol{P}$ from time $k-1$ to time k making a prediction. The matrices $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are from Equations 4.18 and 4.20 respectively and $\boldsymbol{\mathcal{Q}}_k$ is from Equation 4.30.

The state and error covariance correction equations are:

- Measurement.

$$\boldsymbol{v}_k = \boldsymbol{z}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-) \tag{4.38}$$

- Computation of Kalman gain:

$$\boldsymbol{F}_k = \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_{\boldsymbol{k}}^{\boldsymbol{T}} + \boldsymbol{\mathcal{R}} \tag{4.39}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_{\boldsymbol{k}}^{\boldsymbol{T}} \boldsymbol{F}_{\boldsymbol{k}}^{-1} \tag{4.40}$$

- Update estimate with measurement:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \boldsymbol{v}_k \tag{4.41}$$

- Update of the error covariance:

$$\boldsymbol{P}_k = (\boldsymbol{I}_{6\times 6} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^- \tag{4.42}$$

where $\boldsymbol{\mathcal{R}}$ is from Equation 4.32, $\boldsymbol{h}(\hat{\boldsymbol{x}}_k^-)$ is described by Equation 4.31 and $\boldsymbol{H}_k$ is from Equation 4.35. The update equations incorporate the measurements 4.38 to the a priori estimate through 4.41. The *a posteriori* error covariance estimate is obtained incorporating the measurement covariance matrix to the predicted error covariance estimate in 4.42. In both cases the previous computation of the matrix $\boldsymbol{K}_k$ in Equations 4.39 and 4.40 is required. $\boldsymbol{K}_k$ is usually called Kalman gain.

The chosen implementation of the Kalman filter uses the UD covariance factorization according to the Bierman algorithm, which do not involve square roots in the implementation and provides a numerically stable and efficient method for factorization. Details can be found in [Bie77] (pages 101, 125, 132).

4.4 Initialization of the Filter and the Cramér-Rao Lower Bound

This section addresses the problem of the initialization of the Kalman filter when no previous information is available. After that, the Cramér-Rao Lower Bound for the proposed KF is derived. In Section 4.5, this bound is used to show the relative performance of the proposed filter.

4.4.1 Initialization of the Filter

The initialization of the model requires an initial estimation of the state vector (3D position and 3D velocity of the world point) as well as the initial system error covariance matrix. The initial 3D position is obtained with the first measurement $\boldsymbol{z_0} = (u, v, d)^T$ of the feature image position and disparity

$$\boldsymbol{p}'_0 = \boldsymbol{g}(\boldsymbol{z}_0) = \begin{bmatrix} \dfrac{Bu'}{d} \\ \dfrac{Bv's_{vu}}{d} \\ \dfrac{Bf_u}{d} \end{bmatrix}. \tag{4.43}$$

where $u' = u - u_0$, $v' = v_0 - v$ and $s_{vu} = s_v/s_u$ as usual. The covariance matrix corresponding to Equation 4.43 is obtained computing the Jacobian of $\boldsymbol{g}(\boldsymbol{z_0})$

$$\boldsymbol{G}_{[i,j]} = \frac{\partial \boldsymbol{g}_{[i]}}{\partial \boldsymbol{z}_{[j]}}(\boldsymbol{z_0}) = \begin{bmatrix} \dfrac{B}{d} & 0 & -\dfrac{Bu'}{d^2} \\ 0 & -\dfrac{Bs_{vu}}{d} & -\dfrac{Bv's_{vu}}{d^2} \\ 0 & 0 & -\dfrac{Bf_u}{d^2} \end{bmatrix}. \tag{4.44}$$

The covariance matrix of the measurement z_0 is the matrix $\mathcal{R}$ of Equation 4.32. The initial covariance matrix for point position is then obtained by

$$\boldsymbol{P}_{0,\boldsymbol{p}'} = \boldsymbol{G}\mathcal{R}\boldsymbol{G}^T = \frac{B^2}{d^4}\begin{bmatrix} u'^2\sigma_d^2 + d^2\sigma_u^2 & s_{vu}u'v'\sigma_d^2 & f_u u'\sigma_d^2 \\ s_{vu}u'v'\sigma_d^2 & s_{vu}^2(v'^2\sigma_d^2 + d^2\sigma_v^2) & f_u s_{vu} v'\sigma_d^2 \\ f_u u'\sigma_d^2 & f_u s_{vu} v'\sigma_d^2 & f_u^2\sigma_d^2 \end{bmatrix}. \tag{4.45}$$

The geometrical interpretation of this matrix will be given in the next chapter. For the estimation of velocity at least two measurements are required. Therefore, the initial velocity is set to zero with a very large covariance matrix, i.e.

$$\boldsymbol{P}_{0,\boldsymbol{v}'} = k_\infty \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} = \begin{bmatrix} k_\infty & 0 & 0 \\ 0 & k_\infty & 0 \\ 0 & 0 & k_\infty \end{bmatrix} \tag{4.46}$$

with k_∞ very large. Combining Equations 4.45 and 4.46 results in the initial covariance matrix:

$$\boldsymbol{P}_{\boldsymbol{0}} = \begin{bmatrix} \boldsymbol{P}_{0,\boldsymbol{p}'} & 0 \\ 0 & \boldsymbol{P}_{0,\boldsymbol{v}'} \end{bmatrix}. \tag{4.47}$$

Observe that $\boldsymbol{P}_{\boldsymbol{0}}^{-1}$ has only rank 3 since $\boldsymbol{P}_{0,\boldsymbol{v}'}^{-1} = \boldsymbol{0}_{3\times 3}$. The initial covariance matrix of Equation 4.47 propagates according to Equation 4.37 to:

$$\begin{aligned} \boldsymbol{P}_{\bar{1}} &= \boldsymbol{R}_{\boldsymbol{1}} \begin{bmatrix} \boldsymbol{P}_{0,\boldsymbol{p}'} + \Delta \mathrm{t}_1^2 \boldsymbol{P}_{0,\boldsymbol{v}'} & \Delta \mathrm{t}_1 \boldsymbol{P}_{0,\boldsymbol{v}'} \\ \Delta \mathrm{t}_1 \boldsymbol{P}_{0,\boldsymbol{v}'} & \boldsymbol{P}_{0,\boldsymbol{v}'} \end{bmatrix} \boldsymbol{R}_{\boldsymbol{1}}^{\boldsymbol{T}} & (4.48) \\ &= \boldsymbol{R}_{\boldsymbol{1}} \begin{bmatrix} \boldsymbol{P}_{0,\boldsymbol{p}'} + \Delta \mathrm{t}_1^2 k_\infty \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} & \Delta \mathrm{t}_1 k_\infty \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} \\ \Delta \mathrm{t}_1 k_\infty \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} & k_\infty \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} \end{bmatrix} \boldsymbol{R}_{\boldsymbol{1}}^{\boldsymbol{T}} & (4.49) \\ &\approx k_\infty \begin{bmatrix} \Delta \mathrm{t}_1^2 \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} & \Delta \mathrm{t}_1 \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} \\ \Delta \mathrm{t}_1 \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} & \boldsymbol{I}_{\boldsymbol{3}\times\boldsymbol{3}} \end{bmatrix} & (4.50) \end{aligned}$$

since we assume $k_\infty \rightarrow \infty$. For the initialization we assume $\boldsymbol{\mathcal{Q}}_{\boldsymbol{1}} = \boldsymbol{0}_{6\times 6}$. Equation 4.50 reflects the fact that no accurate prediction of point position is possible with only one measurement and unknown velocity. Nevertheless, $\boldsymbol{P}_{\bar{1}}^{-1}$ has rank 3 and is obtained as

$$\boldsymbol{P}_{\bar{1}}^{-1} = \boldsymbol{R}_1 \begin{bmatrix} \boldsymbol{P}_{0,\boldsymbol{p}'}^{-1} & -\Delta \mathrm{t}_1 \boldsymbol{P}_{0,\boldsymbol{p}'}^{-1} \\ -\Delta \mathrm{t}_1 \boldsymbol{P}_{0,\boldsymbol{p}'}^{-1} & \Delta \mathrm{t}_1^2 \boldsymbol{P}_{0,\boldsymbol{p}'}^{-1} + (kI)^{-1} \end{bmatrix} \boldsymbol{R}_1^T \tag{4.51}$$

An alternative expression for the updated error covariance matrix shown in Equation 4.42 is:

$$\boldsymbol{P}_k = (\boldsymbol{P}_{-}{}^{\boldsymbol{-1}}{}_k + \boldsymbol{H}^{\boldsymbol{T}}{}_k \boldsymbol{\mathcal{R}}^{\boldsymbol{-1}} \boldsymbol{H}_k)^{-1} \tag{4.52}$$

Just after updating the model with a new measurement, the error covariance matrix shows only finite elements

$$\boldsymbol{P}_{\boldsymbol{1}} = \begin{bmatrix} \boldsymbol{S}_1^{-1} & (\Delta \mathrm{t}_1 \boldsymbol{S}_1)^{-1} \\ (\Delta \mathrm{t}_1 \boldsymbol{S}_1)^{-1} & \dfrac{1}{\Delta \mathrm{t}_1^2} (\boldsymbol{S}_1^{-1} + \boldsymbol{R}_1 \boldsymbol{P}_{0,\boldsymbol{p}'} \boldsymbol{R}_1^T)) \end{bmatrix} \tag{4.53}$$

where:

$$\boldsymbol{S}_k = \boldsymbol{H}_{\boldsymbol{p},\boldsymbol{k}}^{\boldsymbol{T}} \boldsymbol{\mathcal{R}}^{\boldsymbol{-1}} \boldsymbol{H}_{\boldsymbol{p},\boldsymbol{k}} \tag{4.54}$$

The initialization of the filter can be accelerated initializing multiple Kalman filters in parallel for the same input data but with different system variances. More details of this multi-filter system can be found in Franke *et al.* [FRBG05].

4.4.2 Comparison with Optimal Unbiased Estimator

There is a limit on how much information about unknown parameters can be extracted from a set of noisy measurements. For unbiased estimators, the limit is given by the Cramér-Rao Lower Bound (CRLB). Formally, if f is an unbiased estimator of some parameter α the Cramér-Rao inequality states that the mean square error of f is lower bounded by the reciprocal of the Fisher information [Ger99], i.e.

$$\sigma^2(f) \geq \frac{1}{\mathcal{I}(\alpha)} \tag{4.55}$$

The Fisher information $\mathcal{I}(\alpha)$ about the parameter α is defined to be the expectation of the second derivative of the log-likelihood. An unbiased estimator that achieved the CRLB is said to be *efficient*. With Kalman filters, the information matrix can recursively be obtained [Lof90] as

$$\boldsymbol{\mathcal{I}}_k = \boldsymbol{A^T}_k \boldsymbol{\mathcal{I}}_{k-1} \boldsymbol{A}_k + \boldsymbol{H^T}_k \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{H}_k . \tag{4.56}$$

From Equation 4.56, it can be observed that the information provided by every new measurement is given by the second summand of the right hand side, i.e.

$$\boldsymbol{H^T}_k \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{S}_k & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \end{bmatrix} \tag{4.57}$$

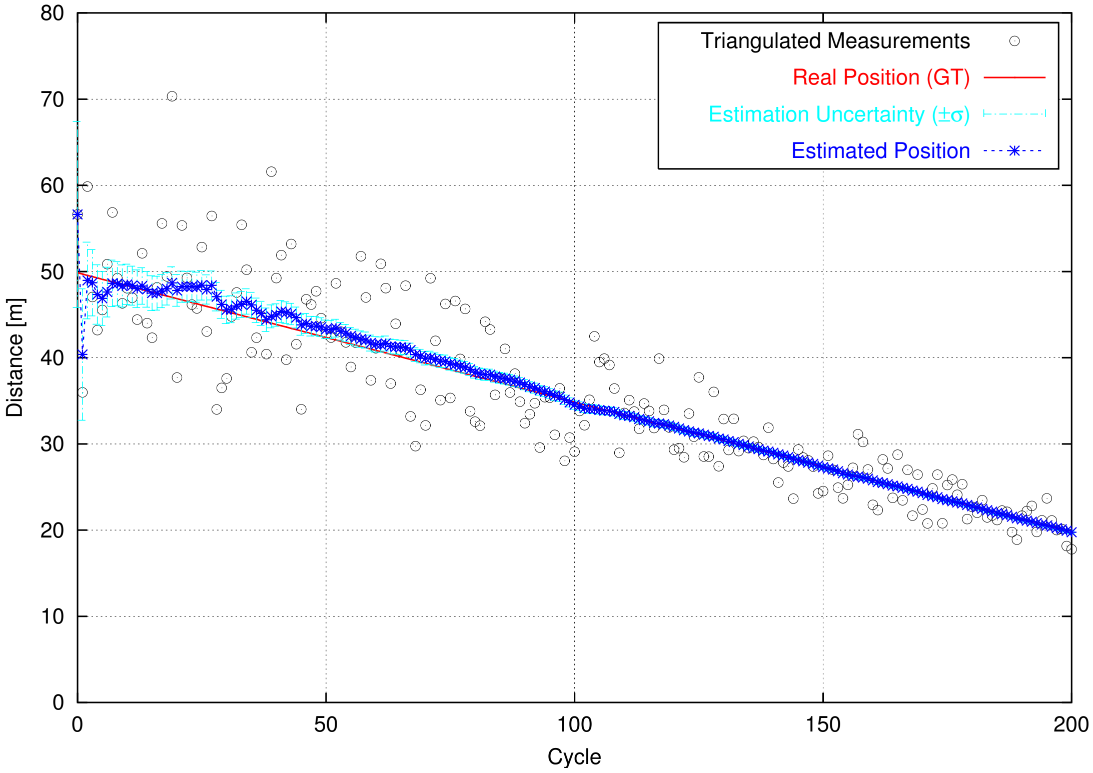
where $\boldsymbol{S}_k$ is from Equation 4.54. This proves that the measurement only provides new information for point position.

4.5 Simulation Results

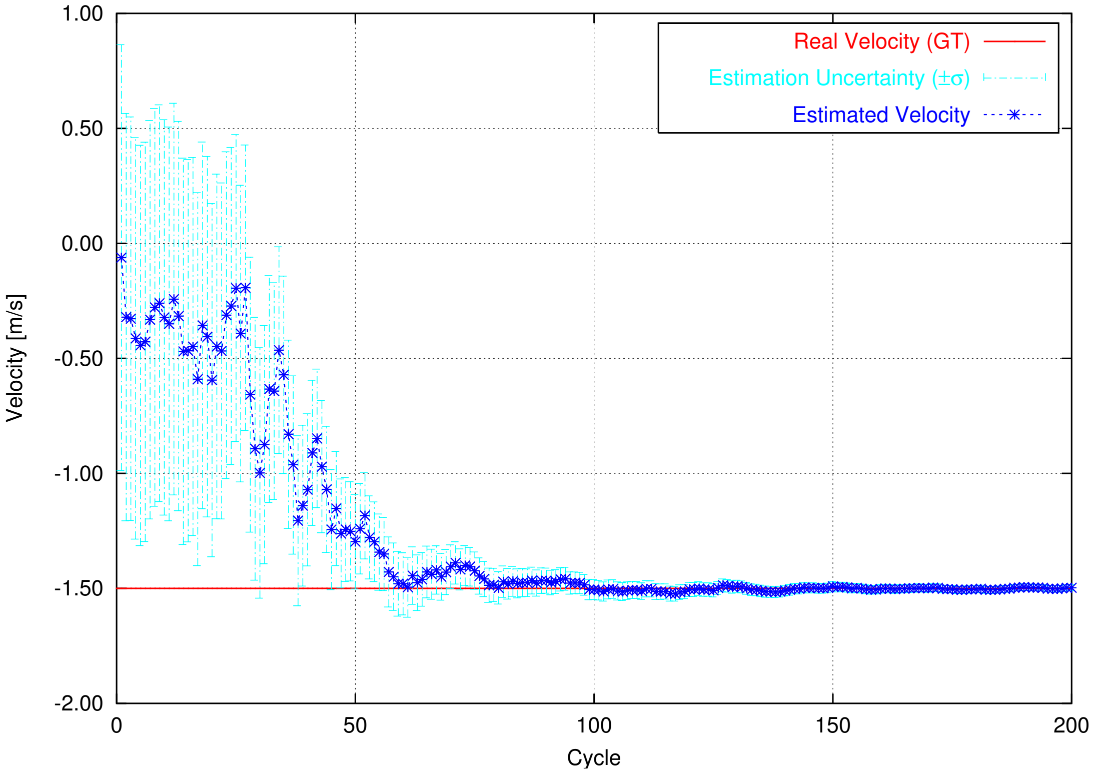
This section evaluates the model under the assumption of known ego-motion, i.e. the rotation matrix $\boldsymbol{R}_k$ and the translation vector $\boldsymbol{t}_k$ of Equations 4.18 and 4.20 are exactly known. Since in reality the ego-motion parameters will contain error, the performance of the filter under noisy ego-motion parameters will be evaluated later in Chapter 7.

A particle is simulated by defining its initial 3D position and corresponding 3D velocity. A standard stereo system is defined with a baseline of $0.35m$, focal length of 830 pixels and VGA image size. At each time instant, the new position of the point (as it moves with constant velocity) is projected into the left and right images of the stereo system. The measurement is simulated by adding Gaussian white noise with a standard deviation of 1 pixel to the image position of the particle. In the simulations only the distance and the speed of the particle in z direction are shown, since z is the component more affected by noise.

In the first simulation a point was located at $50m$ from the camera with a velocity of $-1.5\,m/s$, i.e. moving towards the camera. Figure 4.3(a) shows the Kalman filter estimated position of the point at each time instant. The vertical bars in cyan shows

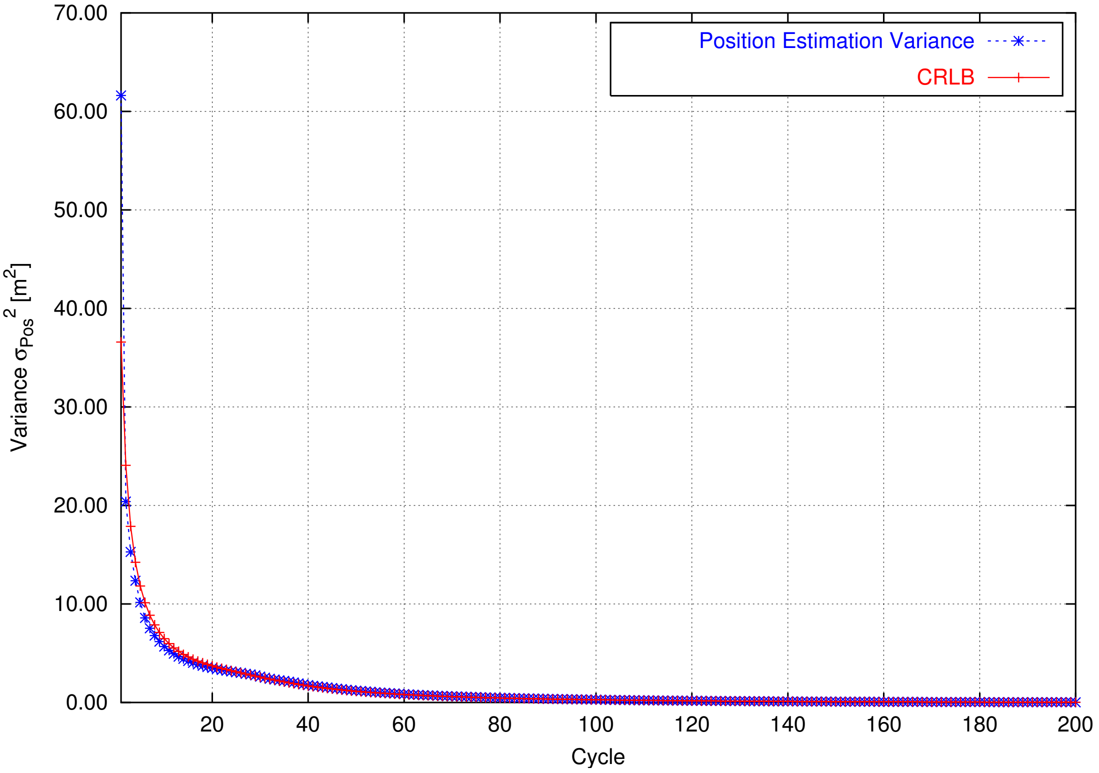


(a) Estimation of the distance.

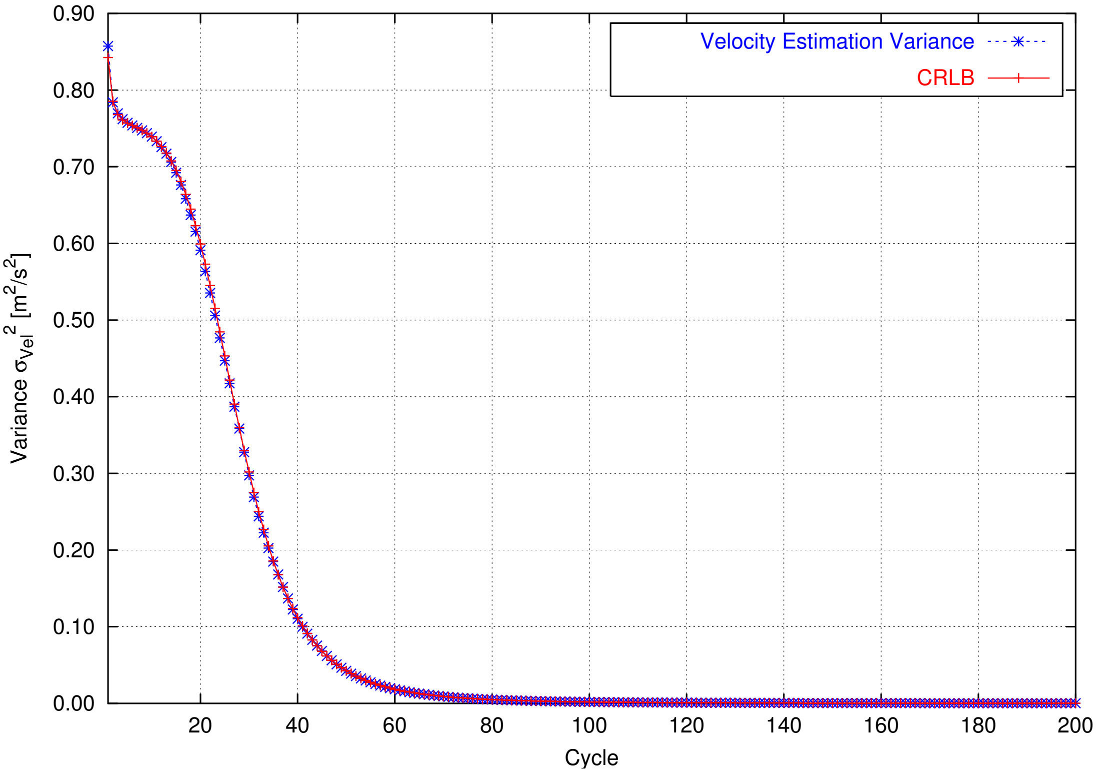


(b) Estimation of the velocity.

Figure 4.3: Estimation of the velocity of a point moving towards the camera.



(a) Estimation of the distance.



(b) Estimation of the velocity.

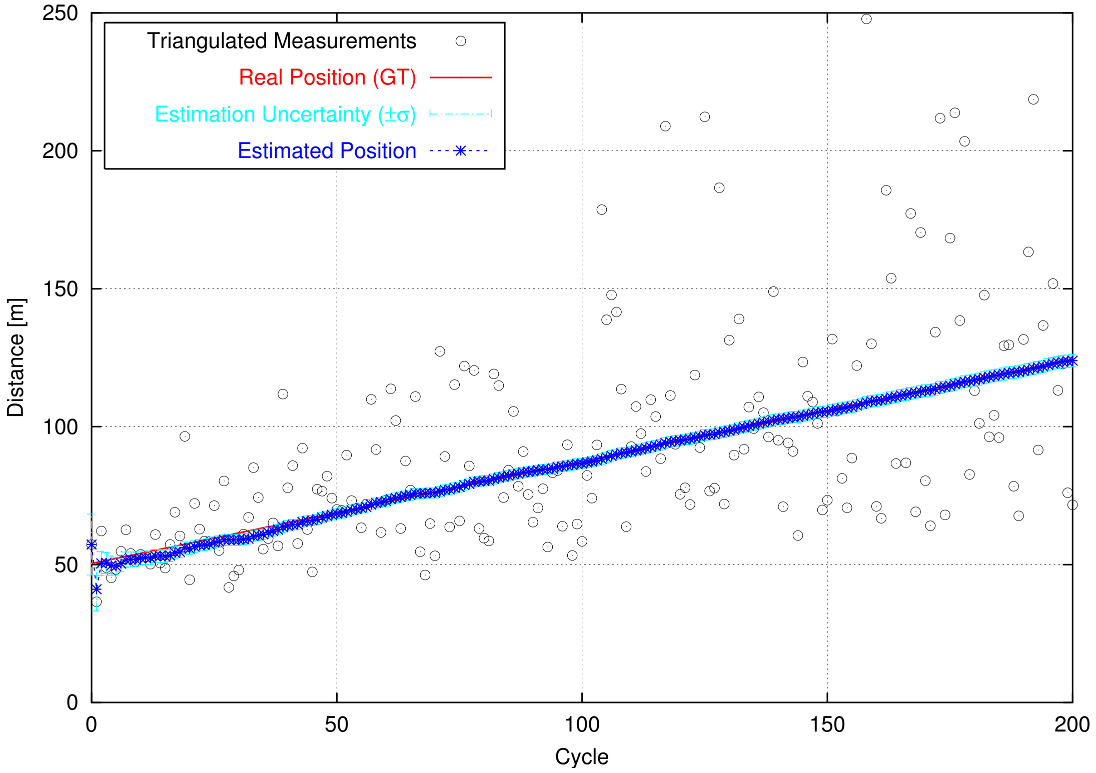
Figure 4.4: Estimation Variance and CRLB for a point moving towards the camera.

the estimation uncertainty, i.e. standard deviation of the estimation. The black circles show the triangulated measurement, i.e. the position of the point without filtering. The red line shows the real value. The position and velocity covariances of equation 4.23 were set to $\sigma_P^2 = 10^{-3}$ and $\sigma_V^2 = 2.5\ 10^{-7}$. Although the triangulated positions are rather scattered around the true position, the Kalman filter estimation converges rather quickly to the real value. Figure 4.3(b) shows the estimated velocity of the particle. Since velocity must be computed by observing the change of the noisy positions of the point and it was initialized to 0, it takes a little longer to converge to the real state. A faster convergence can be achieved if σ_P^2 is assumed to be larger at the expense of reducing the filtering effect giving more weight to the measurements. Figure 4.6 shows the estimation variance for position and speed, as well as the corresponding theoretical Cramér-Rao lower bound, demonstrating an efficient estimation.

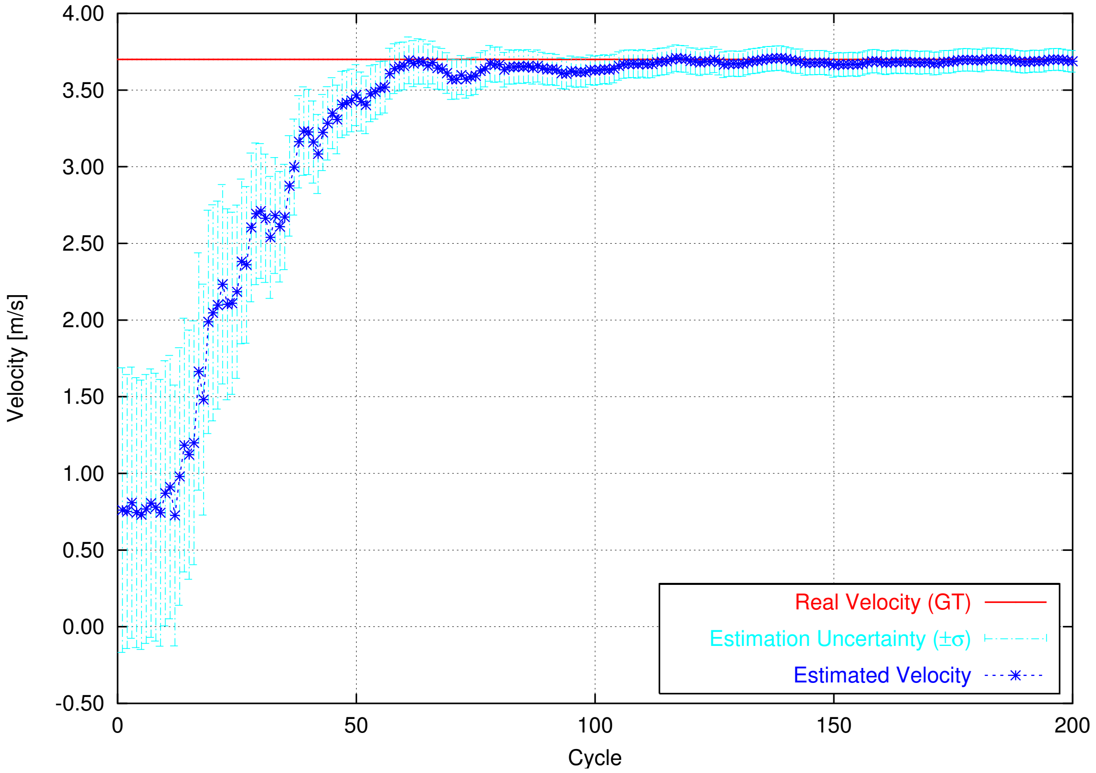
The second simulation scene was generated by locating the particle at $50m$ from the camera but moving away from the camera at a velocity of $3.7\,m/s$. Figure 4.5(a) shows the estimated position, while Figure 4.5(b) shows the estimated velocity of the point. The initial velocity of the point was set to 0. The position and velocity covariances σ_P^2 and σ_V^2 were set as in the previous simulation. Observe that as the point moves away from the camera, the triangulated positions are more and more dispersed, since uncertainty grows with distance. Nevertheless, the filtered position of the point remains stable with no remarkable change of the estimation uncertainty. Figure 4.6(a) shows the estimation variance which increases slow as the point moves away. Since the point is moving away, the estimation of its velocity has a larger uncertainty as in the previous scenario in Figure 4.5(b). Nevertheless, the estimate converges correctly to the right solution. The CRLB is also achieved in this case, as shown in Figure 4.6(b).

In the third simulation a point with constant acceleration was generated. Observe that the filter was designed to estimate only constant velocity, assuming zero acceleration. Nevertheless, Kalman filters can handle deviations from the model if the filters are correctly configured. The process noise covariance matrix $\mathcal{Q}$ of Equation 4.30 allows one to consider uncertainties of the model. Making $\mathcal{Q}$ too large has the effect of trusting less the current state and trusting more the noisy measurements. Letting $\mathcal{Q}$ be too small might prevent the convergence of the filter. In this simulation the the position covariance σ_P^2 was set as in the previous simulation and the velocity covariance of equation 4.23 was increased to $\sigma_V^2 = 0.16$ in order to allow the filter to adapt to changes in the velocity estimation. The system covariance parameters are usually very stable parameters, in the sense that very large variations in the input values generate small differences in the filter output. Therefore, in general, a rough value is enough to ensure a good behavior.

Figure 4.7(a) shows the estimation of the position of a point which moves away from the camera at an initial position of $10m$. A constant acceleration of $-0.2\,m/s^2$ was set, so the point decreases its velocity continuously. The position of the point converges quickly since the point is very close to the camera and therefore its estimation is accurate. The estimation of point position throughout the entire tracking of the point. The velocity estimation shown in Figure 4.7(b) is a little noisier than in previous scenarios and does not fully converge. Nevertheless, the estimated position adapts very well to the constant change of the real velocity, which was actually not

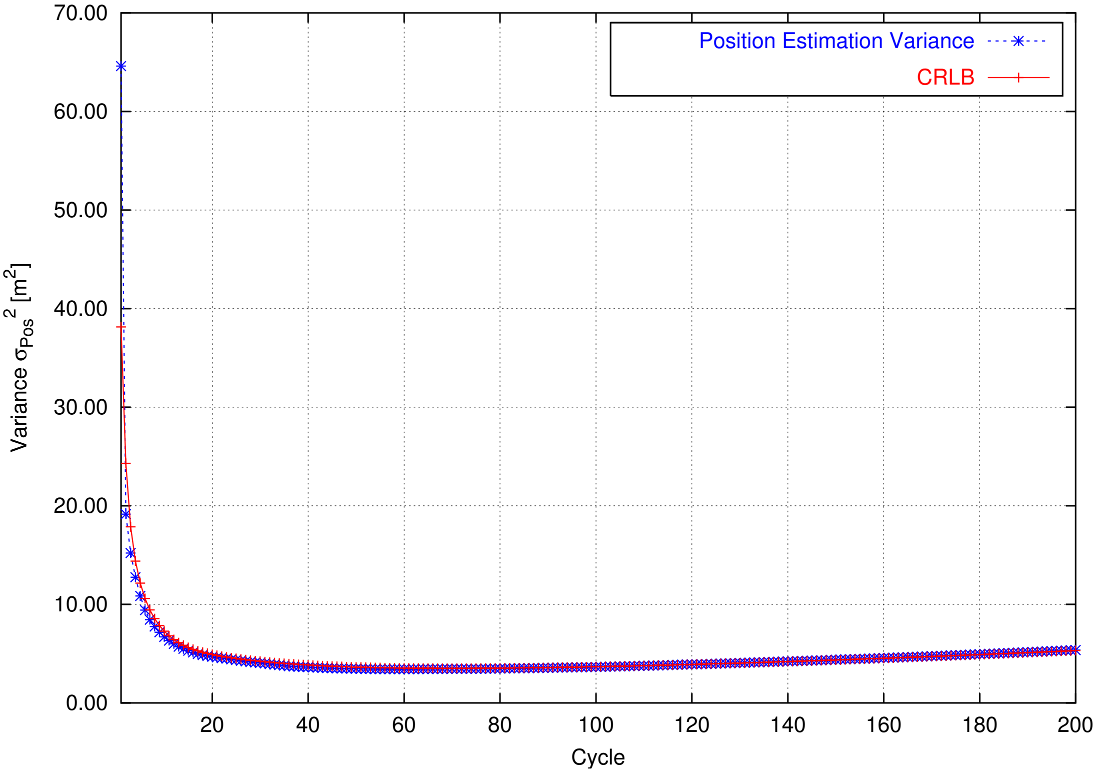


(a) Estimation of the distance.

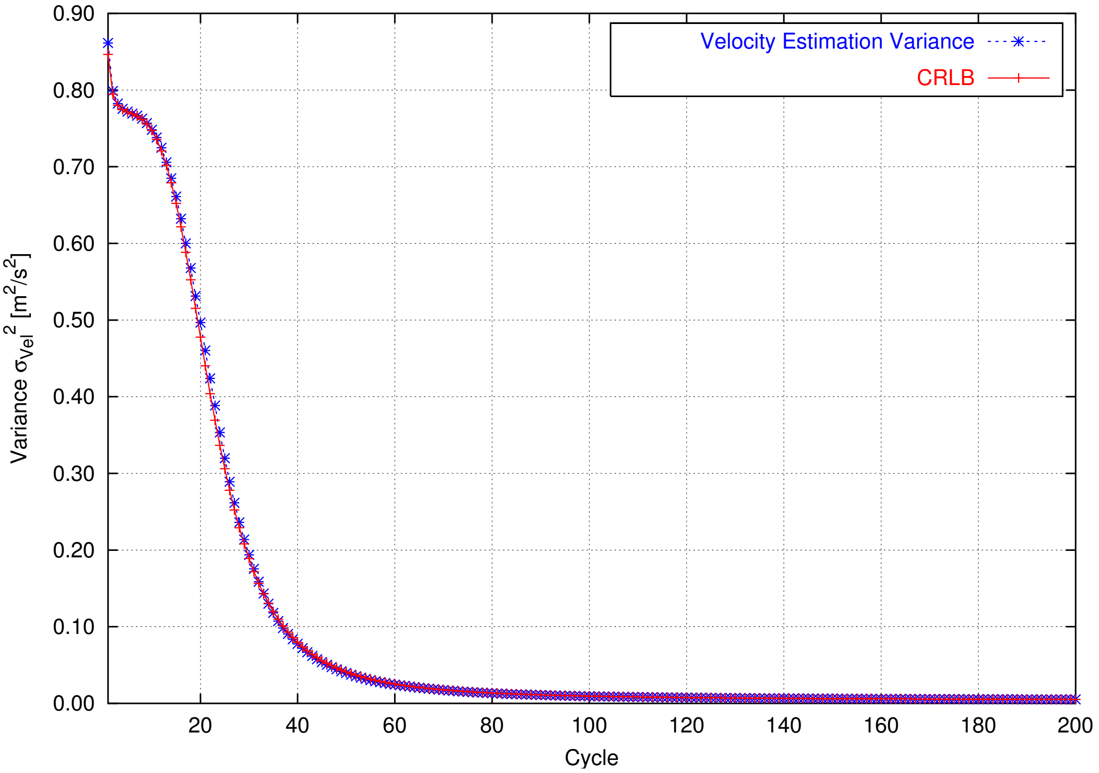


(b) Estimation of the velocity.

Figure 4.5: Estimation of the distance and velocity of a point moving away from the camera.

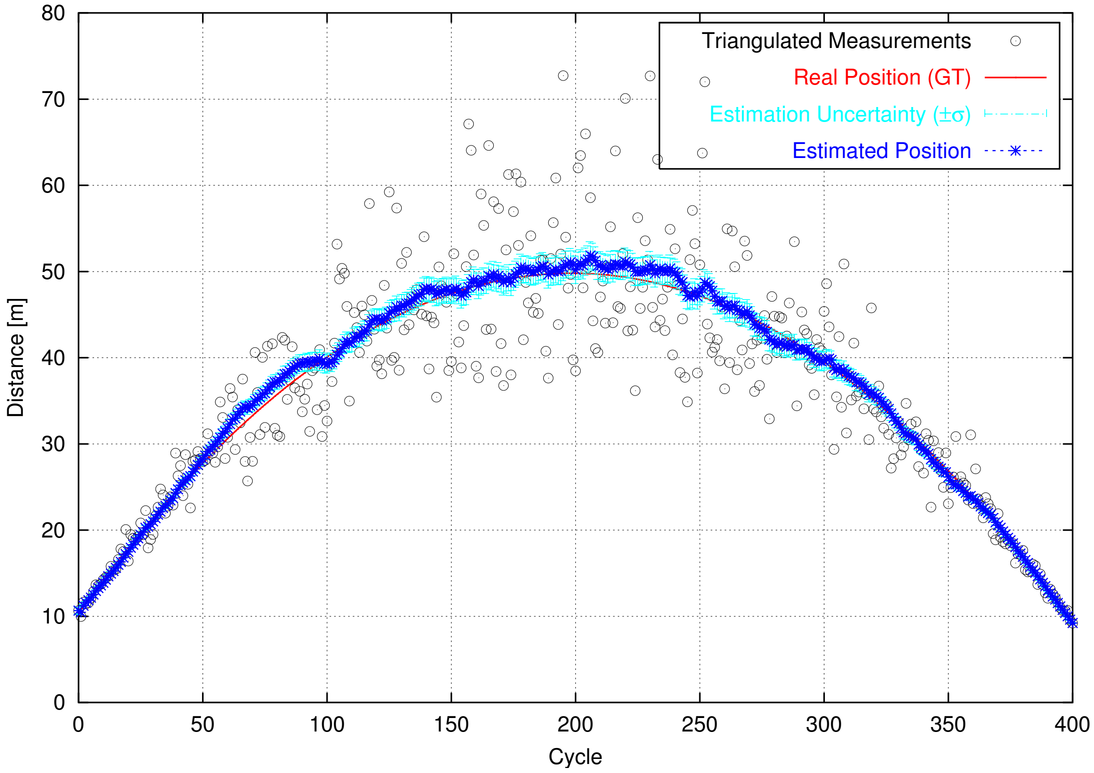


(a) Estimation of the distance.

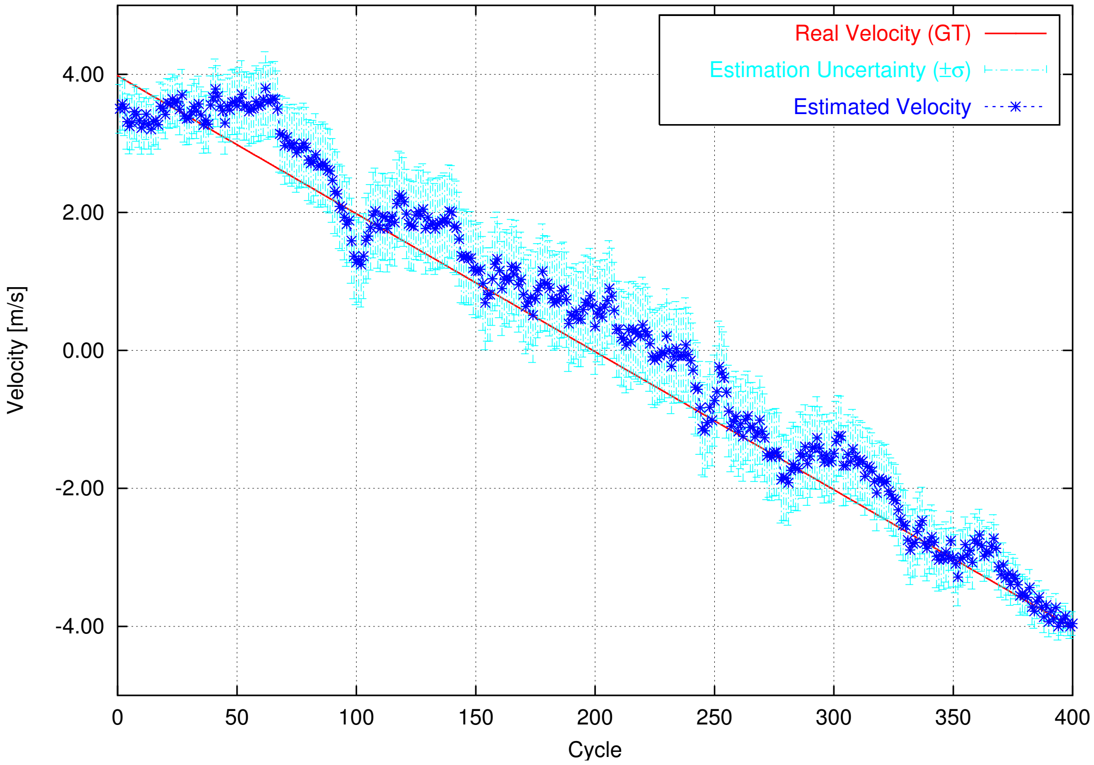


(b) Estimation of the velocity.

Figure 4.6: Estimation Variance and CRLB for a point moving away from the camera.

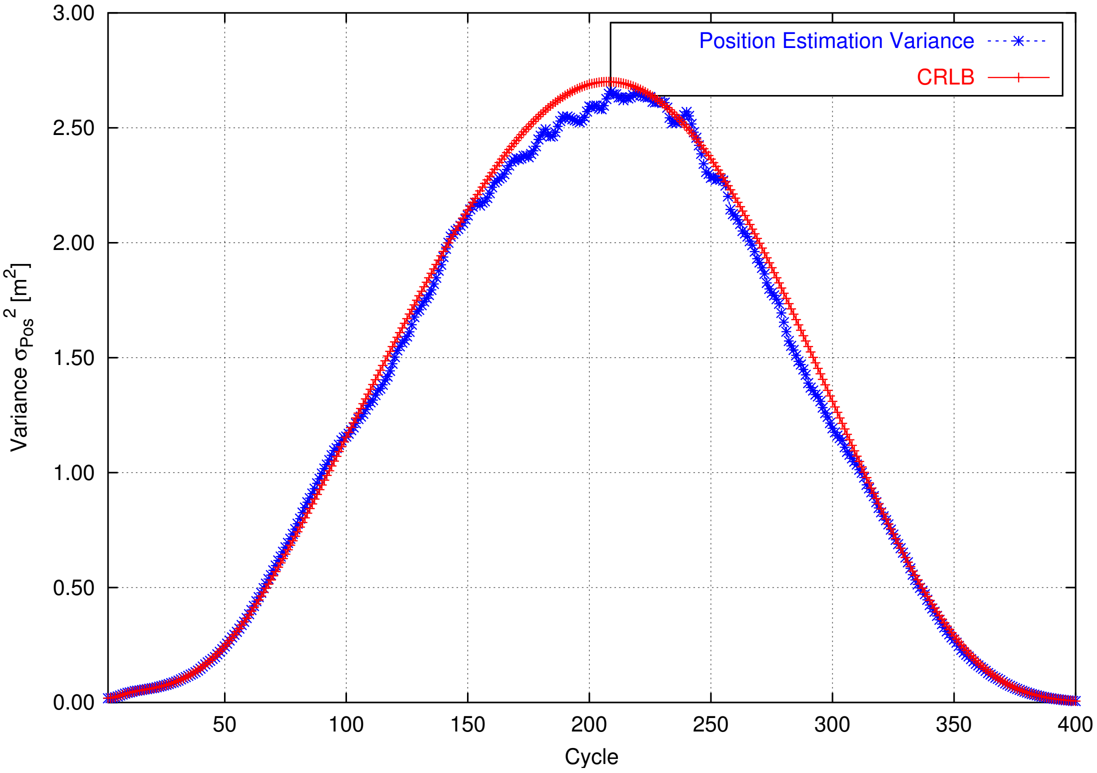


(a) Estimation of the distance.

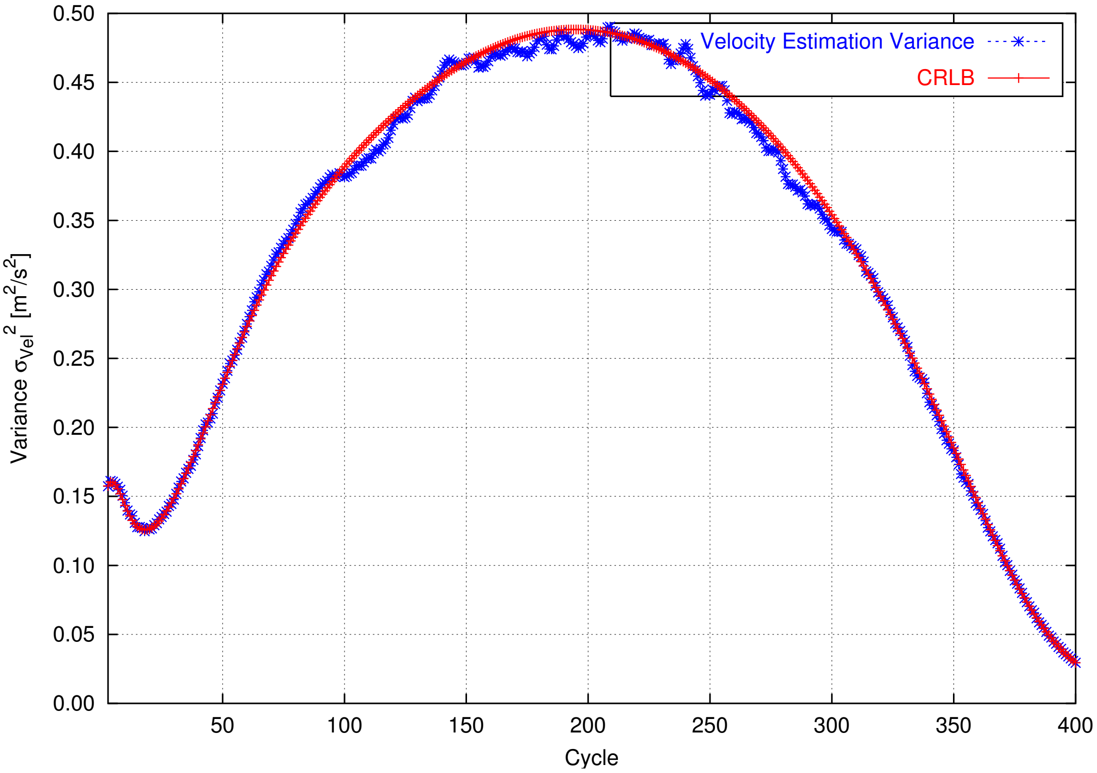


(b) Estimation of the velocity.

Figure 4.7: Estimation of the distance and velocity of a point with constant acceleration.



(a) Estimation of the distance.



(b) Estimation of the velocity.

Figure 4.8: Estimation Variance and CRLB for a point moving with constant acceleration.

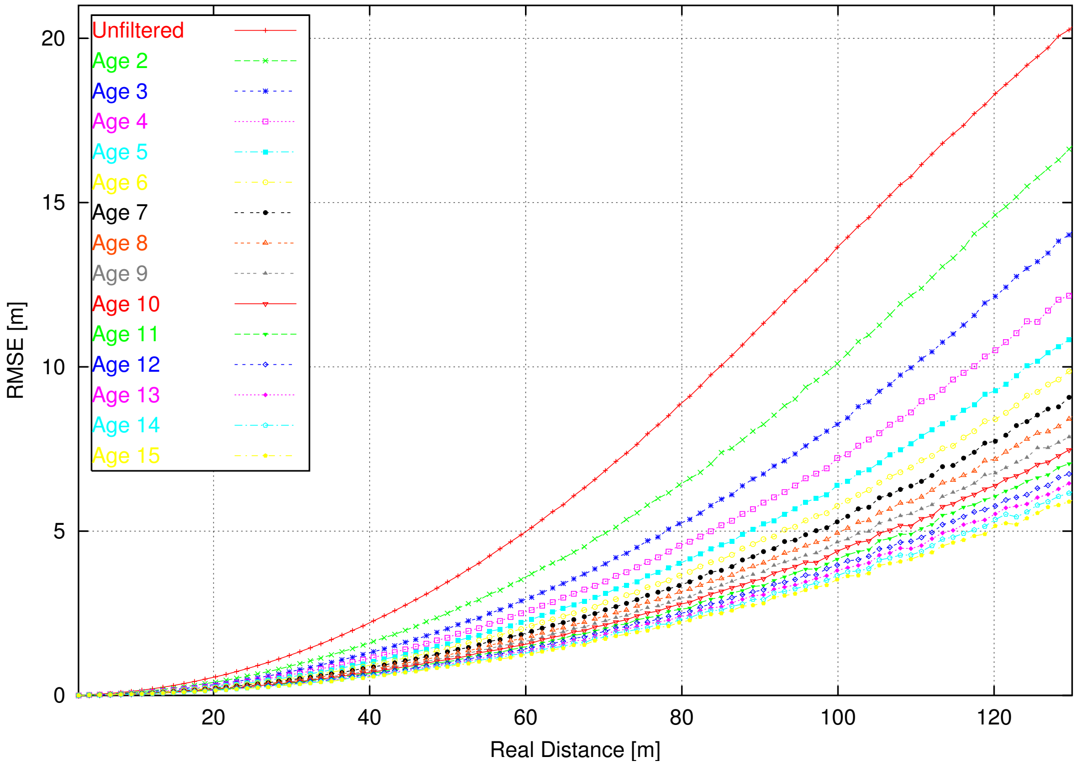


Figure 4.9: Improvement achieved by the integration over time of stereo measurements. Gaussian noise with standard deviation of 0.4 pixels were used in this experiment.

considered in the model.

The estimation variances with corresponding CRLB are shown in Figure 4.8. The dissimilarities between the CRLB curve and the estimation variance curve are larger than in the previous two simulations. This is a consequence of the bias introduced by the non-constant velocity of the particle. The CRLB is a true limit for unbiased estimators, but biased estimators, as the case in this simulation, might achieved smaller variances.

The improvements achieved with Kalman filters are shown with a Monte-Carlo simulation. Figure 4.9 shows curves of expected root mean square distance error as a function of the point distance for different point ages. Every curve shows the results for a specific age. A point of age 5 means that the filter was updated 5 times. Only static points were considered here. The motion of the camera was simulated as constant forward motion. The real benefit of the integration over time for the individual stereo measurements can be seen by comparing each curve with the "Unfiltered" labeled curve, which shows the RMSE for non-filtered stereo measurements. This way, features up to 100 meters away from the camera and with an age of 3 updates have reduced the depth uncertainty by half. Features with an age of 14 updates present only 25% of the standard deviation of unfiltered points.

4.6 Summary

In this chapter, a stochastic model for the recursive estimation of 3D position and 3D velocity of world points was presented. The Kalman filter, as a stochastic model, is a mathematical tool that recursively estimates the state of a dynamic system from data provided by a noisy environment. Under certain assumptions, Kalman filters are optimal estimators, in the sense that they minimize the estimation uncertainty. This is the reason why KFs were, and continue to be, so extensively used for solving computer vision problems. This chapter has reviewed some of the most interesting publications on that subject.

Kalman filters are here used to estimate the position of world points. The simple differentiation over time of the position of the estimated point allows the additional estimation of its velocity. Kalman filters offer a simple way of estimating not only position but also velocity, thus detecting independently moving objects. The motion of a static world point, as being observed from a moving camera coordinate system is modeled by the system model equations. The system model equations incorporate the rigid body motion of the camera with respect to the static environment. The measurements are captured by the measurement model, which fuses stereo and motion data. Without any previous information, the filter requires two initial measurements to be fully initialized. Simulation results have validated the model. Through the comparison with the Cramér-Rao Lower Bound, the method has been shown to be efficient. The reduction of the position uncertainty has also been demonstrated with a Monte-Carlo simulation.

The ego-motion parameters contained in the system model equation were assumed to be provided. When working with simulations, the motion parameters are exactly known. But when working with real images, the ego-motion is unknown and must be estimated. This is performed by computing the absolute orientation between the clouds of static points in the environment. The next chapter addresses the problem of computing the absolute orientation and the problem of modeling the error for the clouds of points.

Chapter 5

The Absolute Orientation Problem

5.1 Introduction

This chapter carries out a complete review on the absolute orientation problem. The focus is given on the closed form solutions for the weighted and total least squares approaches. Section 5.2 reviews the literature on absolute orientation. Sections 5.3.2 to 5.5 are dedicated to the mathematical formulation of the weighted and total least squares approaches.

5.2 Literature Review

The computation of absolute orientation using a set of point correspondences is a problem which was solved almost independently by many authors in different research areas. The same problem was named *Absolute Orientation* in photogrammetry [Hor87], *3D-3D Pose Estimation* in Computer Vision [HJL$^+$87] and even *Orthogonal Procrustes Problem* [SC70] in statistical psychology. The first closed form solutions to the Least Square (LS) problem were proposed in the early 1970's. Schönemann [SC70] presented a method which treats the nine elements of the rotation matrix as unknowns and applies Lagrange multipliers to force the orthogonality. Singular Value Decomposition (SVD) is then applied for the computation of the rotation matrix. Blais [Bla72] and Sansò [San73] also proposed closed form solutions. The method proposed by Sansò finds rotation as unit quaternion, extracting the eigenvector corresponding to the minimum eigenvalue of a symmetrical matrix, the latter constructed directly from the data sets. A decade later, Faugeras and Hebert [FH83], without the knowledge of the work of Sansò, find the same quaternion solution. In the second half of the 80's Horn [Hor87] rediscovers the quaternion technique presenting small variations in the derivation. The optimal rotation is now found as the eigenvector corresponding to the most positive eigenvalue of symmetric 4×4 matrix. Horn *et al.* [HHN88] also published a solution with orthonormal matrices by polar decomposition. In the same year, Arun *et al.* [AHB87] and Haralick *et al.* [HJL$^+$87], in independent investigations, bring to light once again the SVD technique. A fourth solution with dual-quaternions was found by Walker *et al.* [WS91]. The SVD technique was improved by Umeyama [Ume91] by imposing an additional constraint to avoid reflections. These four different techniques were evaluated in accuracy, stability in case of degenerate data forms, and efficiency by Lorusso *et al.* [LEF95]. Kanatani [Kan94] recapitulates the SVD, quaternion and polar decomposition methods in a redefined manner as minimization over proper rotations.

Most solutions to the Total Least Square (TLS) approach to absolute orientation are iterative. Matthies and Shafer [MS87] solves the problem by maximum likelihood estimation, computing rotation and translation separately and minimizing iteratively with Gauss-Newton. Ohta and Kanatani [OK98] use quaternions and the renormalization technique for finding best rotation fit between clouds of points with anisotropic noise. Matei and Meer [MM99] propose a method which uses the same linearization method with quaternions as in the method of Ohta and Kanatani, and extend it to consider a translational term. Bootstrap analysis is also used to estimate the covariance of the data points and a confidence of the motion result. Williams *et al.* [WB99] present another method, computing simultaneously motion and structure with maximum likelihood estimation. Weng *et al.* [WC90], on the other hand, solve the absolute orientation problem in closed form, eliminating a rotation from the covariance matrix and penalizing this way the resulting motion as shown later in this chapter. Nevertheless the proposed method allows the implementation of a very fast algorithm, making it attractive for real-time applications.

5.3 Preliminaries

5.3.1 Introduction to Least Squares

Least squares (LS) is a method for solving overdetermined systems. The least squares method performs a fit of a model to a set of observations. This is achieved by finding the set of parameters of the model for which the sum of the *squared* residual has it *least* value, a residual being the difference between an observation and the output given by the model. The method was first described by Carl Friedrich Gauss around the end of the 18^{th} century.

Let us define a set of m observations

$$b_i = h(\boldsymbol{a}_i, \bar{\boldsymbol{x}}) - \bar{w}_i \qquad i = 1, \cdots, m \tag{5.1}$$

where $h(\boldsymbol{a}_i, \bar{\boldsymbol{x}})$ is a function relating the real unknown vector $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ and independent known vectors $\boldsymbol{a}_i \in \mathbb{R}^q$ with the dependent variable $b_i \in \mathbb{R}$ found by observation. The scalars $\bar{w}_i$ are also unknown and are called *measurement error* or *noise*. The least squares estimator of $\boldsymbol{x}$ is

$$\hat{\boldsymbol{x}} = \arg\min_{x} \left\{ J = \sum_{i=1}^{m} r_i^2 \right\} \qquad r_i = b_i - h(\boldsymbol{a}_i, \boldsymbol{x}). \tag{5.2}$$

where r_i is called residual value and J is the sum of the squared residuals.

The minimum is found where the slope of Equation 5.2 is zero. This leads to a set of n equations of the form

$$0 = -2 \sum_{i=1}^{m} r_i \frac{\partial h(\boldsymbol{a}_i, x_j)}{\partial x_j} \quad j = 1, \cdots, n. \tag{5.3}$$

The n equations in 5.3 are called *gradient equations* and solving them for $\boldsymbol{x}$ depends on the function $h(\boldsymbol{a}_i, \boldsymbol{x})$. If $h(\boldsymbol{a}_i, \boldsymbol{x})$ is linear in $\boldsymbol{x}$, then Equation 5.2 is called *linear* least squares. In particular, if $h(\boldsymbol{a}_i, \boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x}$ where $\boldsymbol{a}_i \in \mathbb{R}^n$ is a vector of constants then Equation 5.2 becomes

$$\hat{\boldsymbol{x}} = \arg\min_{x} \left\{ J = \sum_{i=1}^{m} r_i^2 \right\} \quad r_i = b_i - \boldsymbol{a}_i^T \boldsymbol{x} \tag{5.4}$$

Stacking the observations b_i and independent vectors $\boldsymbol{a}_i$ one over the other

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix} \tag{5.5}$$

and solving Equations 5.3 leads to the closed form solution (assuming $m >= n$)

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{b}. \tag{5.6}$$

in which case the sum of the squared residuals is

$$J = (\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}})^T (\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}) \tag{5.7}$$

If the noise terms have a mean of zero (i.e. $\mathrm{E}[w] = 0$) and are uncorrelated then the least square estimator is the best linear unbiased estimator (BLUE). This result is known as the Gauss-Markov theorem [KK98]. The case of equal variance for all measurements was demonstrated by Carl Friedrich Gauß almost 200 years ago. Aitken [Ait35] extended the Gauss-Markov theorem to the more general case of different non-scalar covariance matrix and correlated errors. In such case the least squares corresponds to the minimizing the Mahalanobis norm of the residual vector

$$J = (\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}})^T \boldsymbol{W}^{-1} (\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}) \tag{5.8}$$

where the matrix $\boldsymbol{W}$ is the second order moment of $\boldsymbol{b}$ known up to some scale factor [OK98].

Observe that the data model 5.1 does not count for errors in the independent variable $\boldsymbol{a}_i$. A more appropriate technique for solving systems in which both the dependent and independent variables are subject to error is the framework of the "Total Least Squares" (TLS) estimation [MM98], also known as errors-in-variables-modeling (EIV) or orthogonal regression. Details about LS and TLS are found in [Müh05].

The next section make a general formulation of the Absolute Orientation Problem. Solutions to the LS and TLS are shown later in this chapter.

5.3.2 The Absolute Orientation Problem

Given the sets $\mathcal{P} = \{\boldsymbol{p}_i\}$ and and $\mathcal{X} = \{\boldsymbol{x}_i\}$ of 3D points where $\boldsymbol{p}_i \leftrightarrow \boldsymbol{x}_i$, i.e. $\boldsymbol{x}_i$ is the transformed version at time t_n of the point $\boldsymbol{p}_i$ at time t_{n-1}, the Absolute Orientation Problem implies the estimation of the rigid-body transformation parameters between the sets $\mathcal{P}$ and $\mathcal{X}$[1]. The sets $\mathcal{P}$ and $\mathcal{X}$ are composed of measurements such that

$$\begin{aligned} \boldsymbol{x}_i &= \bar{\boldsymbol{x}}_i + \bar{\boldsymbol{\delta}}_{x,i} \\ \boldsymbol{p}_i &= \bar{\boldsymbol{p}}_i + \bar{\boldsymbol{\delta}}_{p,i} \end{aligned}$$

i.e. $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are unbiased noisy observations of the real, but unavailable vectors $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{p}}_i$, and $\bar{\boldsymbol{\delta}}_{x,i}$ and $\bar{\boldsymbol{\delta}}_{p,i}$ are independent random vectors with second order moment (i.e. covariance matrices) $\boldsymbol{\Gamma}_{x,i}$ and $\boldsymbol{\Gamma}_{p,i}$ respectively. The Absolute Orientation Problem requires to find the rotation vector $\boldsymbol{\omega} = (\phi, \psi, \theta)^T$ and the translation vector $\boldsymbol{t} = (t_x, t_y, t_z)^T$ such that

$$\bar{\boldsymbol{p}}_i = \boldsymbol{R}(\boldsymbol{\omega})\bar{\boldsymbol{x}}_i + \boldsymbol{t} \tag{5.9}$$

where $\boldsymbol{R}(\boldsymbol{\omega})$ is the rotation matrix corresponding to $\boldsymbol{\omega}$. The direction of the vector $\boldsymbol{\omega}$ defines the axis of rotation and its magnitude the rotation angle. The rotation matrix $\boldsymbol{R}(\boldsymbol{\omega})$ is obtained with Equation 4.7 as seen in the previous chapter. In order to simplify notation, the functional part will be omitted and the rotation matrix will be written as $\boldsymbol{R}$. A statistically optimal solution to Equation 5.9 is found minimizing the Mahalanobis norm of the residual vector ([Müh05] page 75), i.e.

$$J = \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i)^T \boldsymbol{\Gamma}_{x,i}^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i) + \sum_{i=1}^{n} (\boldsymbol{p}_i - \bar{\boldsymbol{p}}_i)^T \boldsymbol{\Gamma}_{p,i}^{-1} (\boldsymbol{p}_i - \bar{\boldsymbol{p}}_i) \tag{5.10}$$

subject to the Equation 5.9. This is a formulation of the Generalized Total Least Squares problem [EBW04]. Under the zero-mean assumptions $\mathrm{E}(\bar{\boldsymbol{\delta}}_{x,i}) = 0$ and $\mathrm{E}(\bar{\boldsymbol{\delta}}_{p,i}) = 0$, Equation 5.10 is the best linear unbiased estimator. Furthermore, if the errors are Gaussian, i.e.

$$\begin{aligned} \bar{\boldsymbol{\delta}}_{x,i} &\sim N(0, \boldsymbol{\Gamma}_{x,i}) \qquad (5.11) \\ \bar{\boldsymbol{\delta}}_{p,i} &\sim N(0, \boldsymbol{\Gamma}_{p,i}) \qquad (5.12) \end{aligned}$$

then Equation 5.10 defines a maximum likelihood estimator [MM98].

With fixed $\boldsymbol{R}$ and $\boldsymbol{t}$, the vectors $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{p}}_i$ that minimize J can be obtained analytically adding Lagrange multipliers, leading to [OK98]

$$J = \sum_{i=1}^{n} (\boldsymbol{p}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t})^T \boldsymbol{W}_{\boldsymbol{i}} (\boldsymbol{p}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}) \tag{5.13}$$

---

[1] Actually, one can generalize the Absolute Orientation Problem in order to consider a scale factor. Nevertheless, only the rigid body transformation parameters are considered here.

where

$$\boldsymbol{W_i} = (\boldsymbol{R}\boldsymbol{\Gamma}_{x,i}\boldsymbol{R}^T + \boldsymbol{\Gamma}_{p,i})^{-1} \tag{5.14}$$

Observe that $\boldsymbol{R}$ belongs to the group of rotations $SO(3)$. The matrix $\boldsymbol{R}$ contains 9 elements, but only 3 degrees of freedom. Finding the rotation which minimizes Equation 5.13 requires to consider the non-linear constraints imposed by $\boldsymbol{R}$.

5.4 Weighted Least Squares Formulation

This section reviews three closed form solution of the absolute orientation problem with Weighted Least Squares (WLS).

If the noise terms of Equations 5.11 and 5.12 are equal in all dimensions, then the covariance matrices are

$$\begin{aligned} \boldsymbol{\Gamma}_{x,i} &= \sigma_{x,i}^2 \boldsymbol{I}_{3\times 3} \qquad (5.15)\\ \boldsymbol{\Gamma}_{p,i} &= \sigma_{p,i}^2 \boldsymbol{I}_{3\times 3} \qquad (5.16) \end{aligned}$$

and the weight matrix of Equation 5.14 becomes

$$\begin{aligned} \boldsymbol{W_i} &= (\boldsymbol{R}\sigma_{x,i}^2 \boldsymbol{I}_{3\times 3}\boldsymbol{R}^T + \sigma_{p,i}^2 \boldsymbol{I}_{3\times 3})^{-1} \\ &= (\sigma_{x,i}^2 \boldsymbol{R}\boldsymbol{R}^T + \sigma_{p,i}^2 \boldsymbol{I}_{3\times 3})^{-1} \\ &= \frac{1}{(\sigma_{x,i}^2 + \sigma_{p,i}^2)} \boldsymbol{I}_{3\times 3} \end{aligned}$$

Observe that the weight matrix does not depend any more on the rotation. The closed form solution for the TLS problem of Weng *et al.* [WC90] relies on this simplification, as shown in the next subsection. The Least Squares formulation results in

$$J = \sum_{i=1}^{n} w_i \|\boldsymbol{p}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|^2 \tag{5.17}$$

with

$$w_i = \frac{1}{(\sigma_{x,i}^2 + \sigma_{p,i}^2)}. \tag{5.18}$$

The first step is to transform both sets so that the translation parameters can be decoupled from the rotation terms. In order to achieve this, both sets are translated with their weighted centroids

$$\begin{aligned} \hat{\boldsymbol{x}}_i &= \boldsymbol{x}_i - \boldsymbol{x_0} \\ \hat{\boldsymbol{p}}_i &= \boldsymbol{p}_i - \boldsymbol{p_0} \end{aligned}$$

where $\boldsymbol{x_0}$ and $\boldsymbol{p_0}$ are the weighted centroids of each set:

$$\boldsymbol{x_0} = \left(\sum_{i=1}^{n} w_i\right)^{-1} \sum_{i=1}^{n} w_i \boldsymbol{x}_i$$
$$\boldsymbol{p_0} = \left(\sum_{i=1}^{n} w_i\right)^{-1} \sum_{i=1}^{n} w_i \boldsymbol{p}_i$$

Observe now that

$$\sum_{i=1}^{n} w_i \hat{\boldsymbol{x}}_i = 0$$
$$\sum_{i=1}^{n} w_i \hat{\boldsymbol{p}}_i = 0$$

The new objective function to minimize is

$$J = \sum_{i=1}^{n} w_i \|\hat{\boldsymbol{p}}_i - \boldsymbol{R}\hat{\boldsymbol{x}}_i - \boldsymbol{t_0}\|^2$$

where

$$\boldsymbol{t_0} = \boldsymbol{t} + \boldsymbol{R}\boldsymbol{x_0} - \boldsymbol{p_0}$$

Expanding the new objective function we obtain

$$J = \sum_{i=1}^{n} w_i \|\hat{\boldsymbol{p}}_i - \boldsymbol{R}\hat{\boldsymbol{x}}_i\|^2 - 2\boldsymbol{t_0} \sum_{i=1}^{n} w_i (\hat{\boldsymbol{p}}_i - \boldsymbol{R}\hat{\boldsymbol{x}}_i) + \|\boldsymbol{t_0}\|^2 \sum_{i=1}^{n} w_i \tag{5.19}$$

The first term of the r.h.s. does not depend on $\boldsymbol{t}$ and the second terms equals 0 since

$$\begin{aligned} \sum_{i=1}^{n} w_i (\hat{\boldsymbol{p}}_i - \boldsymbol{R}\hat{\boldsymbol{x}}_i) &= \underbrace{\sum_{i=1}^{n} w_i \hat{\boldsymbol{p}}_i}_{0} - \underbrace{\sum_{i=1}^{n} w_i \boldsymbol{R}\hat{\boldsymbol{x}}_i}_{\boldsymbol{R}\underbrace{\sum_{i=1}^{n} w_i \hat{\boldsymbol{x}}_i}_{0}} \\ &= 0 \end{aligned} \tag{5.20}$$

Since the last expression of Equation 5.19 cannot be negative, the minimum is obtained when $\boldsymbol{t_0} = 0$ or

$$\boldsymbol{t} = \boldsymbol{p_0} - \boldsymbol{R}\boldsymbol{x_0} \tag{5.21}$$

i.e. translation is obtained as the difference between the rotated centroid of the set $\mathcal{X}$ with the centroid of the set $\mathcal{P}$. Since the second and third terms are zero, the new cost function is

$$J = \sum_{i=1}^{n} w_i \|\hat{\boldsymbol{p}}_i - \boldsymbol{R}\hat{\boldsymbol{x}}_i\|^2. \tag{5.22}$$

Expanding once again the left hand side of Equation 5.22 we obtain

$$J = \sum_{i=1}^{n} w_i \|\hat{\boldsymbol{p}}_i\|^2 - 2\sum_{i=1}^{n} w_i(\hat{\boldsymbol{p}}_i^T \boldsymbol{R}\hat{\boldsymbol{x}}_i) + \sum_{i=1}^{n} w_i(\hat{\boldsymbol{x}}_i^T \boldsymbol{R}^T \boldsymbol{R}\hat{\boldsymbol{x}}_i) \tag{5.23}$$

Rotation is simplified from the last term of the r.h.s. since $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_{3\times 3}$. The optimal $\boldsymbol{R}$ can be found by minimizing the expression in the middle of Equation 5.23 (or maximizing its negative). Applying the known trace property $tr(\boldsymbol{a}^T\boldsymbol{b}) = tr(\boldsymbol{a}\boldsymbol{b}^T)$, the expression in the middle of the r.h.s. is also expressed by

$$\sum_{i=1}^{n} w_i((\hat{\boldsymbol{p}}_i^T \boldsymbol{R})\hat{\boldsymbol{x}}_i) = tr\left(\boldsymbol{R}^T \sum_{i=1}^{n} w_i(\hat{\boldsymbol{p}}_i \hat{\boldsymbol{x}}_i^T)\right) \tag{5.24}$$

The rotation is obtained by finding

$$\boldsymbol{R} = \arg\max_{\hat{\boldsymbol{R}}} \ tr\left(\hat{\boldsymbol{R}}^T \Sigma_{\mathcal{P}\mathcal{X}}\right) \tag{5.25}$$

where $\Sigma_{\mathcal{P}\mathcal{X}}$ is the sample cross-covariance matrix of data sets $\mathcal{P}$ and $\mathcal{X}$

$$\Sigma_{\mathcal{P}\mathcal{X}} = \sum_{i=1}^{n} w_i(\hat{\boldsymbol{p}}_i \hat{\boldsymbol{x}}_i^T) \tag{5.26}$$

The matrix $\Sigma_{\mathcal{P}\mathcal{X}}$ contains all the required information for computing the optimal rotation matrix. In the following subsections, three methods for computing $\boldsymbol{R}$ are shortly reviewed.

5.4.1 Solution by Singular Value Decomposition

Schönemann [SC70], Arun *et al.* [AHB87] and Haralick *et al.* [HJL$^+$89] find the optimal $\boldsymbol{R}$ by singular value decomposition (SVD). If $\Sigma_{\mathcal{P}\mathcal{X}} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, then the optimal rotation matrix is simply

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T \tag{5.27}$$

The additional constraint added by Umeyama [Ume91] in order to force the solution to the group $SO(3)$ can be easily expressed as [Kan94] [BNB04]

$$\boldsymbol{R} = \boldsymbol{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\boldsymbol{V}\boldsymbol{U}^T) \end{bmatrix} \boldsymbol{U}^T \tag{5.28}$$

5.4.2 Solution by Polar Decomposition

Horn *et al.* [HHN88] find the polar decomposition of $\Sigma_{\mathcal{P}\mathcal{X}} = \boldsymbol{U}\boldsymbol{S}$ where

$$\boldsymbol{S} = (\Sigma_{\mathcal{P}\mathcal{X}}^T \Sigma_{\mathcal{P}\mathcal{X}})^{1/2} \tag{5.29}$$

and

$$\boldsymbol{U} = \boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}(\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}^T\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}})^{-1/2} \tag{5.30}$$

Then, the optimal rotation is $\boldsymbol{R} = \boldsymbol{U}$. The square root of the positive definite matrix $\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}^T\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}$ is obtained from its eigenvalues λ_i and corresponding unit eigenvectors $\boldsymbol{u}_i$

$$(\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}^T\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}})^{1/2} = \sqrt{\lambda_1}\boldsymbol{u}_1\boldsymbol{u}_1^T + \sqrt{\lambda_2}\boldsymbol{u}_2\boldsymbol{u}_2^T + \sqrt{\lambda_3}\boldsymbol{u}_3\boldsymbol{u}_3^T \tag{5.31}$$

5.4.3 Solution by Rotation Quaternions

Horn [Hor87] finds the optimal rotation matrix using *quaternions* (see Appendix A). Through the definition of the anti-symmetric matrix:

$$\boldsymbol{A} = (\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}} - \boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}^T)$$

and a vector consisting of the elements of $\boldsymbol{A}$:

$$\boldsymbol{\Delta} = (\boldsymbol{A}_{2,3}\boldsymbol{A}_{3,1}\boldsymbol{A}_{1,2})^T$$

the optimal rotation quaternion $\mathring{\boldsymbol{q}}$ is found as the unit eigenvector corresponding to the maximum eigenvalue of the matrix:

$$\boldsymbol{Q}(\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}) = \begin{bmatrix} \operatorname{tr}(\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}) & \boldsymbol{\Delta}^T \\ \boldsymbol{\Delta} & \boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}} + \boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}}^T - \operatorname{tr}(\boldsymbol{\Sigma}_{\mathcal{P}\mathcal{X}})\boldsymbol{I}_{3\times 3} \end{bmatrix}$$

The rotation matrix $\boldsymbol{R}$ is obtained from the vector $\mathring{\boldsymbol{q}}$ as:

$$\boldsymbol{R} = \begin{bmatrix} (q_0^2 + q_x^2 - q_y^2 - q_z^2) & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 2(q_yq_x + q_0q_z) & (q_0^2 - q_x^2 + q_y^2 - q_z^2) & 2(q_yq_z - q_0q_x) \\ 2(q_zq_x + q_0q_y) & 2(q_zq_y - q_0q_x) & (q_0^2 - q_x^2 - q_y^2 + q_z^2) \end{bmatrix}$$

The computation cost of the WLS operation is $O(n)$, where n is the number of points in the sets. In Chapter 8 some computation times are shown.

5.5 Matrix Weighted Total Least Squares Formulation

A Least Squares approach assumes that the errors are present only in the observation vector, and that the error is equal for all components of the vector. If we measure the 3D position of a world point with stereo at two different times, we expect to obtain two noisy measurements and not only one. The assumption of having equal noise for all the components of the measured data is also too optimistic in most cases. For example, in stereo triangulation the largest uncertainty is given in the viewing direction and it is smaller perpendicular to it.

Much better results are expected if the error for each measurement is modeled with the full covariance matrix instead of a single weighting scalar. Matthies and Shafer [MS87] were the first to propose the use of the full covariance matrix for ego-motion estimation. In the paper, the authors notice the difficulty in eliminating the rotation from the weighting matrix $\boldsymbol{W}_i$ and solve the problem with an iterative method.

Some years later, Weng *et al.* [WC90] proposed a closed form solution, in which the rotation matrix $\boldsymbol{R}$ in $\boldsymbol{W_i}$ (Equation 5.14) is replaced with a constant rotation matrix $\hat{\boldsymbol{R}}$ as an approximation to the sought real rotation matrix $\bar{\boldsymbol{R}}$. If the rotation is assumed to be small then $\boldsymbol{R} \simeq \hat{\boldsymbol{R}} = \boldsymbol{I}_{3\times 3}$. In this case the weighting matrix of Equation 5.14 becomes $(\boldsymbol{\Gamma}_{x,i} + \boldsymbol{\Gamma}_{p,i})^{-1}$ and $\boldsymbol{W}_i$ "does not depend very much on $\boldsymbol{R}$" [WC90]. If the rotation to be estimated can not be approximated with the identity matrix, $\hat{\boldsymbol{R}}$ is obtained from the scalar weighted Least Square solution. In either case, the objective function is simplified to a linear expression in the elements of $\boldsymbol{R}$, and the optimal rotation matrix is found as the closest orthonormal matrix which solves the linearized objective function.

In order to find a closed form solution, we cannot proceed as we did for the scalar case. Observe that the weights w_i are now the matrices $\boldsymbol{W}_i$. The commutation performed between weighting factor and rotation in Equation 5.20 cannot be done, since the weight is now a matrix and the result is not 0 any more. Nevertheless, a solution for translation can be found by means of the following theorem, whose proof can be found in [WC90]:

Matrix-Weighted Centroid-Coincidence Theorem. If $\boldsymbol{R}$ and $\boldsymbol{t}$ minimize Equation 5.13 and $\boldsymbol{W}_i$ does not depend on either $\boldsymbol{R}$ or $\boldsymbol{t}$, then the weighted centroids of $\boldsymbol{p}_i$ and $\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ must coincide, i.e.

$$\sum_{i=1}^{n} \boldsymbol{W}_i(\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t}) = \sum_{i=1}^{n} \boldsymbol{W}_i\boldsymbol{p}_i \tag{5.32}$$

Solving for $\boldsymbol{t}$ we obtain

$$\boldsymbol{t} = (\sum_{i=1}^{n} \boldsymbol{W}_i)^{-1} \sum_{i=1}^{n} \boldsymbol{W}_i\boldsymbol{p}_i - \sum_{i=1}^{n} \boldsymbol{W}_i\boldsymbol{R}\boldsymbol{x}_i \tag{5.33}$$

The objective function of Equation 5.13 can be rewritten as

$$J = \sum_{i=1}^{n} \|\boldsymbol{V}_i(\boldsymbol{p}_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t})\|^2 \tag{5.34}$$

where $\boldsymbol{W}_i = \boldsymbol{V}_i^T\boldsymbol{V}_i$. Substituting Equation 5.33 to Equation 5.34, we obtain an expression which not only depends quadratically on rotation, but where rotation is constrained to remain in the sum of vectors. This can be seen in Equations 5.33 and 5.34 where $\boldsymbol{R}$ is pre-multiplied by the weighting matrix and post-multiplied by vector coordinate $\boldsymbol{x}_i$. In the scalar case, the rotation matrix could be moved out of the sum (Equation 5.20) because scalar multiplication is commutative but matrix multiplication is not. In order to find a closed-form solution we must first be able to find an expression which depends linearly on rotation and then solve for $\boldsymbol{R}$.

If the matrix $\boldsymbol{R}$ is composed of column vectors $(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$, the following 9 dimensional vector is defined.

$$\boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \end{bmatrix}. \tag{5.35}$$

With this notation, a point $\boldsymbol{z}$ can be rotated if it is transformed with the following mapping

$$\boldsymbol{C}(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{z}^T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{z}^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{z}^T \end{bmatrix} \tag{5.36}$$

such that

$$\boldsymbol{R}\boldsymbol{z} = \boldsymbol{C}(\boldsymbol{z})\,\boldsymbol{r} \tag{5.37}$$

Substituting Equation 5.37 to Equation 5.33 we obtain

$$\begin{aligned} \boldsymbol{t} &= \underbrace{(\sum_{i=1}^{n} \boldsymbol{W}_i)^{-1} \sum_{i=1}^{n} \boldsymbol{W}_i \boldsymbol{p}_i}_{\boldsymbol{d}} - \underbrace{\sum_{i=1}^{n} \boldsymbol{W}_i \boldsymbol{C}(\boldsymbol{x}_i)}_{\boldsymbol{D}}\ \boldsymbol{r} \\ &= \boldsymbol{d} - \boldsymbol{D}\,\boldsymbol{r} \end{aligned} \tag{5.38}$$

We substitute also Equation 5.37 to Equation 5.34 to obtain

$$\begin{aligned} J &= \sum_{i=1}^{n} \|\boldsymbol{V}_i(\boldsymbol{p}_i - \boldsymbol{C}(\boldsymbol{x}_i)\boldsymbol{r} - \boldsymbol{D}\,\boldsymbol{r})\|^2 \\ &= \sum_{i=1}^{n} \|\underbrace{\boldsymbol{V}_i(\boldsymbol{p}_i - \boldsymbol{d})}_{\boldsymbol{b}_i} - \underbrace{\boldsymbol{V}_i(\boldsymbol{C}(\boldsymbol{x}_i) - \boldsymbol{D})}_{\boldsymbol{A}_i}\ \boldsymbol{r}\|^2 \\ &= \sum_{i=1}^{n} \|\boldsymbol{b}_i - \boldsymbol{A}_i\,\boldsymbol{r}\|^2 \end{aligned} \tag{5.39}$$

Stacking all $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$ on top of each other

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_n \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_n \end{bmatrix} \tag{5.40}$$

Equation 5.39 can be rewritten in matrix form as

$$J = \|\boldsymbol{b} \quad - \quad \boldsymbol{A}\,\hat{\boldsymbol{r}}\|^2 \tag{5.41}$$

The minimization of Equation 5.41 still requires an iterative approach because of the orthogonality constraints in $\boldsymbol{r}$, which are hard to enforce. The proposed solution computes directly the unconstrained solution for $\boldsymbol{r}$ and finds the closest rotation matrix to the solution found. The optimal solution to Equation 5.41 without constraints is given by

$$\begin{aligned} \check{\boldsymbol{r}} &= (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} && (5.42) \\ &= \left[\sum_{i=1}^{n}(\boldsymbol{C}(\boldsymbol{x}_i) - \boldsymbol{D})^T\boldsymbol{W}_i(\boldsymbol{C}(\boldsymbol{x}_i) - \boldsymbol{D})\right]^{-1}\left[\sum_{i=1}^{n}(\boldsymbol{C}(\boldsymbol{x}_i)\boldsymbol{W}_i(\boldsymbol{p}_i - \boldsymbol{d})\right] && (5.43) \end{aligned}$$

By remapping the optimal unconstrained solution $\check{\boldsymbol{r}}$ to the original 3×3 matrix convention, the closest rotation matrix $\boldsymbol{R}$ and its corresponding $\boldsymbol{r}$ are found by any of the methods specified in Section 5.4. Translation is then obtained from Equation 5.38.

Since the vector obtained from Equation 5.41 is not constrained to be a rotation, but instead the rotation is found by projecting the optimal unconstrained solution into the parameter manifold, the rotation matrix obtained is not optimal any more. Weng *et al*. argue that the penalty imposed in the solution might be "much less significant than the penalty otherwise caused by improper weighting" [WC90]. As shown later in Chapter 7, this is not always the case.

5.6 Summary

The computation of the absolute orientation has been approached many times in the literature. One could get the impression that most of approaches were proposed without the knowledge of previous investigations. This led to the re-discovery of the same methods many times. This happened with the quaternions-based solution and the SVD-based method. Up to date, four methods are known to solve the weighted-least squares approach to the absolute orientation problem in closed form. The two mentioned above, a method based on polar decomposition and a fourth method based on dual-quaternions. The total least squares approaches to the absolute orientation problem are iterative. A closed form solution can be found if an approximation to the sought rotation matrix is available. Nevertheless, the solution found is penalized, since the rotation matrix is found by projecting an unconstrained solution into the parameter manifold.

Between all method for absolute orientation review in this chapter, the quaternion-based method has been selected for the implementation since it allows fast and numerically stable algorithms. The closed form solution of the total least squares formulation was also implemented and will be analyzed in Chapter 7 when computing the ego-motion of the camera.

Chapter 6

Modeling Error in Stereo Triangulation

6.1 Introduction

The estimation of ego-motion is achieved by finding the rotation and translation between the clouds of 3D points obtained with stereo at different time points (frames). It is therefore meaningful to understand the error expected in the 3D points before using them as the input data for ego-motion estimation. This chapter analyzes and model the stereo triangulation error.

When considering the image quantization as the main cause of error in stereo triangulation (because of resolution limits), the hexahedral model approximates the error as equally distributed inside a volume composed as the intersection of two pyramids. If the error distribution of a feature in the image plane can be approximated as Gaussian, the Egg-Shaped model provides a better approximation to the real error distribution in 3D Euclidean space. Since both previous models make the propagation of errors to posterior stages of processing quite difficult, the propagation of the image covariance matrix to the 3D Euclidean space gives place to the ellipsoidal model. The propagation is achieved with a first order approximation of the triangulation equation. This model introduces a bias in the estimate of 3D position. In the next four sections, all these points are analyzed in detail.

6.2 Hexahedral Model

Suppose a 3D point P with coordinate $\bar{\boldsymbol{p}} = (\bar{x}, \bar{y}, \bar{z})$ is projected on the left and right images of a stereo camera system at pixels positions (u_l, v_l) and (u_r, v_r) respectively. Under the assumption of standard stereo $v_l = v_r$ (because of the epipolar constraint) and $d = u_l - u_r$ is the integer disparity corresponding to the 3D point. The recovery of the 3D point through triangulation will produce the coordinate $\boldsymbol{p'} = (x', y', z')$ which is an estimate of the actual point position $\bar{\boldsymbol{p}}$

$$\boldsymbol{p'} = \boldsymbol{g}((u_l, v_l, d)^T) = \frac{B}{d} \begin{bmatrix} u' \\ v' s_{vu} \\ f_u \end{bmatrix} \tag{6.1}$$

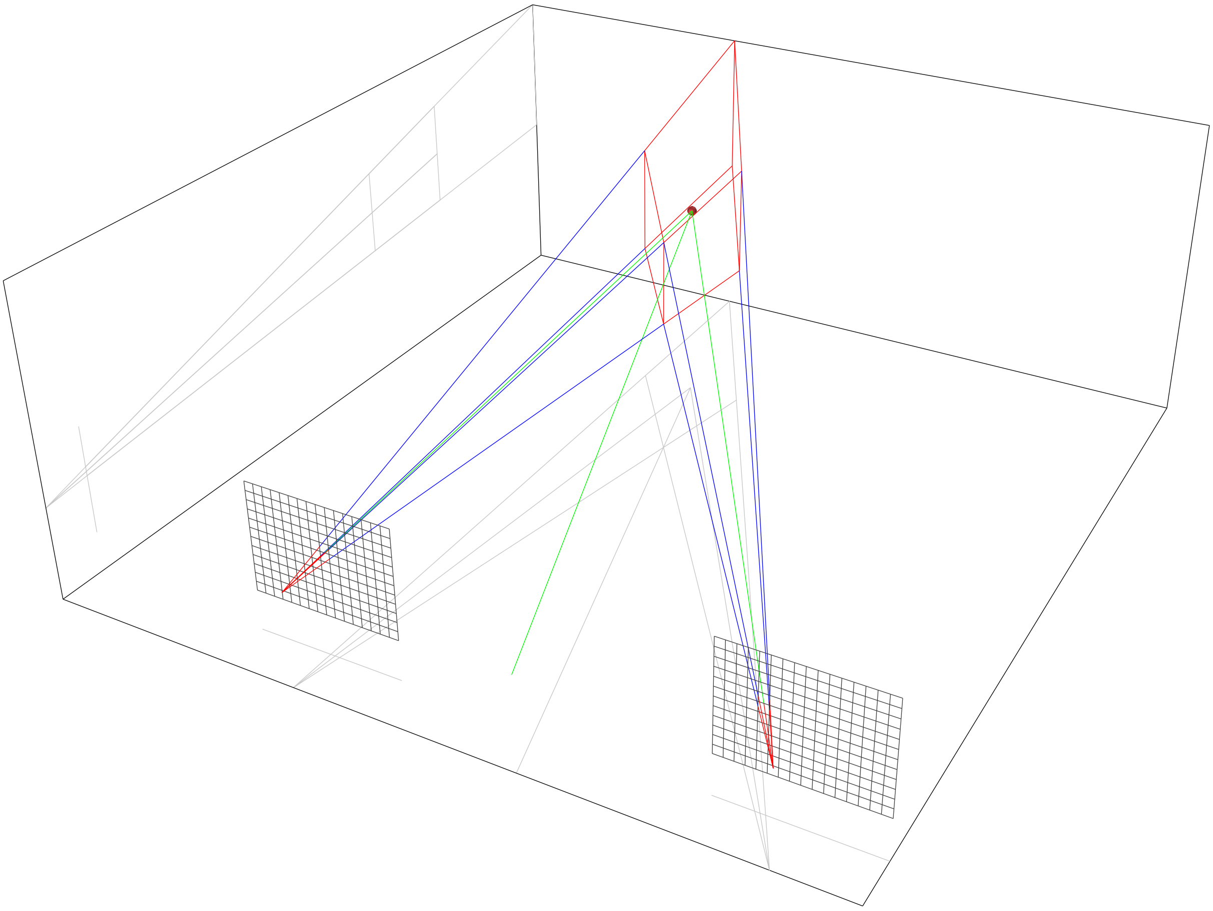


Figure 6.1: Hexahedral model of uncertainty in 3D Recovery.

where $u' = u_l - u_0$, $v' = v_0 - v_l$, $s_{vu} = s_v/s_u$, $f_u = f/s_u$, B is the baseline and (u_0, v_0) is the principal point in the left image. Under this configuration, the projections of P in the left and right cameras are correct up to one pixel. Thus the uncertainty in the projection of a point, for both left and right images is $1\ pixel$. Assuming rectangular pixels, the uncertainty in the projection of the point $\boldsymbol{p'}$ as defined above, produces a *volume of uncertainty* for the recovery of the 3D point. An illustration of this can be seen in Figure 6.1. Every 3D point inside the red hexahedron would be projected into the same image coordinates (u_l, u_l) and (u_r, u_r), so the actual point P could be located anywhere in the hexahedron. Thus, the uncertainty is given by the volume of the hexahedron. The vertices of the hexahedron are defined as the intersection of the rays of each camera with the image plane at half pixel displacement around the projected points.

For a given camera configuration the shape of the hexahedron changes with image position and disparity, but the volume is a function only of the disparity, and is given by [Bad05]

$$V_d = \frac{b^3 f_u}{(d^2-1)^2}\left(1 - \frac{1}{3d^2}\right) \tag{6.2}$$

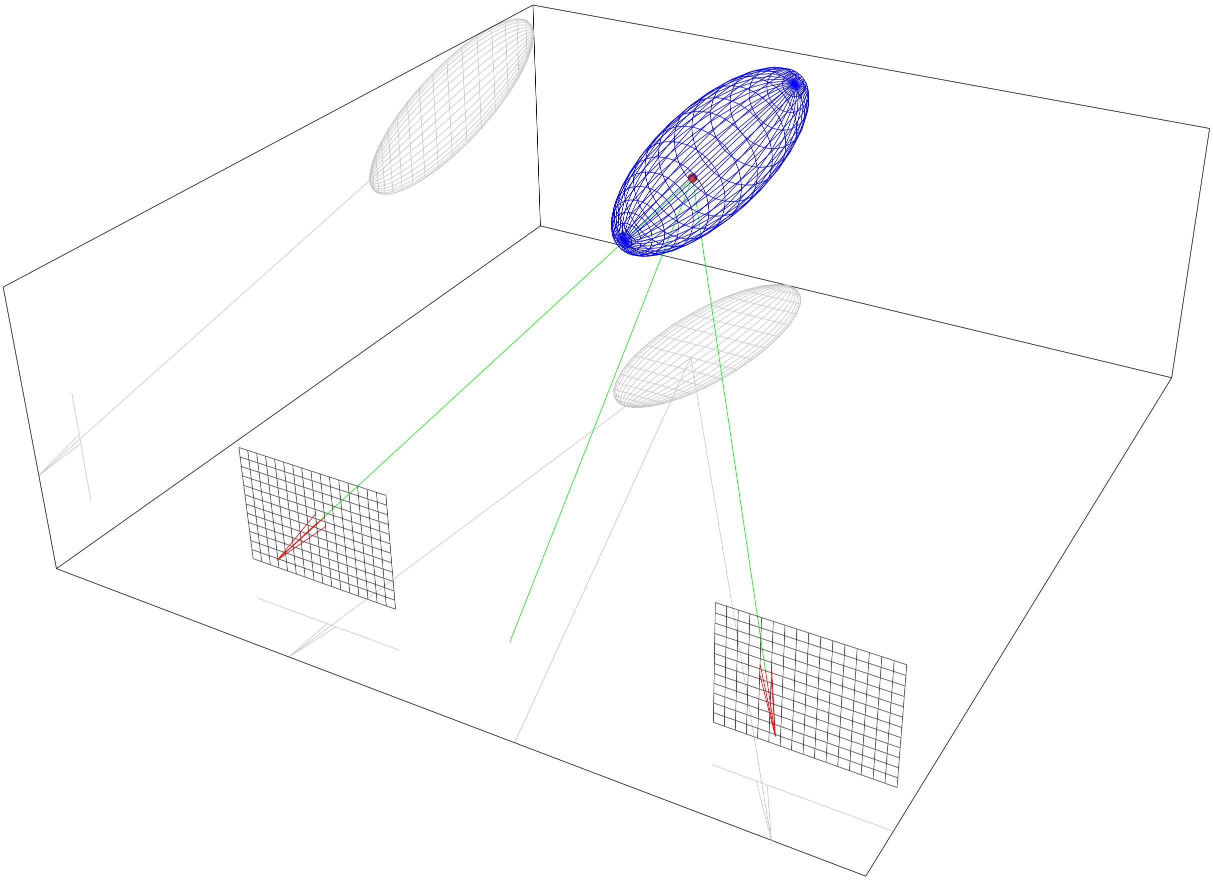


Figure 6.2: Egg-shaped Ellipsoidal model of uncertainty in 3D Recovery.

6.3 Egg-Shaped Ellipsoidal Model

The computation of a 3D point is usually required for the case where image coordinates are non-integer. An example is when a feature point in the image is tracked with sub-pixel precision and a 3D estimate of the feature is required. In this case, the distribution of the feature point position on the image plane can be better approximated by a Gaussian distribution [MLG00] [JL03] [DD01] [DD02] [Mat92]. The real distribution of the 3D point when applying the triangulation Equation 6.1 is not Gaussian, because stereo triangulation is not a linear operation. The geometrical interpretation of the error distribution in Euclidean space is an egg-shaped ellipsoid. An egg-shaped ellipsoid has cylindrical symmetry, but is asymmetric in a plane perpendicular to the long axis, i.e. in this case; the viewing direction. Figure 6.2 shows an example.

6.4 Ellipsoidal Model

The volume of uncertainty is an estimate of the accuracy expected for the 3D position of a point computed with stereo, but says nothing about the distribution of the error in space. As already mentioned, the hexahedron changes its form with image position and disparity. Therefore a better model of the reality must include this change in shape. The exact modeling of the error distribution with the hexahedral or egg-shaped model complicates the propagation of errors to posterior stages of computation. Nevertheless, if the error in the image plane of a feature point is modeled with a normal distribution, the propagation of errors in 3D Euclidean space can be carried out with a first order approximation of the triangulation equation.

Let us suppose again that the measurement $\boldsymbol{m} = (u, v, d)^T$ can be modeled as normally distributed with covariance matrix

$$\boldsymbol{\mathcal{R}} = \begin{bmatrix} \sigma_u^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_d^2 \end{bmatrix}. \tag{6.3}$$

If the triangulation function $\boldsymbol{g}(\boldsymbol{m})$ of Equation 6.1 were linear, $\boldsymbol{p'}$ would also be normal distributed with covariance matrix

$$\boldsymbol{\Gamma}_{\boldsymbol{p'}} = \boldsymbol{G}\boldsymbol{\mathcal{R}}\boldsymbol{G}^T = \frac{B^2}{d^4} \begin{bmatrix} u'^2\sigma_d^2 + d^2\sigma_u^2 & s_{vu}u'v'\sigma_d^2 & f_u u'\sigma_d^2 \\ s_{vu}u'v'\sigma_d^2 & s_{vu}^2(v'^2\sigma_d^2 + d^2\sigma_v^2) & f_u s_{vu}v'\sigma_d^2 \\ f_u u'\sigma_d^2 & f_u s_{vu}v'\sigma_d^2 & f_u^2\sigma_d^2 \end{bmatrix}. \tag{6.4}$$

where $\boldsymbol{G}$ is the matrix of first partial derivatives of $\boldsymbol{g}$. Since $\boldsymbol{g}(\boldsymbol{m})$ is not linear in the elements of $\boldsymbol{m}$ (it involves the division by the disparity), the triangulated 3D point $\boldsymbol{p'}$ is not normally distributed and might lead to an estimation bias for small disparities if higher order moments are neglected, as shown in the next section. Nevertheless, the covariance matrix of Equation 6.4 models the shape of the uncertainty, which is a much better approximation than assuming equal noise for all components. An example is shown in Figure 6.3. Observe how the geometrical representation of the second order moment differentiates from the egg-shape model: the error distribution is assumed to be pure ellipsoidal, and also symmetric with respect to a plane perpendicular to the viewing direction.

An alternative expression for Equation 6.4 is obtained replacing the image coordinate with the corresponding 3D Euclidean coordinate according to the triangulation Equation 2.12,

$$\boldsymbol{\Gamma}_{\boldsymbol{p'}} = \frac{Z^2}{B^2 f_u^2} \begin{bmatrix} \sigma_d^2 X^2 + B^2\sigma_u^2 & \sigma_d^2 XY & \sigma_d^2 XZ \\ \sigma_d^2 XY & \sigma_d^2 Y^2 + B^2 s_{vu}^2\sigma_v^2 & \sigma_d^2 YZ \\ \sigma_d^2 XZ & \sigma_d^2 YZ & \sigma_d^2 Z^2 \end{bmatrix}. \tag{6.5}$$

or equivalently,

$$\boldsymbol{\Gamma}_{\boldsymbol{p'}} = \frac{Z^2}{f_u^2} \left(\frac{\sigma_d^2}{B^2} \boldsymbol{p'}\boldsymbol{p'}^T + \boldsymbol{diag}(\sigma_u^2, s_{vu}^2\sigma_v^2, 0) \right) \tag{6.6}$$

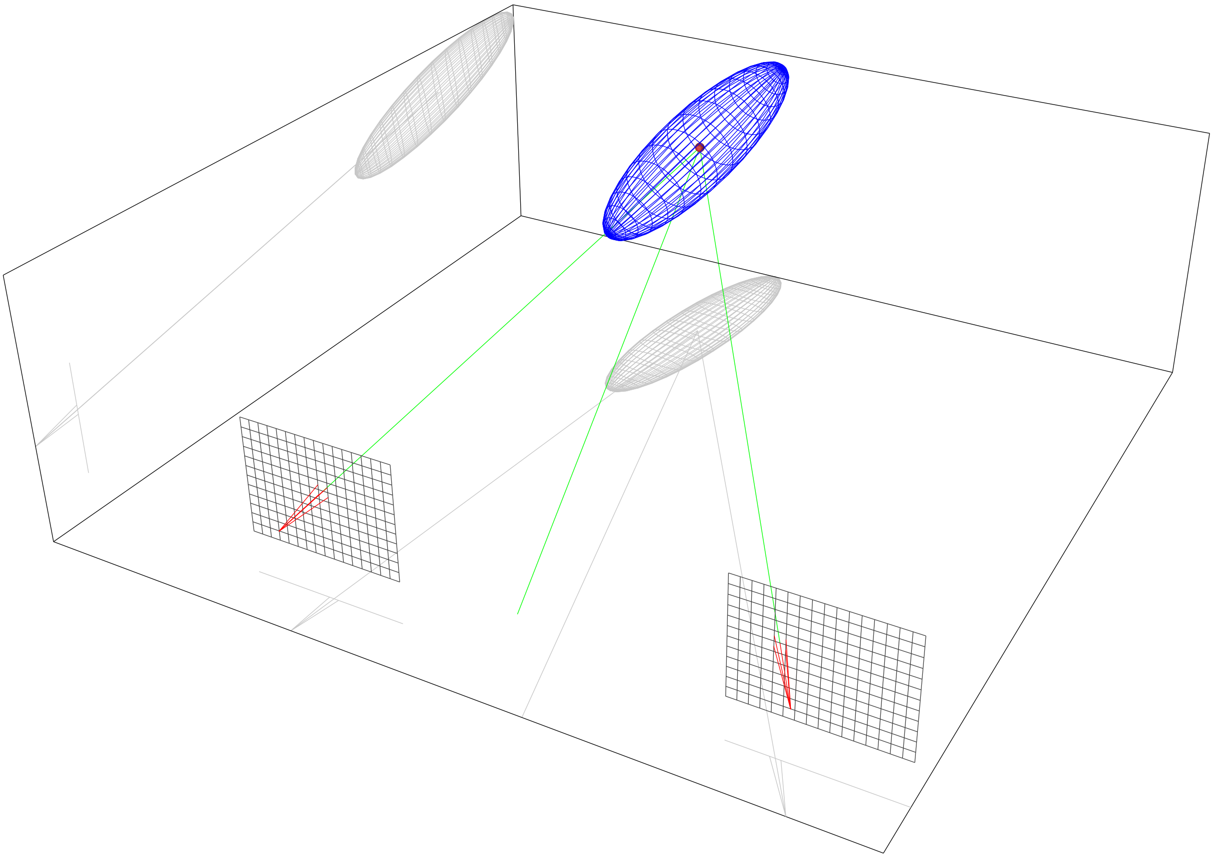


Figure 6.3: Ellipsoidal approximation to the stereo triangulation error.

where $\boldsymbol{diag}(a, b, c)$ is the diagonal 3×3 matrix with the scalars a, b, c in the diagonal. The first term of the sum makes in general the largest contribution to the covariance matrix, while the second term accounts for the uncertainty of the feature point in the image plane [Müh06]. The high anisotropy of the noise is given by the factor Z^2, which becomes even more pronounced because of the term $\boldsymbol{p'p'}^T$ when $Z \gg X, Y$, which is almost always the case in outdoor environments.

Observe finally that the covariance matrix in Equation 6.4 is automatically modeled by the Kalman filter in Equation 4.42.

6.5 Biased Estimation of 3D Position

The covariance propagation from the image plane to the 3D Euclidean space with a first order approximation leads to a statistical bias in the estimation of the 3D position. As shown in the previous section, the real form of the error distribution is egg-shaped, but approximated with a pure ellipsoid. The deformation factor of the egg-shape increases with distance as shown in this section, and the first order approximation deviates from the true distribution. An example is shown in Figure 6.4, where the real egg-shaped distribution is modeled with an ellipsoid. The p.d.f. of the 3D point can be obtained from the p.d.f. of its corresponding projection in the image plane. For instance, the p.d.f. f_z for the distance of a point with disparity

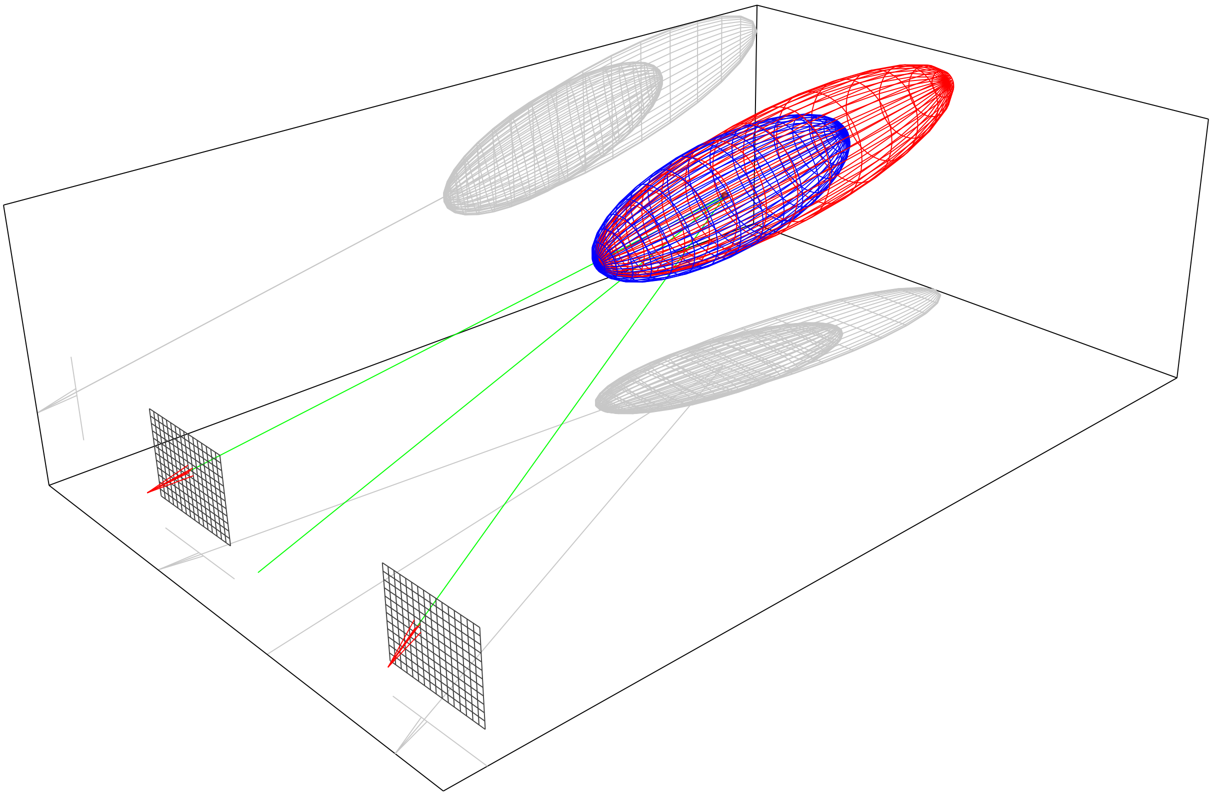


Figure 6.4: Ellipsoidal (red) and Egg-shaped (blue) distributions.

p.d.f. f_d is obtained by [Mat92], [SMS05]

$$f_z(z) = f_d(d) \left| \frac{\partial f_d(d)}{\partial z} \right| \tag{6.7}$$

$$= f_d(g_3^{-1}(z)) \left| \frac{\partial g_3^{-1}(z)}{\partial z} \right| \tag{6.8}$$

$$= f_d(Bf_u/z) \left| \frac{-Bf_u}{z^2} \right| \tag{6.9}$$

where the monotonic function $g_3^{-1}(z)$ is the third component of the inverse continuously differentiable Equation 6.1. Modeling the disparity as Gaussian[1] with mean μ_d and variance σ_d^2, the p.d.f. for the distance z is

$$f_z(z) = \frac{Bf_u}{z^2\sqrt{2\pi\sigma_d^2}} \exp\left(\frac{-(Bf_u/z - \mu_d)^2}{2\sigma_d^2}\right) \tag{6.10}$$

with expected value

$$E[z] = \mu_z = \int_{-\infty}^{\infty} z f_z(z) dz \tag{6.11}$$

Plots of the p.d.f. $f_z(z)$ at three different distances are shown in Figure 6.5(a). Observe that the expected value μ_z diverges from the true value z, proving the existence

---

[1] See [RA90] for the p.d.f. assuming a uniform distribution.

of a bias in the estimation. The bias is caused by the non-gaussianity of the p.d.f. which exhibits a tail in the positive direction of distance. For small distances, the tail is negligible and the p.d.f. is almost Gaussian. For large distances (i.e. small disparities), the tail shifts the mean away and causes the bias in the estimation. The length of the tail increases not only with distance, but also with the variance of the disparity, as shown in Figure 6.5(b).

The bias in the estimation can be reduced to a negligible value if the real variance of the disparity is known. This can be achieved with a Taylor expansion up to the second order of the triangulation equation around the unknown real disparity $\bar{d}$

$$\tilde{g}_3(d) \approx g_3(d) + (\bar{d} - d)\frac{\partial g_3(d)}{\partial d} + \frac{1}{2}(\bar{d} - d)^2\frac{\partial^2 g_3(d)}{\partial^2 d} \tag{6.12}$$

and taking the expectation [SMS05],

$$\tilde{g}_3(d) \approx g_3(d) + E(\bar{d} - d)\frac{\partial g_3(d)}{\partial d} + \frac{1}{2}E(\bar{d} - d)^2\frac{\partial^2 g_3(d)}{\partial^2 d}. \tag{6.13}$$

Disparity is assumed to be zero-mean Gaussian distributed and therefore the expression $E(\bar{d} - d)$ in the second term is 0. Noting that $E(\bar{d} - d)^2 = \sigma_d^2$ and replacing $\bar{d}$ with d the new distance triangulation equation results in:

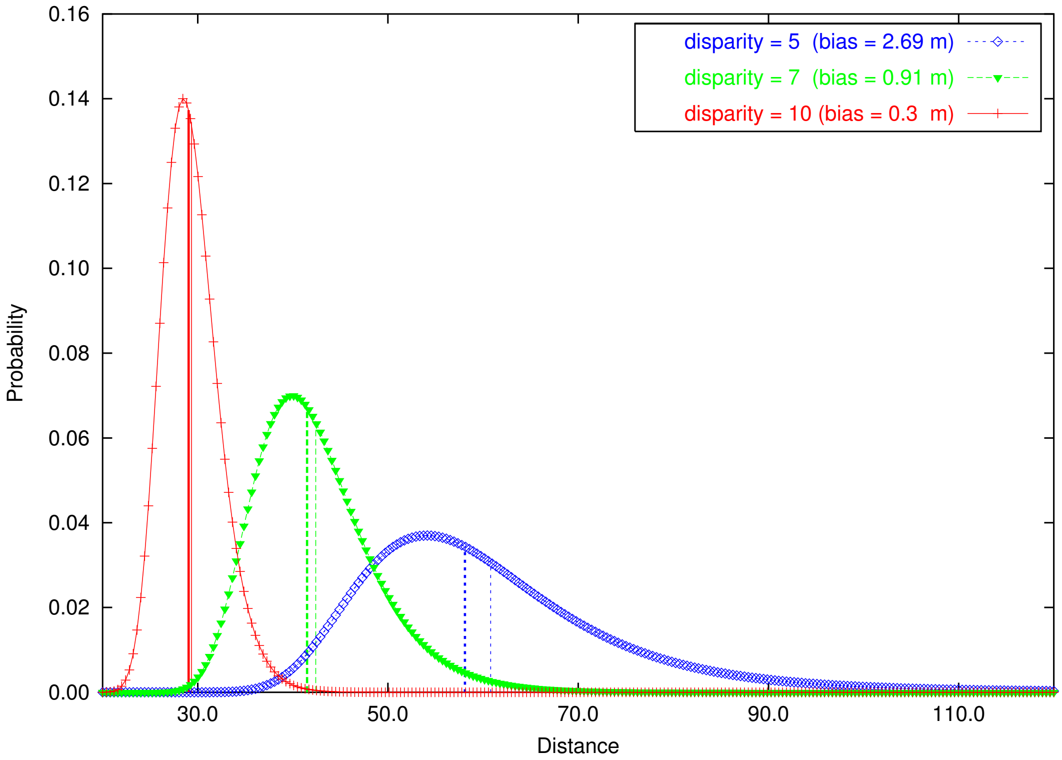
$$g_3(d) \approx \frac{Bf_u}{d} - \sigma_d^2\frac{Bf_u}{d^3} = Bf_u\left(\frac{1}{d} - \frac{\sigma_d^2}{d^3}\right) \tag{6.14}$$

The extension for the other two components of the 3D point is straightforward, since the correction is applied only to the measured disparity. If $\tilde{d}$ is the measured disparity of a feature point, the 3D position is still obtained with Equation 6.1 but with disparity

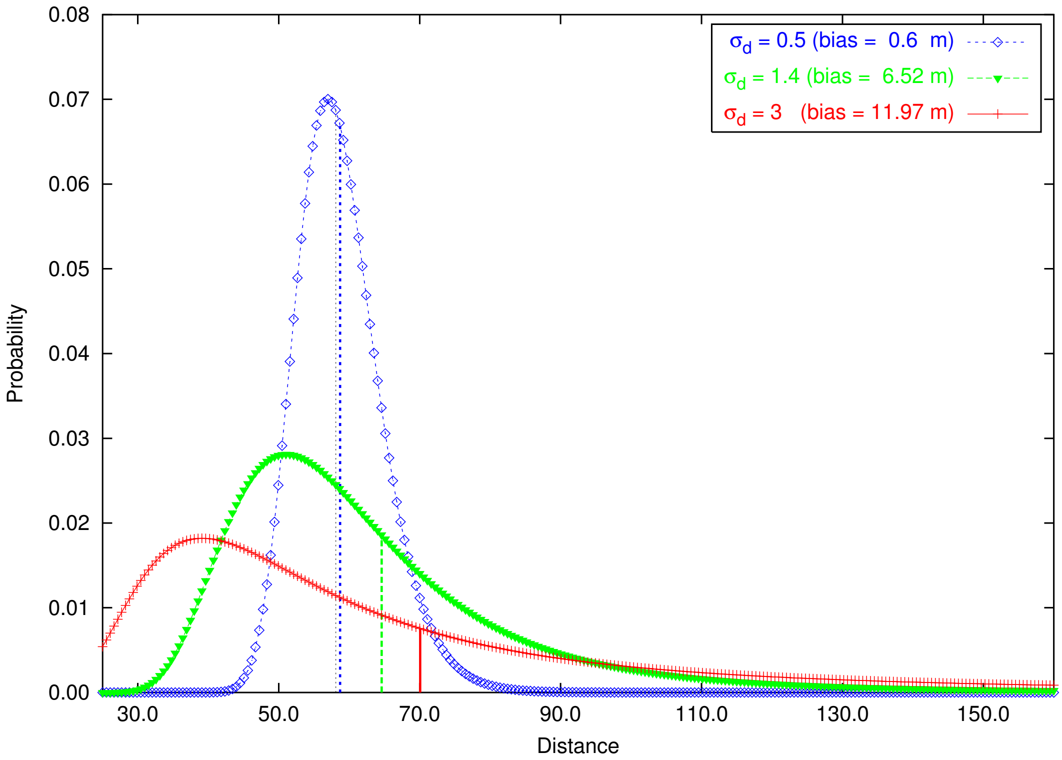
$$d = \frac{1}{\frac{1}{\tilde{d}}\left(1 - \frac{\sigma_d^2}{\tilde{d}^2}\right)} \tag{6.15}$$

The bias-correction with a Taylor series expansion of the triangulation equation has two main drawbacks. First of all, the real disparity variance must be known, which is not feasible. Another problem is given by the second order approximation of the triangulation equation. If the real variance of the disparity is large, Equation 6.12 does not hold any more and higher order terms are required. This means not only knowing the disparity variance, but also of its skewness, kurtosis and higher order moments, which once again, is not feasible. The advantage of this bias-correction method is given by its simplicity: only the disparity must be corrected.

Figure 6.6(a) shows the estimated bias computed over 10^6 trials and assuming a disparity variance of $0.4\ pixels^2$. Observe that the bias is always positive overestimating the true value, as it was already shown in Figures 6.5(a) and 6.5(b). The bias-correction is also shown in the same figure, where it can be seen that the bias is greatly reduced, but now underestimating the true value. Higher order moments would be required to completely eliminate the estimation bias. In fact, the bias correction with second order approximation of the triangulation equation fails when the

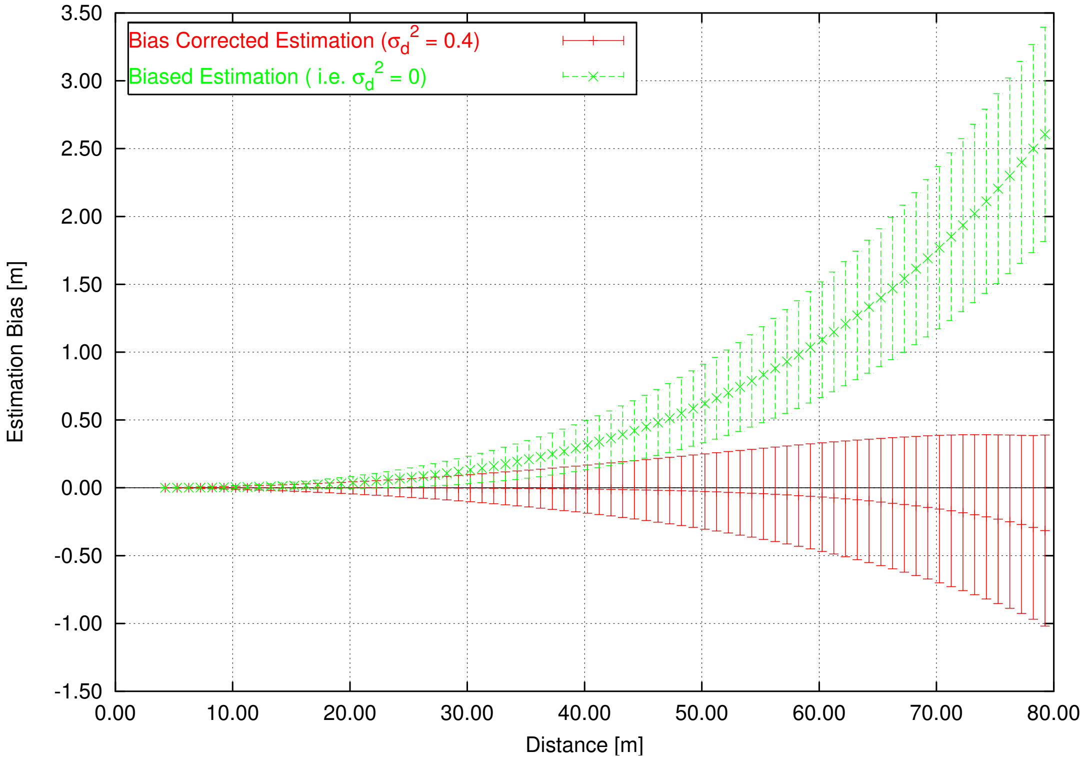


(a) PDFs for three different depths and constant measurement noise. The thick vertical lines show the real disparity while the thin vertical lines show the biased estimate.

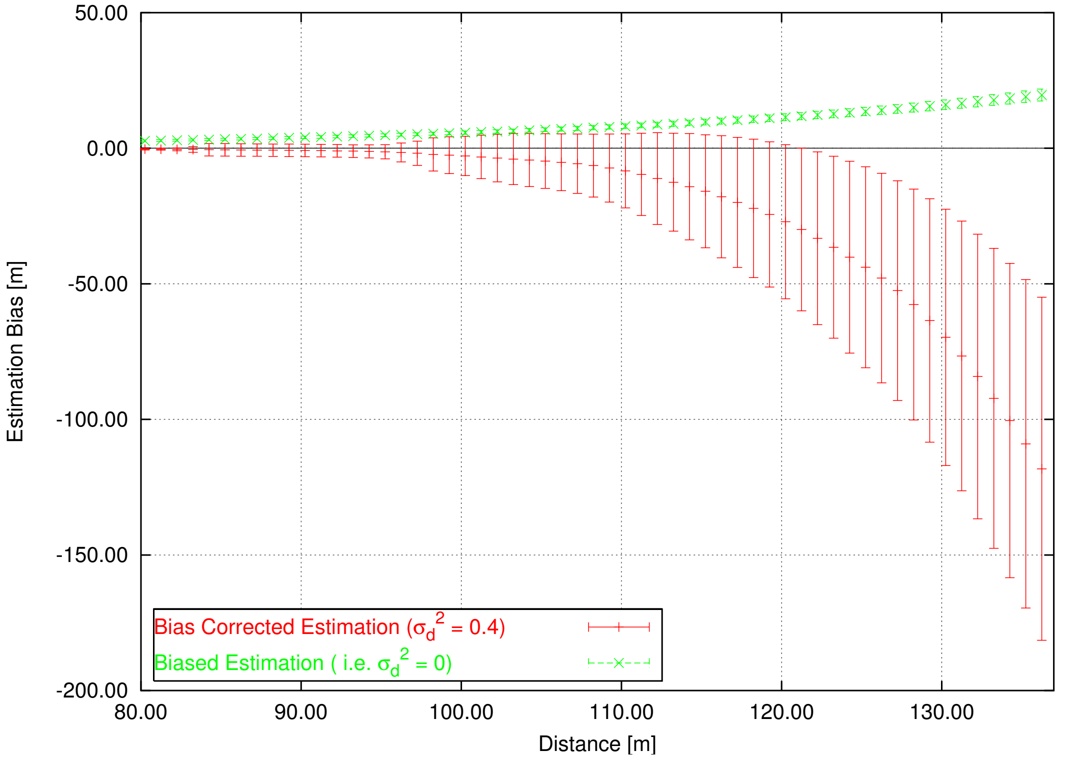


(b) PDFs for the same depth with varying measurement noise. The black dotted vertical line shows the real disparity while the colored lines show the biased estimate for each distribution.

Figure 6.5: p.d.f.s for the estimated distance with stereo triangulation



(a) Unbiased and bias-corrected estimation of distance ($1.e6$ trials). The uncertainty corresponds to a standard deviation of $1/10 pixels$.



(b) Unbiased and bias-corrected estimation of distance for small disparities. The uncertainty corresponds to a standard deviation of $1/50 pixels$.

Figure 6.6: Bias reduction for distance estimation trough stereo triangulation.

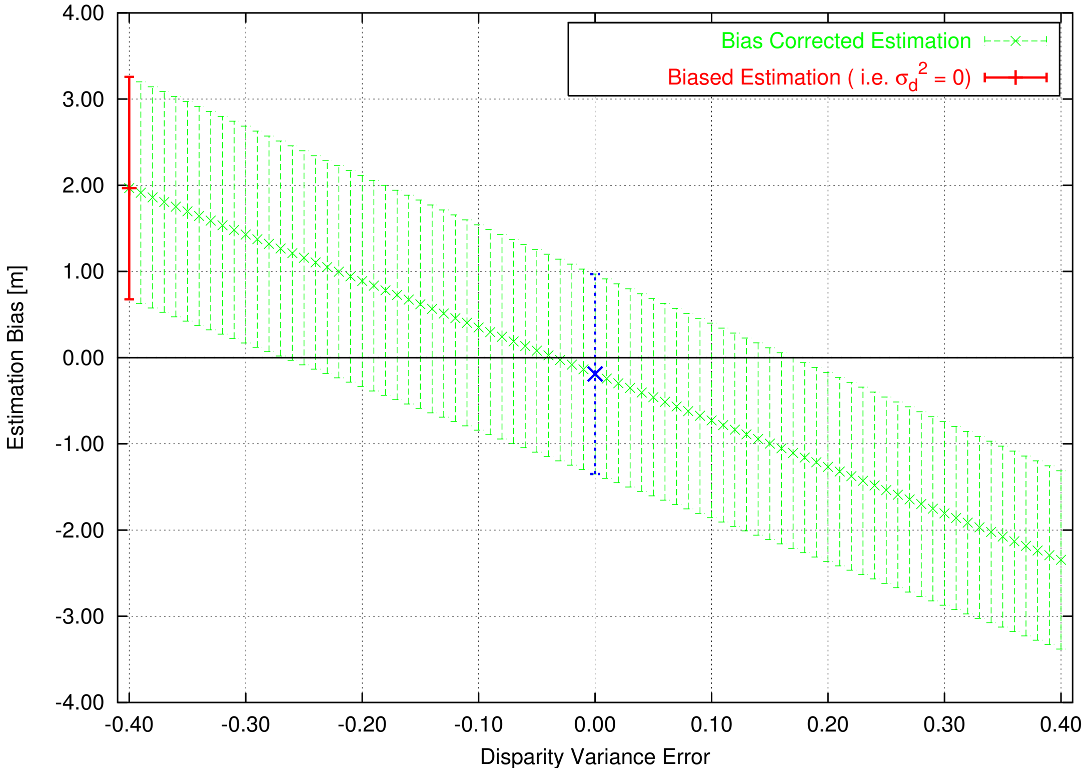


Figure 6.7: Bias obtained for varying σ_d^2 for a feature point with disparity 4 pixels (10^6 trials). The real disparity variance is $0.4\ pixels^2$. The uncertainty corresponds to a standard deviation of $1/10 pixels$. The bias obtained applying the real disparity variance ($\sigma_d^2 = 0.4$) is shown in blue.

signal-to-noise ratio decreases. Figure 6.6(b) shows the plot which follows from Figure 6.6(a) and reflects this effect. Observe that the "bias corrected curve" has a larger bias than the uncorrected curve. According to our experiments the bias-correction of Equation 6.15 fails for values of $d/\sigma_d^2 < 5.18$, factor which is independent of the factor Bf_u and therefore independent of the camera configuration.

Since an accurate estimation of the disparity variance σ_d^2 of Equation 6.15 is not feasible, a sensitivity analysis w.r.t. the change in the bias is required. Figure 6.7 shows the bias for a point with mean disparity $4\ pixels$ and variance $0.4\ pixels^2$ for varying σ_d^2. The plot shows an almost linear relationship between expected bias and error in disparity variance. A variance with 50% error allows approximately 50% of bias reduction. This means that the application of Equation 6.15, even with relative large variance error ($\pm 80\%$), helps in the reduction of the estimation bias.

6.6 Summary

The absolute orientation is computed between clouds of points obtained with stereo. The error of a 3D point obtained with stereo is characterized by the error of the projected point onto the image and the triangulation equation. The hexahedral model is used when it can be assumed that the image quantization is the main cause error.

The egg-shaped model provides a better approximation to the real error distribution if the image error is Gaussian. Approximating the latter with up to second order central moments leads to the ellipsoidal model. The propagation of the covariance matrix from image to world coordinates is achieved with a first order approximation of the triangulation equation. Since higher order moments are neglected, a bias in the estimation of 3D position is introduced. The bias is reduced by redefining the triangulation equation. The new triangulation equation implies a correction of the measured disparity, for which the real disparity variance is required. The simulation results shows that a meaningful reduction of the bias in the estimated 3D point position is possible, even when the real disparity variance is coarsely known.

Chapter 7

Robust Real-Time 6D Ego-Motion Estimation

> *"The motion of a spectator who sees an object at rest*
> *often makes it seem as though the object at rest*
> *had acquired the motion of the moving body,*
> *while the moving person appears to be at rest".*
> Leonardo Da Vinci

7.1 Introduction

The extraction of the observed motion of a camera has been an active area of research over the last decades. Ego-motion computation is motivated by applications like autonomous navigation, self-localization, obstacle-detection and scene reconstruction. Ego-motion is also needed by other applications which require the relative orientation of the cameras with respect to a reference frame. Our interest lies in the computation of the six degrees of freedom of the motion of a vehicle in typical traffic situations, as already stated in previous chapters. For that purpose, a binocular platform has been mounted in the vehicle, which provides the main input to the ego-motion algorithm.

Many approaches have been proposed with monocular and multi-ocular platforms. When using more than one camera the scene structure can be directly recovered through triangulation providing 3D points of the environment. Monocular approaches, instead, do not compute the scene structure, or they do it at the cost of integrating measurements of the image points (and possibly other sensors) over a long time, until a reliable structure is obtained. Therefore multi-ocular approaches perform better in most cases.

Computing the ego-motion from an image sequence is obtaining the motion parameters of the camera with respect to a static environment. The scenarios we are interested in are typical traffic situations. Such an environment presents many objects with independent motion, which can cause our estimation to fail if they are considered static. Also the incorrect computation of a 3D position or the incorrect tracking of features can introduce errors in the computation. As already explained in Section 3.4.1, the Kalman filtered estimated velocity of the points cannot be used to check its immobility until ego-motion is computed. Therefore an effective rejection rule must be applied in order to identify which image points are not showing a coherent motion w.r.t. the camera. We propose a very effective constraint, which we call Smoothness Motion Constraint (SMC).

Since the motion of the camera must be provided for each new acquired frame the motion computed corresponds normally to the motion occurred between consecutive frames. The inter-frame motion is computed considering only the current and previous state, which provides the current relative motion. The complete motion is then obtained concatenating the individual estimates. This may lead to poor results because of error accumulation. A more stable estimation can be achieved if we consider not only the last two frames, but also many frames back in time for the computation of ego-motion (Multi-Frame Estimation).

This chapter proposes a robust method for the computation of the six degrees of freedom of ego-motion (three rotation and three translation parameters) and presents simulation results demonstrating the improvements achieved. Parts of this chapter have been published in [Bad04].

7.1.1 Organization of the Chapter

Section 7.2 summarizes the literature review on ego-motion computation. Section 7.3 describes shortly the algorithm. Section 7.4 presents the Smoothness Motion Constraint while Section 7.5 the Multi-Frame Estimation. The integration of Kalman filtered data is carried out in Section 7.6. The integration with inertial sensor information is addressed in Section 7.7. The last section concludes the chapter.

7.2 Literature Review on Ego-Motion Estimation

Ego-motion was and continues to be a very active research area in computer vision. The visual ego-motion (or visual odometry) problem implies the extraction of the motion parameters of the camera between two time instances by analyzing the changes of brightness patterns in the opportunely acquired images. In accordance with the number and types of motion parameters computed, the motion or pose estimation problem takes different names:

- The relative orientation problem: the problem of estimating the rotation between two camera poses and the *relative direction* between optical centers. In 3D Euclidean space this implies the estimation of 3 parameters for rotation and 2 parameters for translation, since although the direction vector implies 3 components (one for each x,y,z axis), they are specified up to an unknown scale factor.

- The absolute orientation problem: the problem of estimating the rotation, translation and scale factor between two sets of points. In 3D Euclidean space

> this implies the estimation of 7 parameters: 3 parameters for rotation, 3 parameters for translation, and 1 parameter for scaling. When the interest lies in estimating the absolute position between two camera poses with fixed internal parameters, the scale factor is omitted. This is equivalent to computing the 6 rigid-body transformation parameters.

All ego-motion methods can be classified as belonging to one of two main groups: monocular methods and multi-ocular methods. Multi-ocular methods recover the rigid body transformation of the camera platform between two time instances. Monocular methods, instead, solve the relative orientation problem. The missing scale factor can be recovered making assumptions about the motion of the camera or about the structure of the scene.

Both groups at the same time can be classified according to the way in which they integrate the time component; optical flow-based methods, normal flow-based methods, direct methods, and landmark-based methods. Optical flow and normal flow methods assume that the flow field is already provided and they are more or less independent of the method used for it. Landmarks-based methods compute motion w.r.t. 2D landmarks extracted normally from evidence grids [MM96]. Direct method, instead, finds the motion parameters by minimizing some function relating the brightness patterns in consecutive images. This section carries out a brief review of the literature on ego-motion estimation according to this classification.

7.2.1 Monocular methods

Longuet Higgins and Prazdny [LHFP80] were about the first in deriving the relationship between motion parameters and induced motion field in the image plane. Two very important cues are remarked in their paper; the focus of expansion is identified as the 'vanishing point' of all optical flow vectors once eliminated rotation, and the structure of the scene can be achieved only with the translational components, because the motion field induced by rotation is independent of the structure of the scene.

Bruss and Horn [BH81] were one of the first in proposing a solution to the problem of passive navigation using optical flow with a monocular camera. The paper reviews the equations describing the relation between motion of the camera and the optical flow generated. The authors demonstrate that based only on the information provided by optical flow, the motion parameters cannot uniquely be computed. The authors deal separately with the translational and rotational case, and also the general motion case. For the pure translational case, different norms for the least squares formulation are developed and closed form solutions are derived. Special cases where the problem does not have a solution are also addressed. A least squares formulation is also carried out for the rotational case as well as the general motion case.

Tian *et al.* [TTH96] evaluate six methods for ego-motion estimation: Bruss and Horn [BH81], Jepson and Heeger [JH92], Tomasi and Shi [TS93], Prazdny [Pra80] and two approaches proposed by Kanatani [Kan93]. The evaluation was carried out considering bias and sensitivity of the estimates to noise in the flow data, and for quantifying the convergence properties of those algorithms that require numerical search. They found very interesting results: (1) The 6 algorithms were insensitive to to the axis of rotation (e.g. translation along the X-axis and rotation about the Y-axis). This is given by the fact that five of the six algorithms eliminate rotation by algebraic substitution or another technique. Algorithms which do not eliminate rotation are affected by the rotation angle. (2) Fixating does not help when the noise is independent of image velocities. Fixation helps if the noise is proportional to speed, but this is only for the trivial reason that the speeds are slower under fixation. (3) Increasing the FOV at the expense of losing resolution yield worse performance, but increasing the FOV and the resolution give much better results. The Bruss-Horn algorithm [BH81] exhibited the best overall performance followed by the Jepson-Heeger algorithm [JH92]. They found also that the Prazdny algorithm was the most affected by noise.

Suzuki and Tomasi [SK99] estimate camera motion and orientation for vehicle applications measuring the optical flow on the road. The computed optical flow is used to satisfy two equations relating the flow with the orientation and motion of the camera. Kalman filters are used to integrate measurements over time and reduce noise. Carlsson [Car91] applies also linearized Kalman Filters for the computation of structure from motion and motion parameters from optical flow.

Behrad *et al.* [BSM01] present a detection and tracking algorithm for the detection of moving targets. Background motion is obtained by defining a motion model based on affine transformation. Least median squares is used in order to determine the affine transformation between two consecutive frames. A split and merge algorithm is applied for segmentation. The tracking of features is performed using a dynamic tracking algorithm.

Direct methods for computing ego-motion have been first proposed by Negahdaripour and Horn [NH87]. In this paper the authors determine the ego-motion relative to a planar surface directly from the image brightness derivatives. Optical flow is in this way not computed as a previous step. Stein *et al.* [SMS00] compute vehicle ego-motion assuming a planar road with a direct method. The motion of the vehicle is restricted to 3 parameters, and vehicle ego-motion is computed through the definition of a global probability function.

Mandelbaum *et al.* [MSS99] avoid also the explicit computation of optical flow. Starting from an image flow model, the authors define a second-order (quadratic) model, which can be fitted into the correlation of the Laplacian-of-Gaussian images. The algorithm then iterates in a gradient descent style, estimating each component of motion and structure at a time.

Ego-motion estimation was also approached with omnidirectional cameras by Gluckman and Nayar [GN98] mapping velocity vectors to a sphere and testing the results with existing algorithms for ego-motion estimation. In this paper the transformation between planar image vectors to spherical vectors is achieved by means of a Jacobian matrix, which is specific of the omnidirectional system used. Vasallo *et al.* [VSVS02] solve this problem by proposing a general expression for the Jacobian that can be used for many different omnidirectional cameras.

Monocular methods based on normal flow were also proposed. Rives *et al.* [RBE87] present a structure from motion algorithm using normal flow and Kalman filters. Although ego-motion is obtained from inertial sensors, a refinement of the translational motion is achieved by minimizing a cost function relating inertial sensors and normal flow measurements.

7.2.2 Multi-ocular methods

7.2.2.1 Methods based on Stereo and Optical Flow

The first approaches to visual sensing for controlling the motion of a robot were made by Moravec [Mor80] in the late 1970's. The cart where the cameras where installed had a restricted motion which was estimated by stereo measurements and feature tracking. Some experiments and thoughts on visual navigation were also published by Matthies, Thorpe and Moravec *et al.* [MT84] [TMM85] and [LMT$^+$85].

Matthies and Shafer [MS87] present a method for the computation of ego-motion using a binocular platform, finding the six degrees of freedom of motion with a maximum likelihood estimator. In the paper it is assumed that the error in the projection of a world point on the image planes can be approximated as Gaussian. Instead of weighting the world points with scalars, the authors propagate the image covariance matrices to the interpolated 3D world point with a first order approximation. This allows to model world points with ellipsoidal covariances, i.e. the distribution error of the 3D points describe ellipsoids about the nominal mean that approximate the true error distribution. Kalman Filters are used in order to update the estimation of the landmarks. The update of the global robot position is carried out; concatenating the transformation matrices and estimating the uncertainty of the global position by propagating the covariances matrices of the incremental motions into a covariance of the global position.

Demirdjian and Horaud [DH00] compute ego-motion based on 3-D projective constraints (15 degrees of freedom). Camera calibration is not required since epipolar geometry is dynamically estimated together with the tracking algorithm. RANSAC is used in order to detect static points and the corresponding inliers/outliers with respect to a global error estimated over the whole sequence. This method uses a stereo tracking process throughout the sequence of image pairs. Interest points are selected with the Harris corner detector [HS88].

Mallet *et al.* [MLG00] propose an algorithm for the estimation of the 6 parameters of motion of a land-rover with stereo vision and feature tracking. The correlation technique for the point-to-point correspondence is the zero-mean normalized cross correlation. Evaluation of the confidence of the 3D point is carried out through the analysis of the correlation curve (the larger the correlation peaks, the better the 3D-estimate). Every point is auto-correlated (also with ZNCC) to detect the points which are the best traceable. For tracking, a search area for a point is centered in the predicted new position obtained with odometry (inertial sensors), and the size of the search area varies according to the uncertainty of the motion. Finally, the motion estimation is performed using the least-square estimation method based on SVD of Haralick *et al.* [HJL$^+$87].

Olson *et al.* [OMSM03] present an ego-motion estimation system to be used in a Mars land rover. The goal is to perform robust and accurate rover navigation autonomously over long distances in order to reach terrain landmarks with known location, but that are not within sight. The method used the maximum-likelihood formulation of motion estimation of Matthies and Shafer [MS87] that models error of landmark positions more accurately than scalar least-squares formulation. The authors evaluate the error produced when ego-motion is computed incrementally from frame to frame and find that the accumulated error grows super-linearly with the traveled distance. In order to achieve a linear growth of the error they proposed the use of an absolute orientation sensor (sun sensor, compass, etc.) Experiments with the optimal field-of-view conclude that the optimal FOV is approx. 35° when all other camera parameters remain constant. Stereo and tracking errors were also evaluated, with the conclusion that the navigation error varies much faster as the stereo error is varied compared to when the tracking error is varied. Errors of pitch and yaw angles are estimated with far away (background) objects using a large template window. With the estimation of pitch and yaw angles and the previous ego-motion estimate, the tracking procedure can be sped up, reducing the search area for landmarks. Four outlier rejection strategies are also described.

Jung and Lacroix [JL03] describe a method for computing ego-motion and 3D Maps from aerial stereo images. The method is based on the SLAM (Simultaneous Localization And Mapping [DNC$^+$01]) approach, where EKF is used to simultaneously estimate ego-motion and 3D landmark position of world points. Dense stereo is computed with a correlation based algorithm. Harris is used for selecting interest points. Landmarks are so detected and assigned to one of three possible categories: non-landmarks, which are used for computing ego-motion; new landmarks, which are cautiously added to the filter state and observed landmarks, which were already tracked over many images. Ego-motion is computed finding the optimal rotation and translation between the clouds of 3D points. An iterative procedure is used in order to successively eliminate pairs of points with large error.

Nistér *et al.* [NNB06] present a system for the computation of 6 d.o.f. of motion with binocular and monocular cameras. RANSAC is used for both monocular and binocular approaches. In the monocular method a 5 point algorithm is used for hypothesis generation, while in the stereo version of the algorithm a 3 point algorithm is used. In both cases 3D points are triangulated and a second triangulation step is performed after motion estimation in order to avoid a positive feedback between triangulation and motion estimation. The authors emphasize the fact of not computing the absolute orientation between frames using the 3D triangulated points. The main reason is that they could not find an efficient method for using the full 3D covariance matrices in the hypothesis generation step of RANSAC.

Talukder and Matthies [TM04] present a method for the detection of independently moving objects. Moving objects are found as outliers, computing first ego-motion, and then observing the error in the change of disparity and optical flow. No triangulation of feature points is required for the computation of ego-motion, since the least squares is expressed in terms of the optical flow expected given the rigid-body motion parameters. This method requires dense stereo and dense optical flow computation.

Demirdjian and Darrel [DD01] estimate 3D rigid motion estimation from binocular images without requiring explicitly computing 3D points. The authors demonstrate that the disparity-space image is a projective space and derive the equations relating disparity-spaces (which they call *d-motion*), as well as an error function to be minimized over the motion parameters. The computation of 3D points is, in this way, avoided and the error modeling is directly done in the image plane.

The same authors also present an extended version of the latter, which can handle multi-hypothesis [DD02]. The authors use Gaussian mixtures to model correspondence uncertainties for image velocity and disparity estimation. The rejection of outliers and the handling of multiple hypotheses are carried out with a random sampling process. Data samples are selected, and for each sample, "d-motion" [DD01] is estimated. Finally, the solution with the best criterion is selected.

Ho and Chung [HC97] adapt the factorization method of Tomasi and Kanade [TK92] to the stereo case in a very elegant way. The authors solve the correspondence problem in stereo, applying first the factorization method independently for left and right image sequences. From these results, initial stereo correspondences are established using the epipolar constraint and a basis for the rest of the 3D points is computed taking the first four columns of the U matrix. This basis is then used to recreate the list of 3D point of the right camera from the estimation of the left camera. Correspondences of columns are then found when the least-squares error is small enough.

Barron and Eagleson [BE95] compute ego-motion separately in the left and right images. Correspondences between left and right images are not needed but it requires a rotational motion and large baseline (otherwise only monocular parameters can be recovered). Kalman filters are used to integrate measurements.

Li and Duncan [LD93] present a method based on the binocular image flows of Waxman and Duncan [WD86] for the computation of translational parameters. 3D translational motion (without rotation) is obtained from linear equations of measured optical flow fields in the left and right images, without point-to-point correspondence between the left and right images. Additional equations are derived for the case of having binocular normal flow. Features correspondences between left and right images are then found using the computed translational motion parameters and the binocular flow information.

Sünderhauf *et al.* [SKLL05] review four methods for the estimation of ego-motion. The maximum likelihood method of [OMSM03] and [MS87] is shortly described. Two bundle adjustment methods are also presented. A comparison of the two bundle adjustment methods is carried out and experimental results are shown. The last method reviewed is a Kalman Filter based estimator.

Molton and Brady [MB00] estimate motion and structure of a rigid environment. Ego-motion is computed only with three points in three images and not in the Least Squares sense, which makes the method not appropriate for outdoor applications. Harris corner detector [HS88] is used to find corners and a matching strategy for stereo and tracking is presented.

Gehrig [Geh00] estimates the vehicle ego-position with Kalman Filters for a vehicle-following system. The system state includes position in the x-z plane, velocity, acceleration and an orientation parameter. The outputs provided by the speedometer, the steering angle sensor, 2D landmarks obtained with stereo as well as lane markings extracted from the image are fused together in the measurement equations of the Kalman Filter. An observability analysis is performed, leading to the minimal amount of landmarks required in order to make the system observable.

7.2.2.2 Methods based on Stereo and Normal Flow

Morency and Darrel [MD02] present a method to compute rigid object motion with stereo and normal flow. A hybrid error function which combines constraints from the Iterative Closest Point (ICP) algorithm [BM92] and Normal Flow Constraint (NFC) is presented. The algorithm first matches corresponding points between images using 4D points (3D Euclidean space and 1D for brightness). The normal flow constraint (NFC) is computed with an inverse calibration approach. Error functions for the ICP and NFC are both expressed as linear systems, which then are merged with a sigmoid function in order to compensate for the errors. Small movements are better estimated with the NFC and large movements are better estimated with the ICP algorithm. The experimental results are shown with a face tracker. Morency and Gupta [MG03] apply later the same hybrid technique for the computation of ego-motion.

Argyros *et al.* [AO97] [ATO98] present a method for the detection of independent 3D Motion using least median of squares (LMedS) and normal flow. The authors derive an equation starting from the known equations of relation between 3D velocity and its corresponding 2D velocity in the image, which is also applied in the stereo computation eliminating the motion parameters which do not take part in stereo due to the left-right configuration of the cameras (i.e. only rotation about the Y axis and translation in the X axis are considered). An expression for using LMedS is obtained, which is applied iteratively using the outliers found in previous iterations for detecting multiple motions. Ego-motion is supposed to be obtained in the first cycle (due to the assumption that at least 50% of the flow field corresponds to the static scene) and the rest of the motions detected are extracted from the static scene. Experimental results show a good detection of motion but no detail about the accuracy of the method and of the estimated ego-motion is given.

7.2.3 Fusion of Multiple Sensors for Ego-Motion and Positioning Estimation

Publications which deal not directly with visual odometry but review the state of the art in multiple sensor fusion for ego-motion and positioning estimation can be found in [JB96] and [Kel94].

SLAM methods (e.g. [DNC$^+$01]) are also alternative methods for computing ego-motion, but where the focus is given more to self-localization of the robot and therefore a map of the environment must be maintained. SLAM has also been initially performed using other ranging sensors, rather than regular cameras, although in the last years some new methods using visual odometry were proposed [JL03] [GS04] [SKLL05]. The method presented in this dissertation can be adapted for SLAM applications.

7.2.4 Summary of the Literature Review

A large number of publications on visual odometry estimation have been proposed in the last 20 years. The methods were categorized based on spatial and temporal characteristics, since it allows to classify almost any work based on this properties.

Most monocular techniques measure apparent shifts or velocities of gray-value structures and find the motion parameters which minimizes the difference between theoretical and measured motion field (e.g. , [BH81], [JH92], [Pra80], [Kan93]). The factorization method requires feature tracking, but works differently since it accumulates all coordinates of the feature over many frames in a measurement matrix and then obtains motion and structure by factoring with singular value decomposition or any other rank reduction technique (e.g. [TK92]). Another monocular techniques apply direct methods avoiding the computation of optical flow (e.g. [NH87], [SMS00]), or compute normal flow instead (e.g. [RBE87]).

Multi-ocular approaches can also be sub-categorized. Most stereo techniques obtain 3D points of the environment and compute the absolute orientation between the clouds of points. If inter-frame correspondences are given by optical flow or feature tracking, the computation of absolute orientation is direct (e.g. , [MS87] [MLG00] [OMSM03] [JL03], [SKLL05]). Otherwise, methods like the Iterative Closest Point algorithm are required for finding the correspondences, while solving the absolute orientation(e.g. [MD02]). Some other techniques do not compute 3D points as an intermediate step but relate projections of image points with motion parameters (e.g. [DD01]) requiring numerical optimization. Normal flow was also used for the computation of ego-motion from sequences of images using multiple-cameras (e.g. [AO97]). Methods which work almost independently in the left and right images and integrate stereo information at a later stage were proposed too (e.g. , [HC97] [BE95]).

The method presented next corresponds to the category of multi-ocular approaches with feature tracking. Feature tracking and stereo are required, as well, by other applications like obstacle detection, which run in parallel to the ego-motion estimation, so their availability is granted. Our solution has two main contributions. A fast and effective method for the rejection of outliers and the assignment of weights to the data points, and a method that reduces the accumulation of errors in the motion integration. In the following sections the algorithm is analyzed in detail.

7.3 Overview of the Algorithm

Figure 7.1 shows the block diagram of the method with a flow diagram of the algorithm. The algorithm works in an iterative way, computing the absolute orientation between the most recent cloud of 3D points and the cloud of points obtained at a previous time instant. The method works as a predictor/corrector algorithm integrating new measurements in each iteration. Each iteration carries out four main steps: 1) motion prediction; 2) Smoothness Motion Constraint application; 3) computation of absolute orientation (correction); and 4) integration of the new motion result to the current estimation. The motion prediction helps for the application of SMC. The SMC is a rejection rule applied to detect outliers (moving points and false correspondences) and to lessen the contribution of noisy measurements assigning weights to the data. The SMC reads the data points from the 3D Point List and generates two clouds of points. The correction of the previously predicted motion is accomplished by computing the absolute orientation between the selected clouds of points, as described in the Chapter 5. The fourth step is the integration of the newly acquired motion information to the current estimate. This step is performed by a motion decomposition followed by a interpolation of the motion parameters, as will be shown in Section 7.5.

The algorithm stops iterating when the amount of points available for the computation of absolute orientation falls below a threshold, or when a predefined maximal amount of iterations levels is achieved. As a result, the estimated motion is stored in the Motion Step List, which is used next by the Kalman Filter in order build the $\boldsymbol{A}$ and $\boldsymbol{b}$ matrices and update the filters, as shown in Chapter 4.

7.3.1 Motion Representation with Matrices

In order to simplify the notation of the following subsections, we represent the motion in homogeneous coordinates. The computed motion of the camera between two consecutive frames, i.e. from frame $k-1$ to frame k, is represented by the matrix $\boldsymbol{M}'_k$ where:

$$\boldsymbol{M}'_k = \begin{bmatrix} \boldsymbol{R}_{C_k} & \boldsymbol{t}_{C_k} \\ 0 & 1 \end{bmatrix} \tag{7.1}$$

The rotation matrix $\boldsymbol{R}_k$ and translation vector $\boldsymbol{t}_k$ from Equations 4.25 to 4.20 are obtained by inverting $\boldsymbol{M}'_k$, i.e. :

$$\boldsymbol{M}'^{-1}_k = \begin{bmatrix} \boldsymbol{R}_k & \boldsymbol{t}_k \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^T_{C_k} & -\boldsymbol{R}^T_{C_k}\boldsymbol{t}_{C_k} \\ 0 & 1 \end{bmatrix} \tag{7.2}$$

The total motion of the camera since initialization can be obtained as the products of the individual motion matrices:

$$\boldsymbol{M}_k = \prod_{i=1}^{k} \boldsymbol{M}'_i \tag{7.3}$$

Observe that total motion matrices are written without any accent symbol while one step motion matrices are denoted with an apostrophe. A sub-chain of movements from time t_n to time t_m is:

$$\boldsymbol{M}_{n,m} = \boldsymbol{M}_n^{-1}\boldsymbol{M}_m = \prod_{i=n+1}^{m} \boldsymbol{M}'_i \tag{7.4}$$

and per definition

$$\boldsymbol{M}_{k-1,k} = \boldsymbol{M}'_k \ \text{ and } \ \boldsymbol{M}_{k,k} = \boldsymbol{I}_{4\times4} \tag{7.5}$$

The rotation and translation of of a 3D point vector $\boldsymbol{p}$ into a new vector $\boldsymbol{q}$ with the

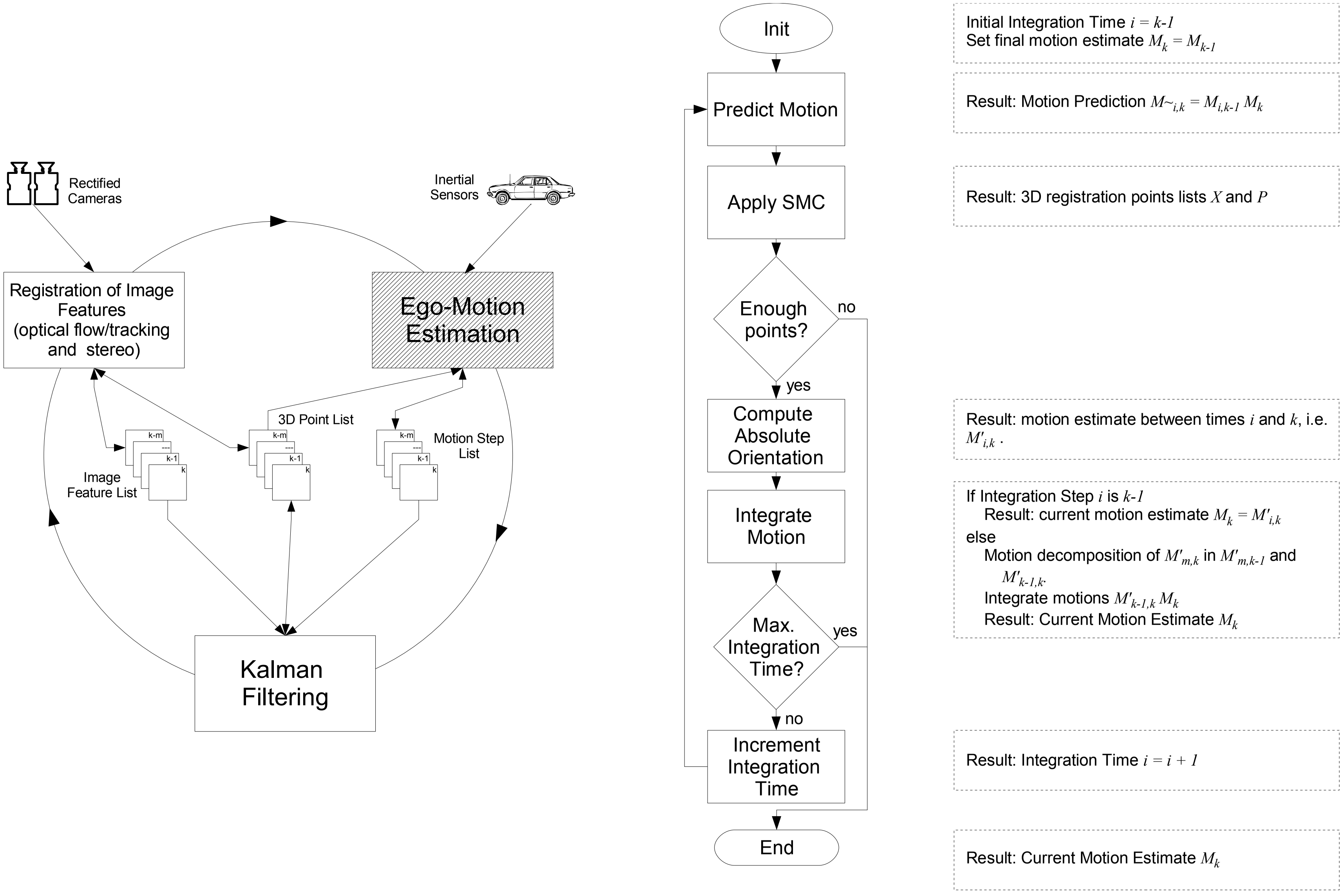

Initial Integration Time $i = k\text{-}1$
Set final motion estimate $M_k = M_{k-1}$
Result: Motion Prediction $M\sim_{i,k} = M_{i,k-1}\, M_k$
Result: 3D registration points lists X and P
Result: motion estimate between times i and k, i.e. $M'_{i,k}$.
If Integration Step i is $k\text{-}1$
Result: current motion estimate $M_k = M'_{i,k}$
else
Motion decomposition of $M'_{m,k}$ in $M'_{m,k-1}$ and $M'_{k-1,k}$.
Integrate motions $M'_{k-1,k}\, M_k$
Result: Current Motion Estimate M_k
Result: Integration Time $i = i + 1$
Result: Current Motion Estimate M_k
Init
Predict Motion
Apply SMC
Enough points?
no
yes
Compute Absolute Orientation
Integrate Motion
Max. Integration Time?
yes
no
Increment Integration Time
End
Rectified Cameras
Inertial Sensors
Registration of Image Features (optical flow/tracking and stereo)
Ego-Motion Estimation
3D Point List
Motion Step List
Image Feature List
k-m
---
k-1
k
Kalman Filtering

Figure 7.1: Block Diagram of the Approach

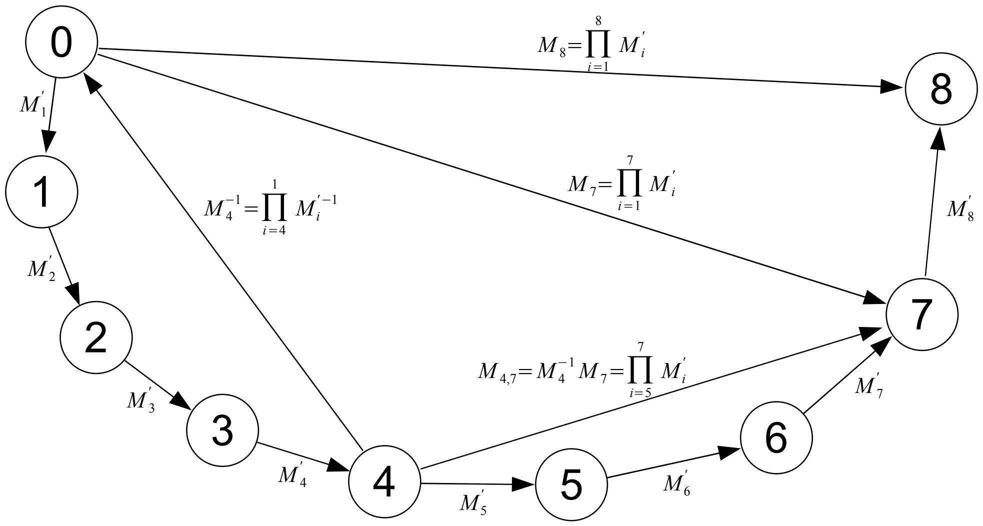

0
1
2
3
4
5
6
7
8
M_1'
M_2'
M_3'
M_4'
M_5'
M_6'
M_7'
M_8'
$M_8=\prod_{i=1}^{8} M_i'$
$M_7=\prod_{i=1}^{7} M_i'$
$M_4^{-1}=\prod_{i=4}^{1} M_i'^{-1}$
$M_{4,7}=M_4^{-1}M_7=\prod_{i=5}^{7} M_i'$

Figure 7.2: The integration of single-step estimates can be obtained by multiplying the individual motion matrices. Every circle denotes the state (position and orientation) of the camera between time t_0 and time t_8. Vectors indicate motion in 3D-space.

motion matrix $\boldsymbol{M}_k'$ is performed as

$$\dot{}\boldsymbol{q} = \boldsymbol{M}_k' \, \dot{}\boldsymbol{p} \tag{7.6}$$

where $\dot{}\boldsymbol{p}$ is the homogeneous version of vector $\boldsymbol{p}$.

Figure 7.2 shows an example of motion integration with matrices. As we will show later in Section 7.5, Equation 7.4 will support the integration of the motion between two non-consecutive frames (multi-frame estimation).

7.4 Smoothness Motion Constraint

7.4.1 Introduction

The purpose of weighting the data in a least squares approach is to allow the integration of measurements with different reliability, increasing the amount of data that can be used, and reducing the uncertainty (covariance matrix) of the estimate. Weighting offers the modeling of the *extrinsic noise*, which is produced mainly due to the measurement process of the sensor. On the other hand there is an additional source of noise, which does not depend on the sensor but on the data correspondence. The *intrinsic noise* has the most dominant effect in correspondence problems and is mainly due, not only to incorrect matches, but also due to correct matches which cannot be described by the system being modeled [EBW04], [HJL$^+$89], [AO97], [ATO98]. Incorrect matches originate in the stereo and feature tracking algorithms, and are caused by the temporal (optical flow) or spatial (stereo) incorrect matching of points between views. Intrinsic noise in the form of *outliers* can also be obtained when some assumptions about the measurements are violated. This is the case, for example, when a point assumed to be static belongs to an independently moving object. Thus, the appropriate modeling of extrinsic noise and the opportunely detection of outliers is of paramount importance when a robust registration algorithm is desired.

A variety of rejection strategies and weighting methods has been proposed. Jung and Lacroix [JL03] and Mallet *et al.* [MLG00] detect outliers by computing a 3D transformation between triplets of points and rejecting the triplets with error larger than k times the mean of the residual error. The residual error is a measurement of the goodness of the fit of the rigid body motion between the 3D points triplets. This procedure is applied iteratively starting with a large k and reducing it after each iteration until $k = 3$. Hirschmüller [HIG02] *et al.* analyze the consistency of distance and rotation between subsets of points in the first and second cloud, taking into account the stereo error distribution. Bandari *et al.* [BNB04] make use of a similar criterion based on spectral clustering. The authors assume independent identically distributed noise and make use of the rigidity constraint, i.e., the distance between two 3D points of a rigid body motion is maintained. Estépar *et al.* [EBW04] estimate the absolute orientation at each iteration and reassign weights according to the residual values. The weights are used to add an isotropic part to the covariance matrices of corresponding points. Haralick *et al.* [HJL$^+$89] apply M estimators. Ohta and Kanatani [OK98] use the renormalization method.

With the exception of the methods of Bandari *et al.* and Hirschmüller, which only detect outliers, all the methods listed above imply the iterative refinement of the motion parameters by the successive elimination of outliers and re-weighting of point correspondences according to some criterion. This section presents a method for the detection of outliers and the automatic assignment of weights to the measurements in a single step. We call this method Smoothness Motion Constraint (SMC). In the following two subsections, the SMC for the weighted least squares (Section 5.4) as well as for the total least squares (Section 5.5) is formulated. The effectiveness of this method is shown in this chapter with simulated data, and in the next chapter with real data.

7.4.2 SMC for Weighted Least Squares

Dealing with 3D points obtained through stereo triangulation means dealing with anisotropic noise as was shown in Chapter 6. The first problem that arises when modeling anisotropic noise with weighting scalars is: which is the correct weighting scalar for each measurement? In other words, what value must have each w_i of Equation 5.17. Weights should be large for data points with small noise and vice versa [HJL$^+$89] [Kan94]. Since both $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are noisy measurements, the weight must reflect the joint reliability of the vector $\boldsymbol{v}_i = \overrightarrow{\boldsymbol{x}_i \boldsymbol{p}_i}$. A reasonable choice is the inverse mean square error of possible errors $\boldsymbol{\Gamma}_{x,i}$ and $\boldsymbol{\Gamma}_{p,i}$ [Kan94], i.e.

$$w_i = \frac{1}{\mathrm{tr}(\tilde{\boldsymbol{R}} \boldsymbol{\Gamma}_{x,i} \tilde{\boldsymbol{R}}^T) + tr(\boldsymbol{\Gamma}_{p,i})}$$

The covariance matrices $\boldsymbol{\Gamma}_{x,i}$ and $\boldsymbol{\Gamma}_{p,i}$ are obtained from Equation 6.6. The rotation matrix $\tilde{\boldsymbol{R}}$ should be an approximation to the sought rotation matrix $\boldsymbol{R}$. For small rotations $\tilde{\boldsymbol{R}}\tilde{\boldsymbol{R}} \approx \boldsymbol{I}_{3\times 3}$ and observing that $\mathrm{tr}(\tilde{\boldsymbol{R}}\boldsymbol{\Gamma}_{x,i}\tilde{\boldsymbol{R}}^T) = \mathrm{tr}(\tilde{\boldsymbol{R}}\tilde{\boldsymbol{R}}\boldsymbol{\Gamma}_{x,i}) \approx \mathrm{tr}(\boldsymbol{\Gamma}_{x,i})$ Equation 7.7 becomes

$$w_i = \frac{1}{\mathrm{tr}(\boldsymbol{\Gamma}_{x,i}) + \mathrm{tr}(\boldsymbol{\Gamma}_{p,i})}. \tag{7.7}$$

Since the main error component is given in the viewing direction, the weights obtained with Equation 7.7 will mainly reflect the uncertainty due to distance. Nevertheless, weighting with Equation 7.7 has two main drawbacks. First of all, it does not account for outliers. Second, in order to model the different accuracy of each measured point, an individual estimation of the feature variance $(\sigma_u^2, \sigma_v^2, \sigma_d^2)$ is required. We now proposed a weight function, which is able to cope with both problems simultaneously.

If the frame rate is high enough to obtain a smooth motion between consecutive frames, then the motion observed at any moment is similar to the immediate previous motion. Therefore, before including the pair of points $\boldsymbol{p}_i$ and $\boldsymbol{x}_i$ into their corresponding data sets $\mathcal{P}$ and $\mathcal{X}$, we evaluate if the vector $\boldsymbol{v}_i = \overrightarrow{\boldsymbol{x}_i\boldsymbol{p}_i}$ indicates a coherent movement. Based on our previous estimation of motion $\boldsymbol{M}'_{k-1}$ at time t_{k-1} we evaluate the motion coherence of the vector $\boldsymbol{v}_i$ as:

$${}^{\cdot}\boldsymbol{c}_i = \boldsymbol{M}'_{k-1}\,{}^{\cdot}\boldsymbol{x}_i - {}^{\cdot}\boldsymbol{p}_i \tag{7.8}$$

i.e. the error of the point position with respect to our prediction. Let us define d_W as the maximal accepted error of the position of 3D point with respect to a predicted position. The proposed weight is defined as

$$w_i = \begin{cases} \dfrac{1}{\|\boldsymbol{c}_i\|} & \text{, if } \|\boldsymbol{c}_i\| < d_W \\ \\ 0 & \text{, otherwise} \end{cases} \tag{7.9}$$

that is, the inverse error magnitude, if the error is smaller than the predefined maximal threshold d_W. If this is not the case, then the vector $\boldsymbol{v}_i$ is identified as an outlier and not included for the computation of absolute orientation. Equations 7.8 and 7.9 define the Smoothness Motion Constraint for the scalar case.

7.4.3 SMC for Total Least Squares

The scalar version of the SMC defines the contribution of the pair of measurements, $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$, to the least squares estimation. The TLS version, instead, defines the matrices $\boldsymbol{\Gamma}_{x,i}$ and $\boldsymbol{\Gamma}_{p,i}$ to be used in Equation 5.13. The current points $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are actually estimates of 3D position of the measurements $\boldsymbol{x'}_i = (u_{x,i}, v_{x,i}, d_{x,i})^T$ and $\boldsymbol{p'}_i = (u_{p,i}, v_{p,i}, d_{p,i})^T$ respectively. Given $\boldsymbol{p}_i$ and camera motion prediction $\boldsymbol{M}'_{k-1}$, prediction of point $\boldsymbol{x'}_i$ is defined as

$${}^{\cdot}\tilde{\boldsymbol{z}}_{x',i} = {}^{\cdot}\boldsymbol{h}(\boldsymbol{M}'_{k-1}\,{}^{\cdot}\boldsymbol{p}_i) \tag{7.10}$$

where the function ${}^{\bullet}\boldsymbol{h}$ is the homogeneous version of the projection Equation 4.31. Equivalently, given ${}^{\bullet}\boldsymbol{x}_i$ and the prediction of inverse camera motion $\boldsymbol{M}_{k-1}^{\prime -1}$, the prediction of the feature point ${}^{\bullet}\boldsymbol{p'}_i$ is

$$
{}^{\bullet}\tilde{\boldsymbol{z}}_{p',i} = {}^{\bullet}\boldsymbol{h}(\boldsymbol{M}_{k-1}^{\prime -1}\, {}^{\bullet}\boldsymbol{x}_i) \tag{7.11}
$$

Let us define the error in the prediction of point projection on the image plane as

$$
\boldsymbol{e}_{x',i} = \boldsymbol{x'}_i - \tilde{\boldsymbol{z}}_{x',i} \quad \text{and} \quad \boldsymbol{e}_{p',i} = \boldsymbol{p'}_i - \tilde{\boldsymbol{z}}_{p',i}. \tag{7.12}
$$

Let us define d_I as the maximal accepted error (in pixels) for the projection of a world point on the image plane. If $\boldsymbol{e}_{p',i} > d_I$ or $\boldsymbol{e}_{x',i} > d_I$ the vector $\boldsymbol{v}_i = \overrightarrow{\boldsymbol{x}_i\boldsymbol{p}_i}$ is identified as an outlier. Otherwise, the weight matrices for point $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are computed as

$$
\boldsymbol{\Gamma}_{x,i} = \boldsymbol{G}_{x',i}\boldsymbol{E}_{x',i}\boldsymbol{G}_{x',i}^T \quad \text{and} \quad \boldsymbol{\Gamma}_{p,i} = \boldsymbol{G}_{p',i}\boldsymbol{E}_{p',i}\boldsymbol{G}_{p',i}^T \tag{7.13}
$$

where $\boldsymbol{G}_{\check{r},i}$ is the Jacobian of Equation 6.1 for a feature point $\check{\boldsymbol{r}}_i$, as already derived in the previous chapter; and $\boldsymbol{E}_{x',i}$ and $\boldsymbol{E}_{p',i}$ are diagonal matrices such that

$$
\boldsymbol{E}_{x',i,jk} = \delta_{jk}\sqrt{\boldsymbol{e}_{x',i,j}} \quad \text{and} \quad \boldsymbol{E}_{p',i,jk} = \delta_{jk}\sqrt{\boldsymbol{e}_{p',i,j}} \tag{7.14}
$$

where δ_{jk} is the Kronecker delta.

7.4.4 Discussion of the Scalar and Matrix SMC

When applying the scalar version of the SMC, the outliers are forced to have small or zero weights, reducing their contribution in the least squares solution. Also, points with small noise will have, on an average, larger weights than those with larger noise error. In general, the weights must be chosen such that the measurement $\boldsymbol{v}_i = \overrightarrow{\boldsymbol{x}_i\boldsymbol{p}_i}$ contributes positively to the solution. This might lead to the case where a point is assigned a very small weight because its depth error is too large, although the lateral and height position were measured with high accuracy. This is the case of feature points located around the principal point of the image: We know certainly that the corresponding 3D point is straight ahead, but not exactly at which depth. TLS takes advantage over the WLS, since it allows to model the contribution of each individual measurement.

The accuracy of tracking a feature point or computing the disparity normally depends on multiple factors. The tracking and stereo algorithm benefits from high textured image regions. In [JL03] it is shown that the larger the magnitude of the correlation peak at its minimum, the higher the accuracy of the measured stereo disparity. It has also been shown [OMSM03] that there is a positive correlation between the time a feature point can be tracked and the stability and accuracy of its corresponding measurement. We would like to detect cases like these and assign larger weights for these measurements. With the SMC such detection is not required, since a highly accurate measurement will have - on an average - smaller prediction error than noisy measurements and, therefore, a higher weight.

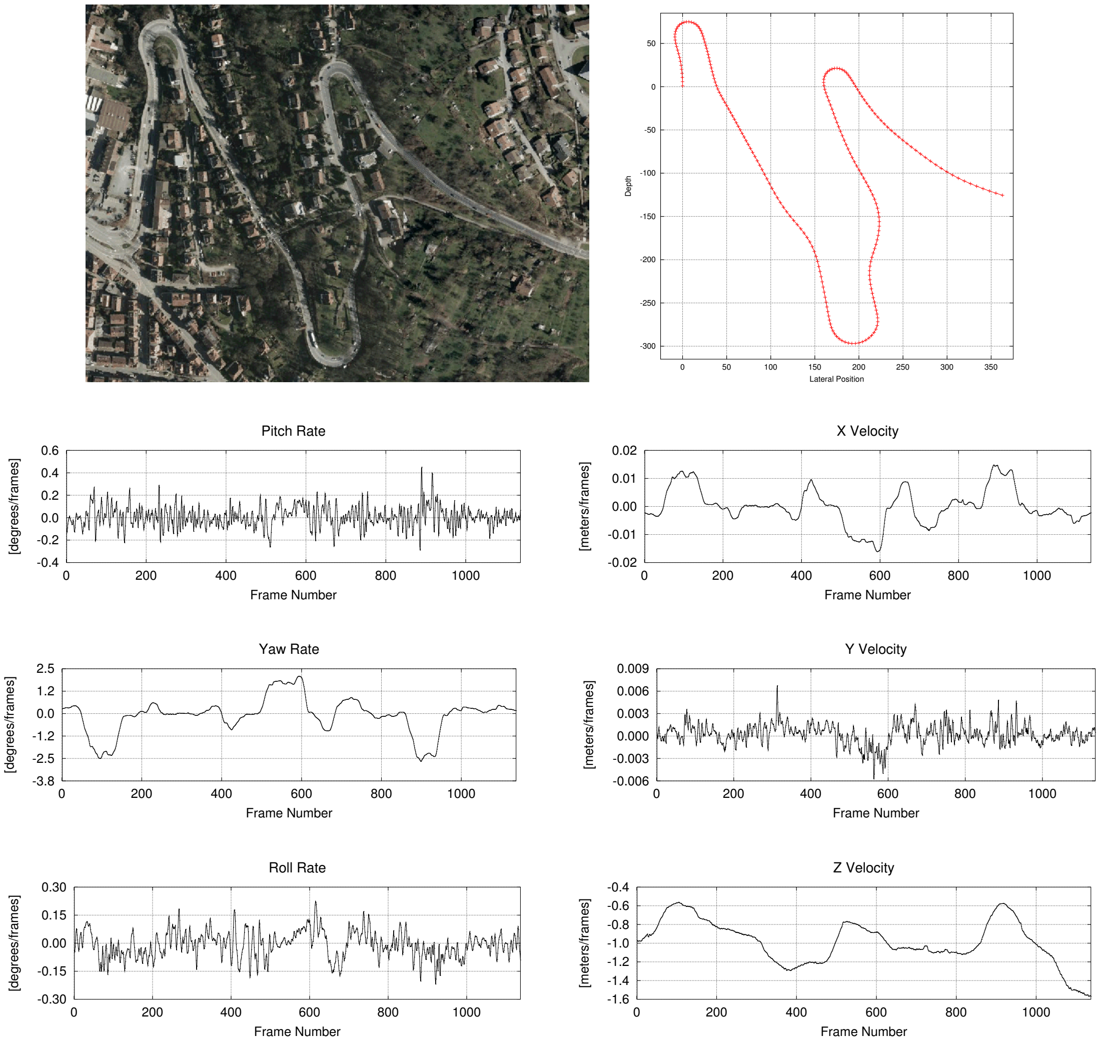


Figure 7.3: Simulated motion sequence

A very important feature of both versions of the SMC is also the benefit obtained from the integration of temporal data. Many methods ignore valuable previous information and just compute the best fit by considering only the instantaneous measured data. It is only possible to obtain a meaningful result if at most 50% of the data are outliers, i.e. such estimators have a maximum breakdown point of 50%. Nevertheless, if previous information is integrated in the estimation process, the breakdown point of robust estimators will not depend on the percentage of contaminated data, but more on the number of non-contaminated data, allowing breakdown points larger than 50%. The simulation results in Section 7.4.6 show this behavior.

7.4.5 Generation of Simulated Data

In order to show the effectiveness of the SMC and compare it with the ordinary Least Squares weighting, a simulated motion sequence is used. The motion sequence was generated by computing the ego-motion of the cameras installed on a vehicle while the vehicle was traveling the path shown at the top-left of Figure 7.3. An optimal parameter set in sights of accuracy (and at the expense of real-time) was choosen in this step for the computation of ego-motion. The frame rate was 10 frames/second and the distance traveled was approximately of 1 km with a total of 1137 stereo images. The motion parameters were smoothed in order to reduce noise and the pitch and roll rate as well as the vertical translation were amplified by a factor of 1.33, in order to obtain a slightly rougher motion than normally expected in real traffic situations. Figure 7.3 shows plots of the motion sequence.

The image features and corresponding 3D points of the environment used as measurements in the Least Squares approach are generated according to the following procedure:

1. the feature image position $(\bar{u}_{p_i}, \bar{v}_{p_i})$ is randomly generated, equally distributed in the left image;
2. the depth $\bar{Z}_{p_i}$ of the corresponding 3D point is randomly generated with equal distribution between minimal and maximal distance $(dist_{min}, dist_{max})$;
3. the disparity $\bar{d}_{p_i}$ of the feature and the corresponding lateral and height position $(\bar{X}_{p_i}, \bar{Y}_{p_i})$ of the 3D point are obtained from the data generated in the two previous points. At this point the world point position with coordinate $\bar{\boldsymbol{p}}_i = (\bar{X}_{p_i}, \bar{Y}_{p_i}, \bar{Z}_{p_i})^T$ and its projection $\bar{\boldsymbol{p}}'_i = (\bar{u}_{p_i}, \bar{v}_{p_i}, \bar{d}_{p_i})^T$ are already generated;
4. noise is added to $\bar{\boldsymbol{p}}'_i$ in order to obtain $\boldsymbol{p}'_i = (u_{p_i}, v_{p_i}, d_{p_i})^T$;
5. the image feature point $\boldsymbol{p}'_i$ is triangulated in order to obtain $\boldsymbol{p}_i = (X_{p_i}, Y_{p_i}, Z_{p_i})^T$, i.e. the noisy version of $\bar{\boldsymbol{p}}_i$;
6. the current world point position is obtained by $\bar{\boldsymbol{x}}_i = \bar{\boldsymbol{R}}_{\boldsymbol{k}}\bar{\boldsymbol{p}}_i + \bar{\boldsymbol{t}}_{\boldsymbol{k}}$ where $\bar{\boldsymbol{R}}_{\boldsymbol{k}}$ and $\bar{\boldsymbol{t}}_{\boldsymbol{k}}$ are the inverse rotation matrix and translation vector of the camera obtained from the motion sequence at step k;
7. the point $\bar{\boldsymbol{x}}_i$ is projected on the image obtaining $\bar{\boldsymbol{x}}'_i = (\bar{u}_{x_i}, \bar{v}_{x_i}, \bar{d}_{x_i})^T$;
8. noise is added to $\bar{\boldsymbol{x}}'_i$ in order to obtain $\boldsymbol{x}'_i = (u_{x_i}, v_{x_i}, d_{x_i})^T$;
9. (when required) equally distributed noise in the range $(-m, m)$ pixels is added to every component of $\boldsymbol{x}'_i$ generating a potential outlier; and finally
10. $\boldsymbol{x}'_i$ is triangulated in order to obtain $\boldsymbol{x}_i = (X_{x_i}, Y_{x_i}, Z_{x_i})^T$, i.e. the noisy version of $\bar{\boldsymbol{x}}_i$.

The above procedure ensures uniform distribution in the position of the feature in the image plane and in the depth of the corresponding 3D world point. The noise of steps 4 and 8 has a Gaussian or Slash distribution according to the evaluation desired[1]. A Slash distributed random variable can be obtained as a Gaussian random

---

[1] When testing robust estimators Slash distribution are preferred to Cauchy distribution, because of the smaller peakedness at the origin [Dav02].

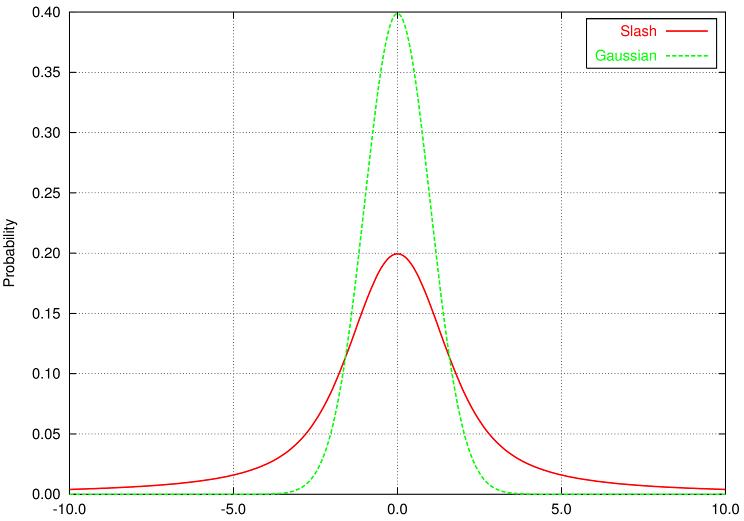


Figure 7.4: Comparison of Slash and Gaussian p.d.f. assuming a mean of 0 and a variance of 1.

variable with mean 0 and variance σ_I^2 divided by a uniform random variable over the unit interval $[0, 1]$. The Slash distribution has larger tails than the Gaussian distribution, as Figure 7.4 shows. In either case, a zero-mean multi-variate noise term $\boldsymbol{\eta}$ is added to feature point vectors $\bar{\boldsymbol{x}}'_i$ and $\bar{\boldsymbol{p}}'_i$ and not to the world points as is usually done ([HJL$^+$89]). 3D points are obtained with the noisy feature positions, generating the real error distribution expected in the triangulated points. If, as a consequence of noise or motion, an image feature results with a position outside the image coordinates or its corresponding 3D point depth lies outside the range $(dist_{min}, dist_{max})$, the point is discarded. This process is repeated until N valid points are obtained.

In order to evaluate the performance of the ego-motion algorithm the Root-Mean-Square Error (RMSE) of the rotation angle and RMSE of the translation distance are used. The error in the translation distance is obtained as the Euclidean distance between real and estimated translation, i.e. if $\bar{\boldsymbol{t}}_E = \|\boldsymbol{t} - \bar{\boldsymbol{t}}\|$ is the error in translation, the translation distance error is $\|\bar{\boldsymbol{t}}_E\|$. Since rotations are elements of the group $SO(3)$, a rotation error cannot be obtained just as the Euclidean distance of rotation parameters. The error measure used for the evaluation of the rotation error is the minimal angle of rotation needed, around some axis, to transform estimated rotation to true rotation [Kan94] [OK98] [EBW04]. If $\boldsymbol{R}$ is an estimation of the true rotation matrix $\bar{\boldsymbol{R}}$, then the error rotation matrix is $\bar{\boldsymbol{R}}_E = \boldsymbol{R}\bar{\boldsymbol{R}}^T$. The rotation expressed in $\bar{\boldsymbol{R}}_E$ is represented as the angle of rotation $\overline{\Delta\Omega}$ around some axis $\bar{\boldsymbol{l}}_r$; $\overline{\Delta\Omega}$ is the error rotation angle.

7.4.6 Simulation Results

This section shows some experimental results using the simulated data of the previous section. For all the tests; the minimal distance is $dist_{min} = 3.63125$ meters, which corresponds to a disparity of 80, pixels and the maximal distance $dist_{max} = 141.62$, which corresponds to a disparity of 2 pixels. The camera parameters are $f_u = f_v = 830$, $u_0 = 320$, $v_0 = 240$, $B = 0.35$ meters with an image size of 640 pixels width and 480 pixels height. The number of points N is 500 (when not stated otherwise).

The improvements achieved with the scalar and matrix versions of the SMC are shown by comparison with the standard weighting methods. When using the WLS approach, the results of the SMC defined in Equation 7.9 (labeled "WLS SMC" in the graphs) is compared with the results using the standard weighting expressed in Equation 7.7 (labeled "WLS Without SMC" in the graphs)[2]. For the TLS approach, the results of the SMC of Equation 7.13 (labeled "TLS SMC" in the graphs) is compared with the results using the standard covariance matrices obtained with Equation 6.4 for points $\boldsymbol{p}$ and $\boldsymbol{x}$ (labeled "TLS Without SMC" in the graphs). Each point of the following plots were obtained running 10 times the whole motion sequence, and therefore the RMSE corresponds to $11,370$ motion estimation steps.

Figures 7.5(a) and 7.5(b) show the performance of the methods under varying Gaussian image position noise. Both, for the rotational and translational plots, the SMC curves outperform the standard versions. As expected the TLS approach without SMC performs better than the WLS approach, since the covariance matrices allow the modeling of the direction and size of the error for each point independently, as already shown in Chapter 6. Nevertheless, comparing both SMC curves, the scalar LS performs better than the matrix weighted LS. This is an effect of the penalization imposed by the closed-form solution: rotation is found by projecting the optimal unconstrained solution into the parameter manifold, as already discussed in the Section 5.5 of Chapter 5. The real benefit of the TLS is given in the translational component, which is at least 3 times more accurate than the WLS approach.

The penalization imposed in the rotation matrix for the closed form solution of the TLS approach is small and when Gaussian noise is used instead of Slash noise, the TLS with SMC shows the more robust behavior as shown in Figures 7.6(a) and 7.6(b). Observe that under Slash noise the scalar version of the SMC is the least affected in the translation estimation, since the error increases by a factor smaller than 1.3.

The estimation accuracy with respect to the number of points in the sets behaves as expected, i.e. the accuracy becomes considerably worse with decreasing number of points [HJL$^+$89]. Figure 7.7 and 7.8 show RMSE of rotation and translation respectively, for varying number of points and fixed Gaussian noise with $\sigma_I = 0.4$. The observations made for Figure 7.6 are also true for these graphs, with an exception. The TLS SMC version shows a better performance for rotation estimation than the WLS with SMC, when the number of points decreases, demonstrating once again a more stable solution. In Figure 7.7(a) this can be seen when the number of points is less than 65.

Figure 7.9 shows the results when step 9 of the procedure of the previous section is used for an increasing percentage of features. As with the previous evaluation, a zero-mean Gaussian noise term η with $\sigma_I^2 = 0.4^2$ is added to the features points. The percentage of points, for which step 9 of the procedure is applied, is varied between 0 and 90. For this percentage of features, a uniform random value κ in the interval $[-32, 32]$ (i.e. $\pm 5\%$ of the image width) is added to the image vector generating potential outliers, and simulating correspondence errors. Figure 7.9(a) shows the rotational error as a function of the percentage of outliers generated in this way, while

---

[2] If Equation 7.7 is used instead of Equation 6.2, the same results are obtained.

Figure 7.9(b) shows the translational error. In both cases, the estimation remains quite accurate even when more than 50% are actually false correspondences.

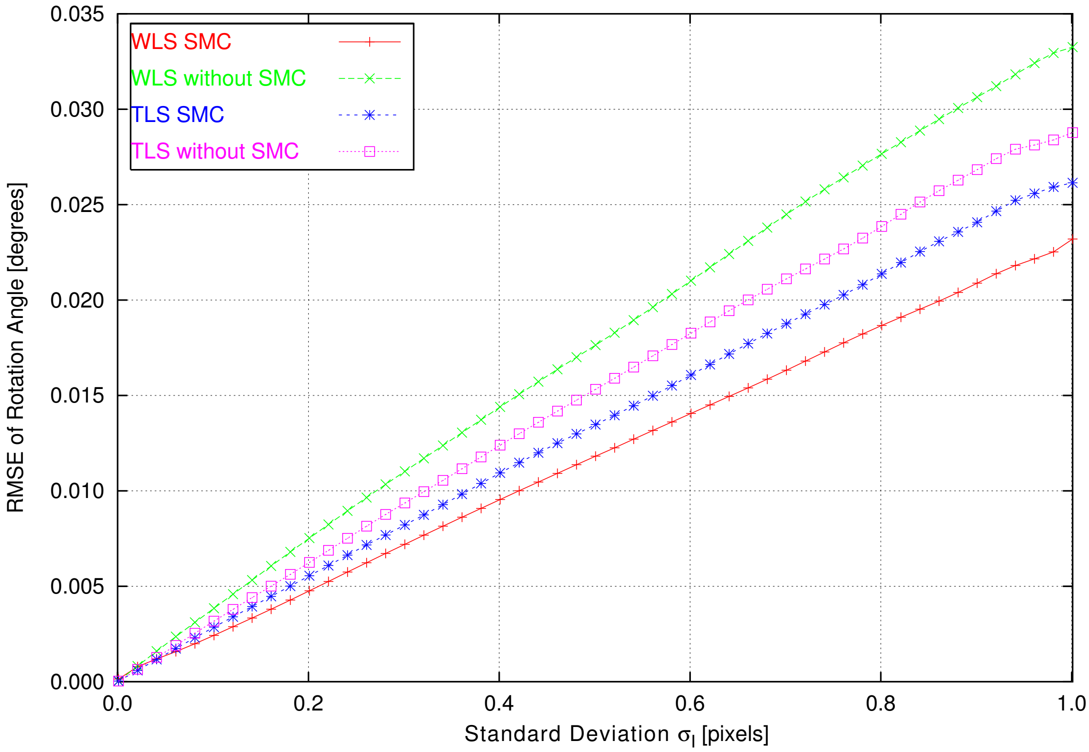
Figure 7.10 shows an example of the motion results obtained with SMC TLS version when adding Gaussian noise with standard deviation 0.4 pixels and using 500 pairs of points.

7.5 Multi-Frame Estimation

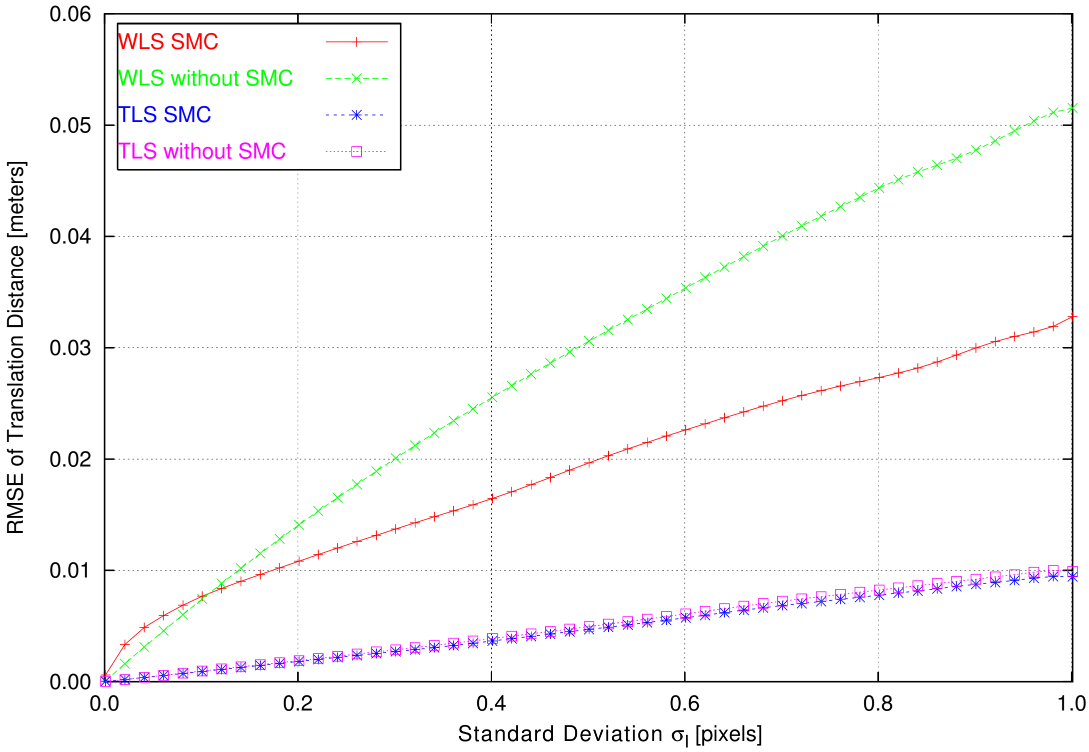
7.5.1 Introduction

Two-Frame estimation, i.e. the estimation of the motion parameters from the current and previous frames is the standard case in the literature. When the motion parameters are filtered over time, for instance with Kalman Filters ([MS87] [OMSM03] [JL03] [SKLL05] [BE95]), the measurement of the process still continues to be the clouds of points of the last two available frames. The main problem with this approach is the large error accumulation. Olson *et al.* [OMSM03] have found that the error in the ego-position grows super-linearly over time. The overall error has two dependent components, which are orientation and translation. The variance of the integrated orientation is found as a term that is proportional to the sum of the variances of all estimation steps. Instead, the ego-position variance increases as a sum of two terms: i) a term which is proportional to the sum of the translation variances of each estimation step; and ii) a term which increases with the integral orientation error. Rotation errors eventually turn into translation errors, because of the dependency of translation and rotation. It is therefore expected an asymptotic growth in the translation component of $t_E = O(n_f^{3/2})$ where n_f is the number of integrated frames. Olson *et al.* [OMSM03] reduce the error function to a linear term in the number of frames by making periodic corrections of the orientation using an absolute orientation sensor.

Figure 7.11(a) exemplifies the ego-position error with two curves. A pure translational motion in depth was simulated with random uniform Z velocity, at two different frame rates. The absolute orientation was computed at every frame and the total motion was integrated. At each frame the Euclidean distance error between estimated and true position was computed and plotted against frame number. Figure 7.11(a) show the super-linearly increment for both curves. Observe that with a larger frame-rate, the estimation error per frame is smaller. This is because the motion uncertainty with a higher frame rate is smaller, due to the shorter distance traveled between frames. When working with real images, a higher frame rate means less false correspondences. The correspondence problem between frames (tracking) is easier to solve because the search space in the image area is reduced. Nevertheless, a higher frame rate means also the integration of more errors in the same time interval. Figures 7.11(b) shows the same curves as in Figure 7.11(b) but showing the position error as a function of distance traveled (or time). At the same traveled distance (or time) the curve of the smaller frame rate shows less accumulated error. A low frame rate is preferable, because fewer errors are accumulated in the same distance (or time) as with a higher frame rate. But large distances between frames means less feature correspondences (because more features disappear from the field

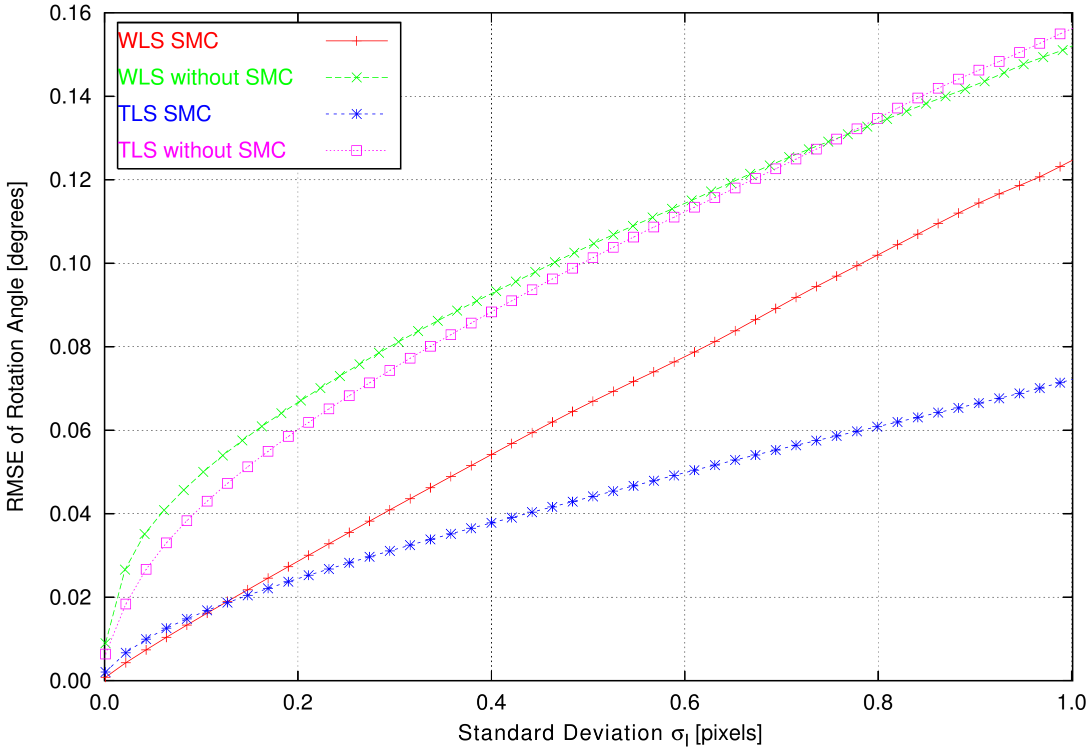


(a) RMSE of rotation angle for varying Gaussian image position noise.

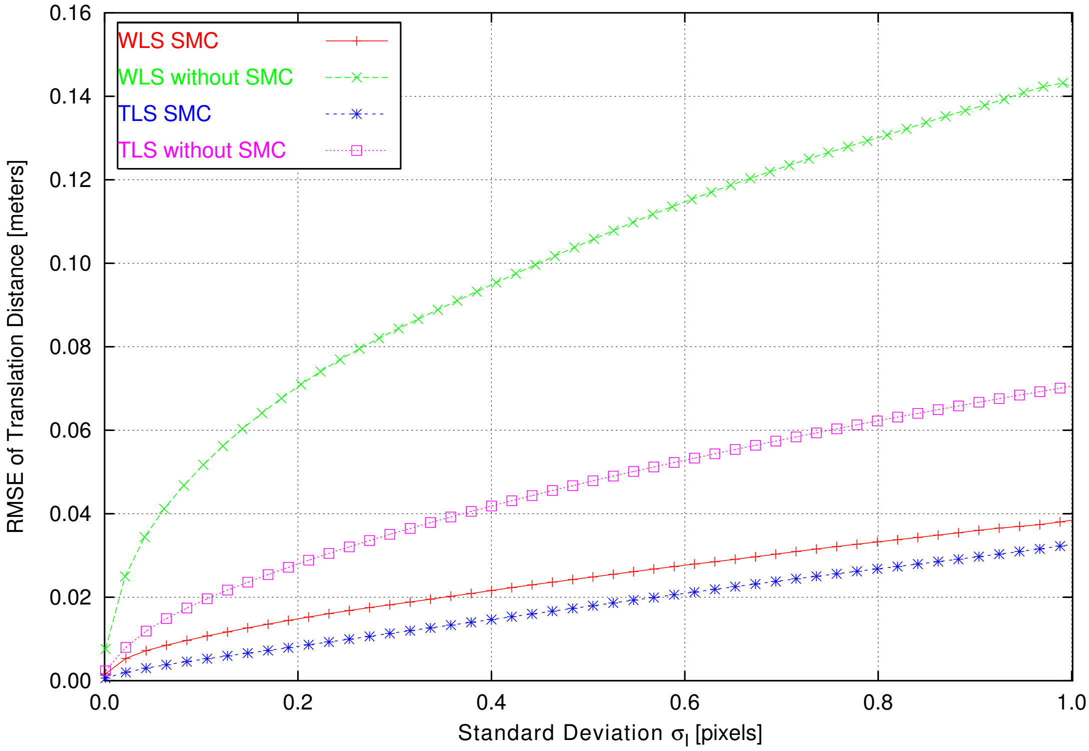


(b) RMSE of translation distance for varying Gaussian image position noise.

Figure 7.5: Rotational and Translational accuracy for varying Gaussian image position noise.

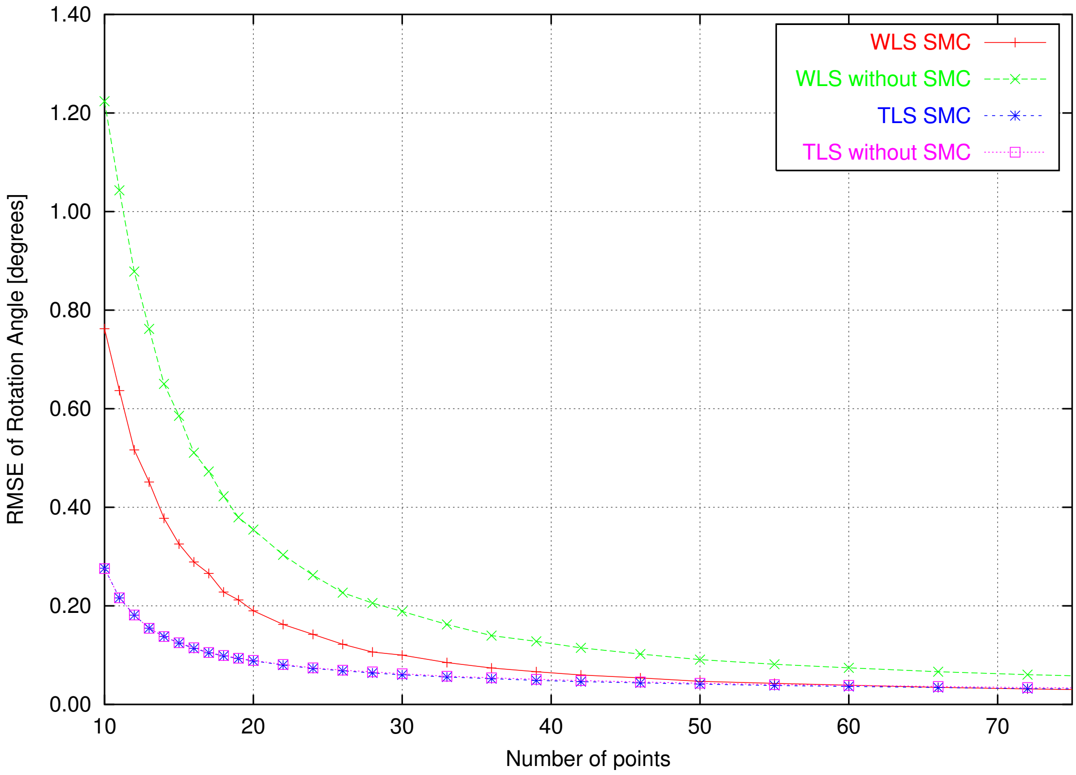


(a) RMSE of rotation angle for varying Slash image position noise.

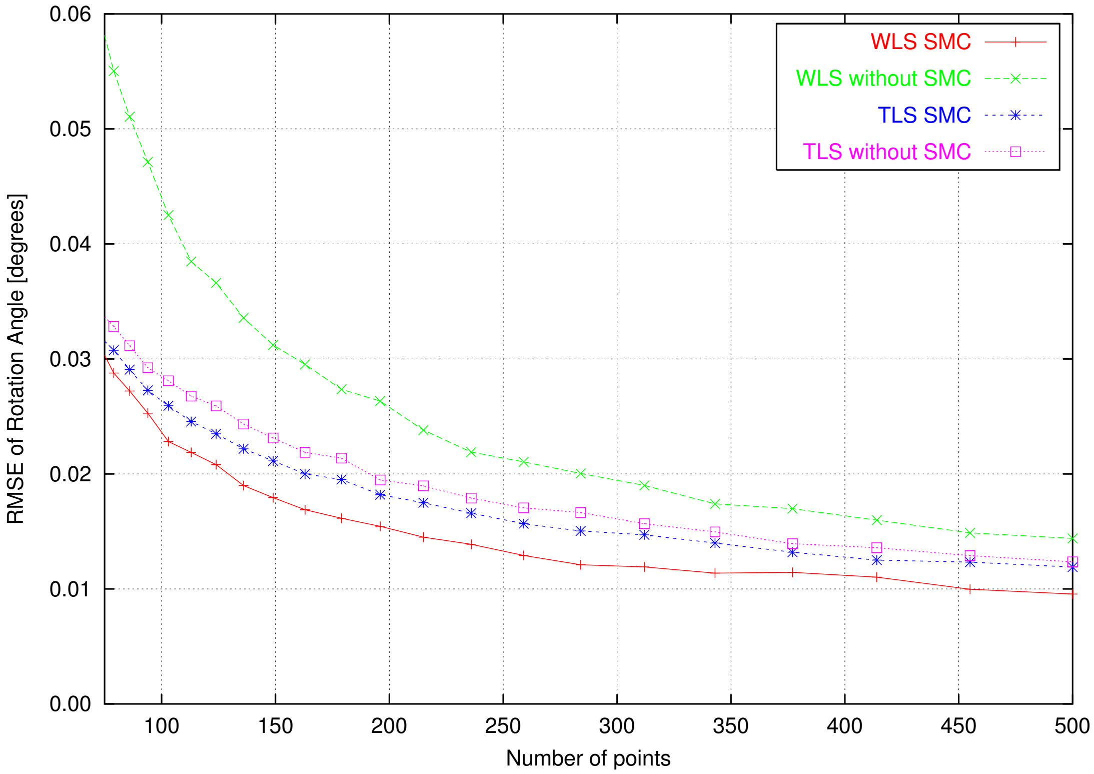


(b) RMSE of translation distance for varying Slash image position noise.

Figure 7.6: Rotational and Translational accuracy for varying Slash image position noise.

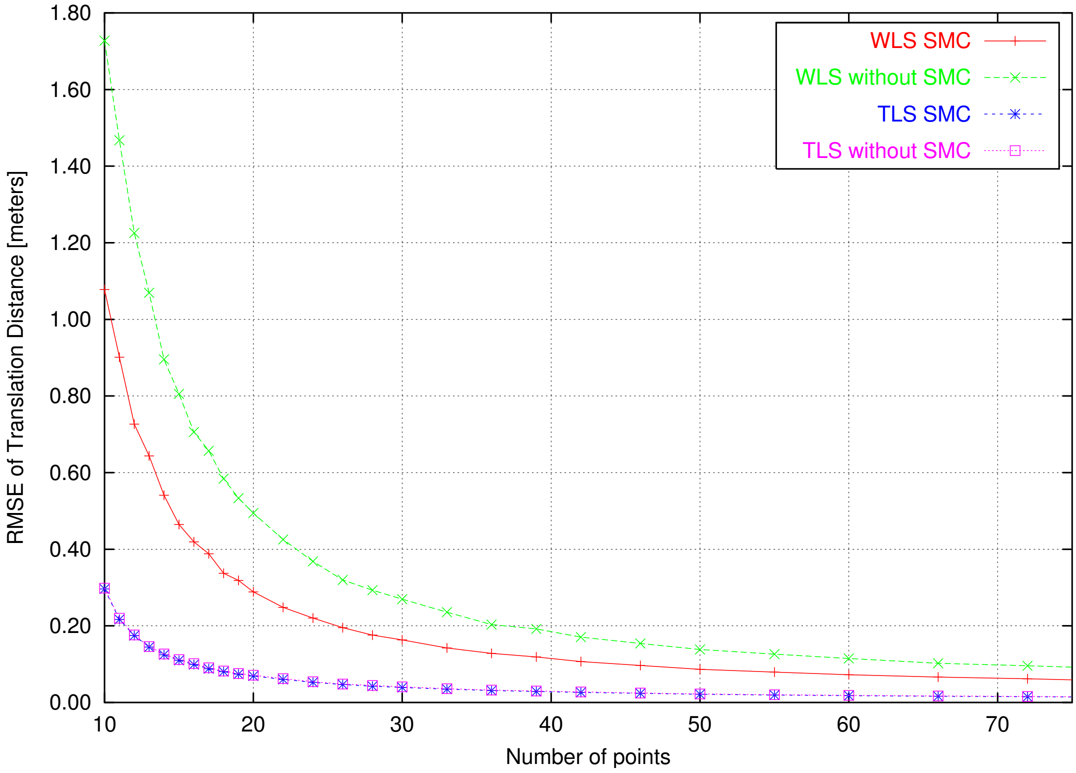


(a) RMSE of rotation angle for interval of 10 to 75 points.

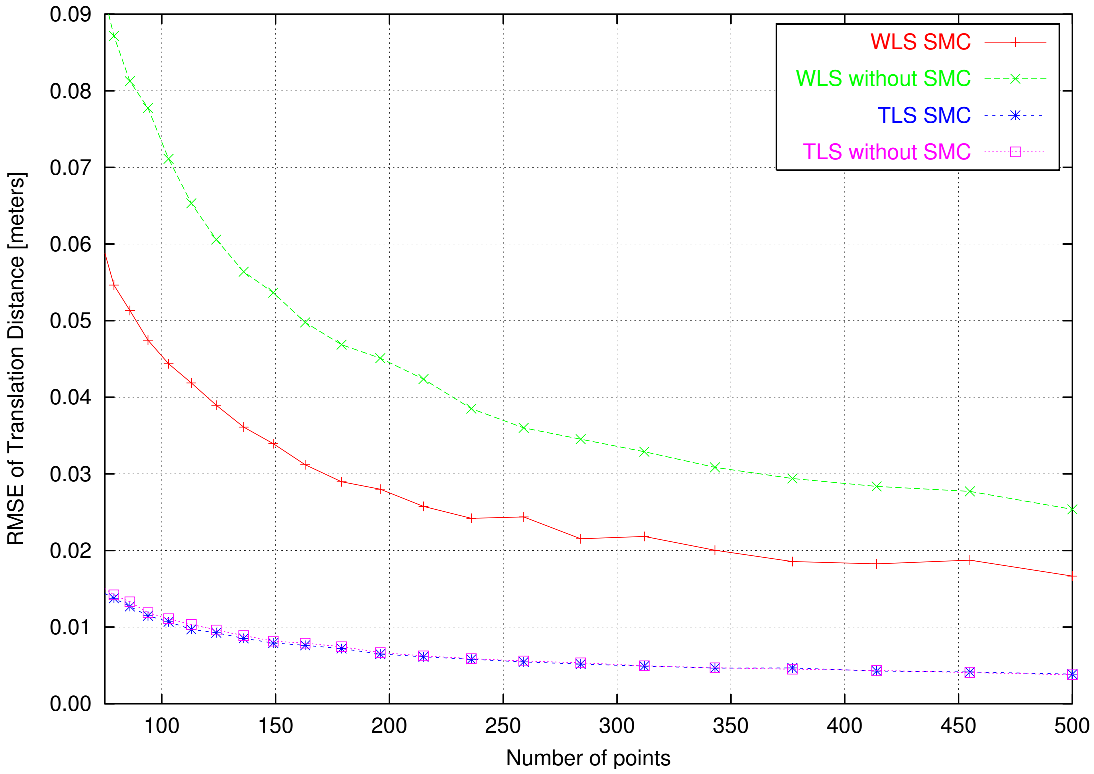


(b) RMSE of rotation angle for interval of 75 to 500 points.

Figure 7.7: Comparison of weighting methods for rotation angle and varying number of points.

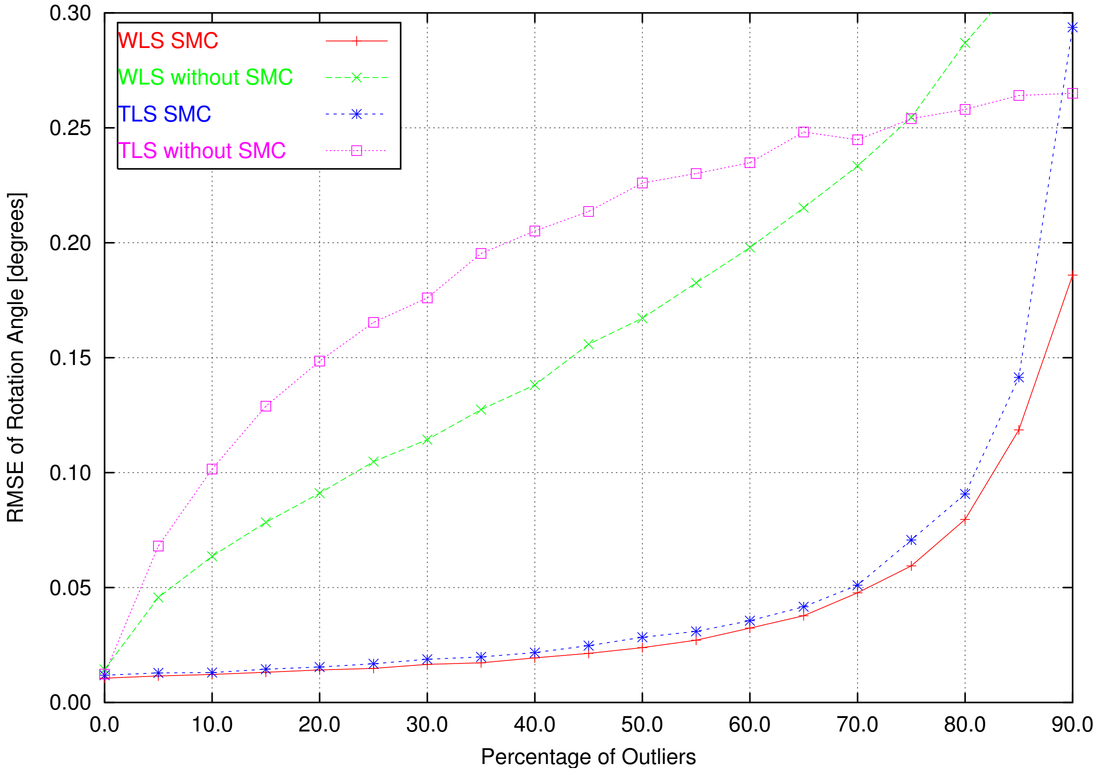


(a) RMSE of translation distance for interval of 10 to 75 points.

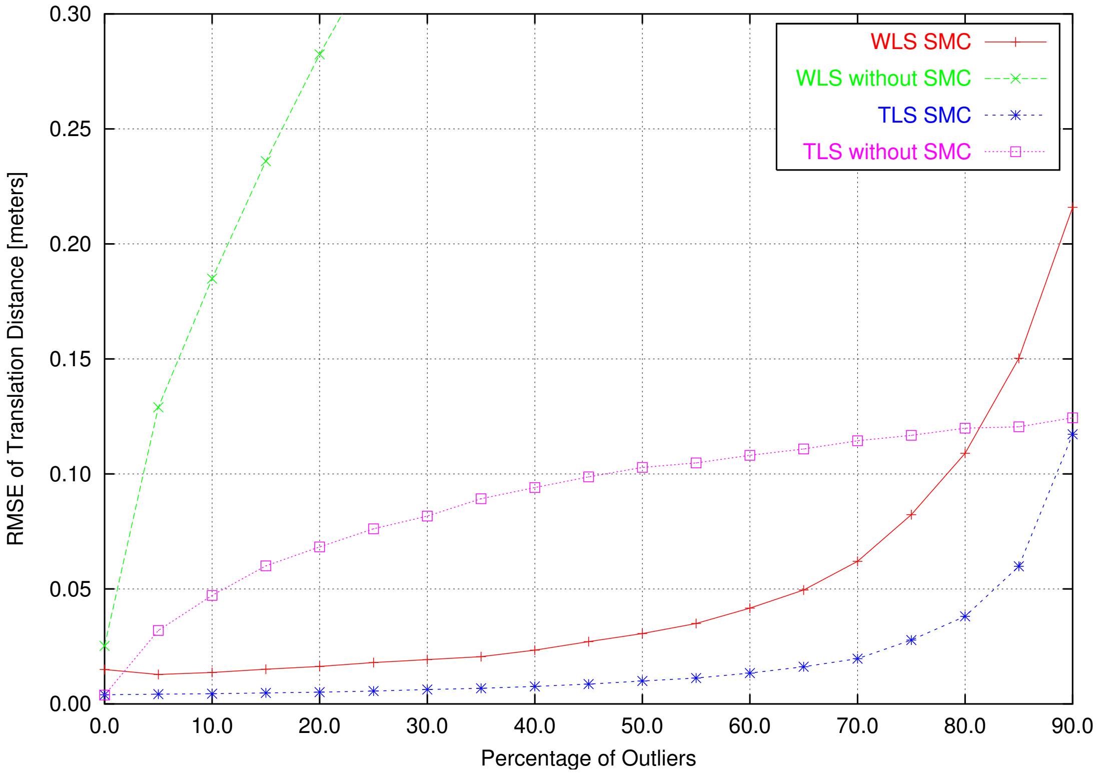


(b) RMSE of translation distance for interval of 75 to 500 points.

Figure 7.8: Comparison of weighting methods for translation distance error and varying number of points.



(a) RMSE of rotation angle for varying percentage of outliers.



(b) RMSE of translation distance for varying percentage of outliers.

Figure 7.9: Comparison of weighting methods for varying percentage of outliers.

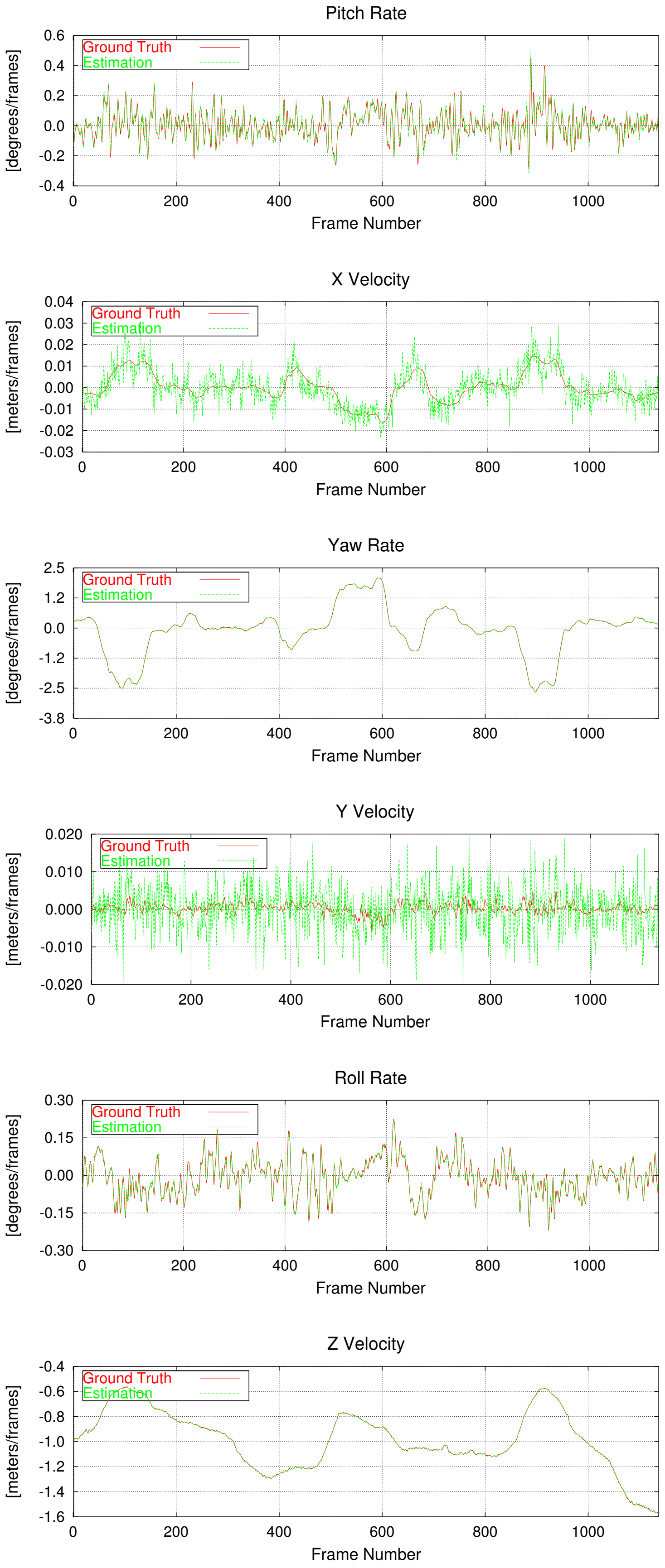
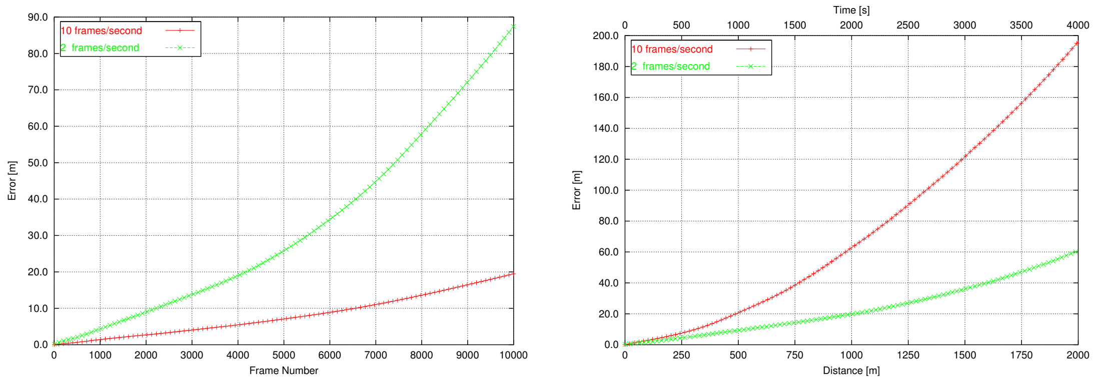


Figure 7.10: Example of motion results with TLS SMC.



(a) Euclidean Ego-Position error vs. frame number.

(b) Euclidean Ego-Position error vs. distance traveled.

Figure 7.11: Super-linearly increment of the Ego-Position Error.

of view), more correspondence problems, more motion uncertainty and less decision capacity because of the low time resolution. Hence, there is a trade-off between frame-rate and the accumulation of estimation errors. This section presents a method which allows a high frame-rate with a low accumulation of errors and it is named *Multi-Frame Estimation* (MFE).

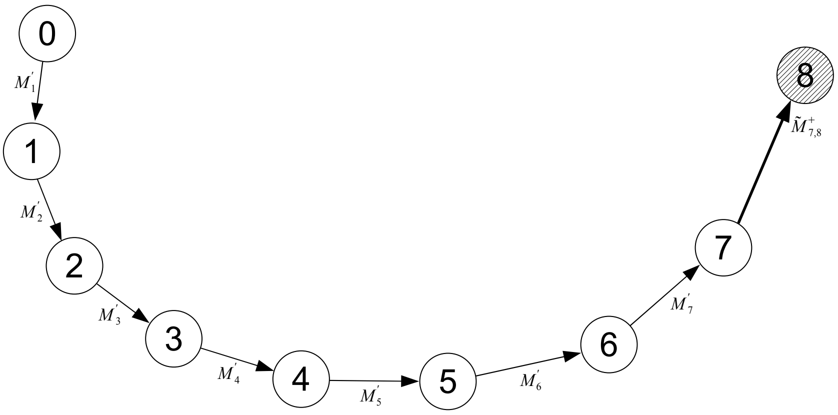
7.5.2 Integration of Multiple-Frames

In this and the following sections, the term *multi-step estimation* will be used to indicate that the estimation of motion between two frames is actually obtained as a function of the information obtained at multiple time steps. On the contrary, *single-step estimation* will denote the estimation of absolute orientation with the frames obtained at only two time instances. The term *Multi-Frame Estimation* (MFE) will denote the method which approaches a multi-step estimation, while the term *Two-Frame Estimation* (TFE) will denote the method which approaches a single-step estimation.

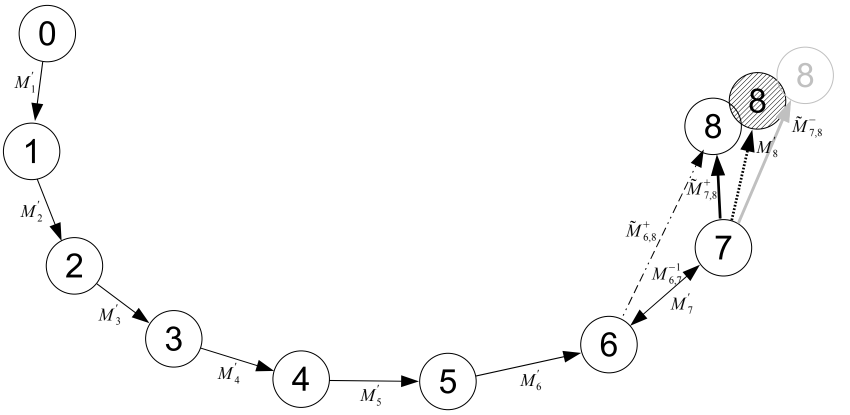
Let us suppose that the tracking algorithm is able to track at least n features over m frames. At each time step, the list of tracked features and corresponding 3D points are stored in lists. Let us define the set of tracked world points for time t_l as $\mathcal{X}_l$. At the current time t_k, the motion $\boldsymbol{M'}_k$ of the vehicle between times t_{k-1} and t_k is required. Two-Frame Estimation implies computing a single-step estimation, by finding the absolute orientation between the data sets obtained at both times, i.e. $\mathcal{X}_k$ and $\mathcal{X}_{k-1}$. The resulting motion is the observed inverse rigid motion of the static scene, i.e. the ego-motion of the camera. Multi-Frame Estimation, instead, iteratively computes the absolute orientation between the clouds $\mathcal{X}_k$ and $\mathcal{X}_{k-i}$ for $i = 1, ..., m-1$. At the end of each iteration the current motion estimate $\boldsymbol{M'}_k$ is refined by the integration with the multi-step result (i.e. the absolute orientation between the clouds obtained at time t_{k-i} and time t_k, denoted as $\tilde{\boldsymbol{M}}^{+}_{k-i,k}$).

Each iteration carries out four main steps:

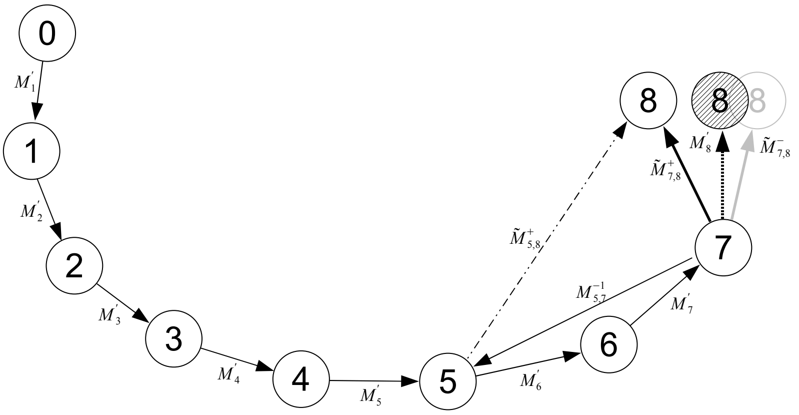
- *Motion Prediction*: at iteration i a motion prediction between time t_{k-i} and time t_k is required before applying the SMC. In the first iteration (i.e. $i = 1$) the



(a) As a first step the absolute orientation between current and previous frame is computed, obtaining a first motion estimate.

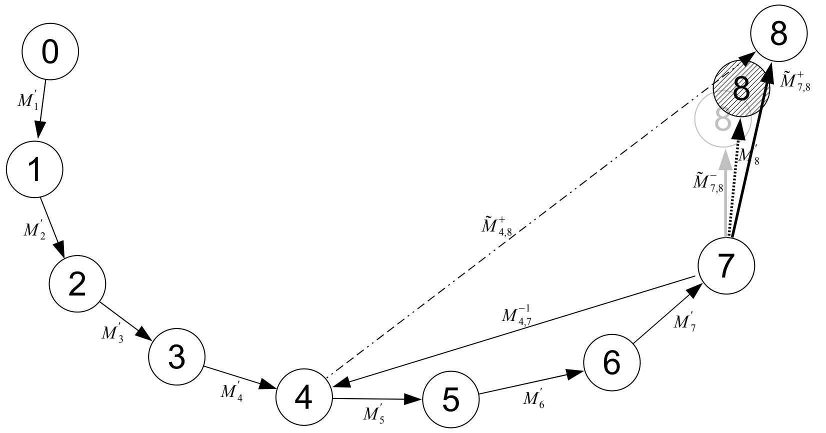


(b) The motion estimate of step 1 (showed in grey) is used as the predicted motion for applying the SMC. The absolute orientation between times 6 and 8 is computed (unfilled black circle) and the motion decomposed in two parts: the first part from state 6 to state 7 and the second part from state 7 to state 8. The first part was computed before and is known. The second motion part is interpolated with the current motion, obtaining a new motion estimate (diagonal hatched circle).

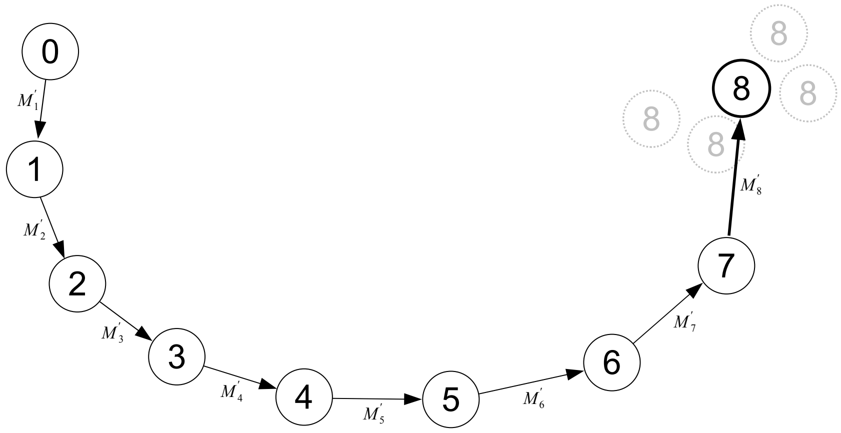


(c) The same procedure as in the previous step is applied here. The resulting absolute orientation between times 5 and 8 is now decomposed as the motion chain $(5 \to 7, 7 \to 8)$. The last link is integrated with the current motion obtaining a new motion estimate.

Figure 7.12: Example of Multi-Frame Estimation.



(d) The same procedure as in step 2 and 3 is applied here, but between times 5 and 8.



(e) The resulting motion step is the interpolation of all multi-step estimates.

Figure 7.12: Example of Multi-Frame Estimation.

motion prediction is taken from the initialization which provides exactly a first motion estimation between the previous and current time. For the second and following iterations the motion prediction is computed as[3]

$$\tilde{\boldsymbol{M}}_{k-i,k}^{-} = \boldsymbol{M}_{k-i}^{-1}\boldsymbol{M}_{k-1}\boldsymbol{M'}_{k} \tag{7.15}$$

i.e. the observed motion between times t_{k-i} and t_{k-1} updated with the current motion estimation of the current time.

- *Smoothness Motion Constraint*: the prediction obtained in the previous step is required in order to apply the SMC. The rejection of outliers, as well the assignments of weights for each 3D point pair in the iteration phase, is more precise than in the initialization phase, because the prediction is much more accurate. The application of the SMC for multi-steps follows exactly the same procedure as explained in Section 7.4, i.e. Equations 7.9 and 7.8 for the WLS and Equations 7.11 to 7.14 for TLS are still valid. Only a small change takes place since the prediction $\boldsymbol{M}'_{k-1}$ used in those equations must be replaced

---

[3] Equation 7.15 is also valid for the first iteration since when $i = 1$ then $\boldsymbol{M}_{k-1}^{-1}\boldsymbol{M}_{k-1} = \boldsymbol{I}_{4\times 4}$ and $\tilde{\boldsymbol{M}}_{k-1,k}^{-} = \boldsymbol{M'}_{k}$.

with the motion prediction matrix $\tilde{\boldsymbol{M}}^{-}_{k-i,k}$ obtained from Equation 7.15.

- *Computation of Absolute Orientation*: the absolute orientation between the sets $\mathcal{X}_{k-i}$ and $\mathcal{X}_k$ is computed, resulting in the multi-step motion $\tilde{\boldsymbol{M}}^{+}_{k-i,k}$.

- *Motion Integration*: Once the camera motion matrix $\tilde{\boldsymbol{M}}^{+}_{k-i,k}$ between times t_{k-i} and t_k is obtained, it must be integrated with the current step estimation $\boldsymbol{M'}_k$. This is performed by interpolation. The interpolation of matrices makes sense, if they are estimates of the same motion. This is not the case, since the current step motion matrix $\boldsymbol{M'}_k$ is an estimation of motion between times t_{k-1} and t_k (one step motion), and the multi-step motion matrix $\tilde{\boldsymbol{M}}^{+}_{k-i,k}$ is an estimation of motion over i frames. If $i = 1$, the interpolation is straightforward. If $i > 1$, the multi-step motion matrix must be decomposed in two motion matrices: the motion matrix from time t_{k-i} to t_{k-1} and the motion matrix from time t_{k-1} to time t_k. The latter is the motion matrix to be interpolated. Thus, the multi-step matrix obtained in the previous step is expressed as the product of the two matrices

  $$\tilde{\boldsymbol{M}}^{+}_{k-i,k} = \boldsymbol{M}_{k-i,k-1}\tilde{\boldsymbol{M}}^{+}_{k-1,k} \tag{7.16}$$

  where the first matrix of the right hand side is be obtained from the total motion matrices as

  $$\boldsymbol{M}_{k-i,k-1} = \boldsymbol{M}^{-1}_{k-i}\boldsymbol{M}_{k-1}. \tag{7.17}$$

  Replacing Equation 7.17 in Equation 7.16 and solving for $\tilde{\boldsymbol{M}}^{+}_{k-1,k}$ we obtain

  $$\tilde{\boldsymbol{M}}^{+}_{k-1,k} = \boldsymbol{M}^{-1}_{k-1}\boldsymbol{M}_{k-i}\tilde{\boldsymbol{M}}^{+}_{k-i,k}. \tag{7.18}$$

  The rotation matrices of $\boldsymbol{M'}_k$ and $\tilde{\boldsymbol{M}}^{+}_{k-1,k}$ are converted to quaternions in order to apply a spherical linear interpolation (see appendix A for details). The translation vectors are linearly interpolated. The interpolation factors are obtained in the following way. Let us define $f_{k-i,k}$ as the inverse of the square of the sum of Mahalanobis distances (or residuum) of Equation 5.13, obtained when computing the absolute orientation between data sets $\mathcal{X}_{k-i}$ and $\mathcal{X}_k$. The interpolation factors are obtained from the values:

$f_{k-i,k}$	for the multi-step motion matrix; and
$\sum_{l=1}^{i-1} f_{k-l,k}$	for the integrated motion matrix.

  which corresponds to the weighted average of all multi-step motions. This means that every additional step has less and less impact on the final result.

Figures 7.12(a) to 7.12(e) exemplifies one iteration of the Multi-Frame Estimation. With this algorithm, the estimation improves because of the integration of more measurements. Also, the accumulation of errors is reduced considerably, as shown in the next section.

7.5.3 Simulation Results for MFE

This section shows simulation results for Multi-Frame Estimation. The simulation data used in the previous section is also used in this section. However, in order to generate a long distance navigation data set, the motion sequence shown in Figure 7.3 is used repeatedly, starting over after the end of the sequence. This way we obtain a sequence, which can be so long as desired.

The generation of point data is performed using the procedure of Section 7.4.5. In the next procedure n is the amount of tracked feature points and $LostFeat$ is a parameter to model the percentage of lost features at each time step:

```
1. Load first motion step, i.e. $\bar{\boldsymbol{R}}_{\boldsymbol{k}}$ and $\bar{\boldsymbol{t}}_{\boldsymbol{k}}$ for $k = 0$.
2. For $i = 1$ to $i = n$.
    (a) Run the procedure of Section 7.4.5 from step $1$.
    (b) Make $i = i + 1$.
3. Load next motion step, i.e. $\bar{\boldsymbol{R}}_{\boldsymbol{k}}$ and $\bar{\boldsymbol{t}}_{\boldsymbol{k}}$ for $k = k + 1$.
4. For $i = 1$ to $i = n$.
    (a) Generate uniform random number $\lambda$ in the range $[0, 100]$.
    (b) If $\lambda > LostFeat$ make $\bar{\boldsymbol{p}}'_i = \bar{\boldsymbol{x}}'_i$, $\bar{\boldsymbol{p}}_i = \bar{\boldsymbol{x}}_i$, $\boldsymbol{p}'_i = \boldsymbol{x}'_i$ and $\boldsymbol{p}_i = \boldsymbol{x}_i$ and go to step $6$. of the procedure of Section 7.4.5.
    (c) If $\lambda <= LostFeat$ go to step $1$. of the procedure of Section 7.4.5.
    (d) Make $i = i + 1$.
5. Go to step $3$.
```

In the following experiments we use $n = 500$ and $LostFeat = 25\%$.

Figure 7.5.3 compares the performance between Two-Frame Estimation and Multi-Frame Estimation using Weighted Least Squares. Two-Frame Estimation and Multi-Frame Estimation for different levels of integration were computed and the error in position was plotted against the distance traveled. "MFE level m" means that the maximal integration time is m, i.e. MFE is computed iteratively between current time t_k and time t_{k-m}, as specified in previous section. Observe that MFE allows a substantial reduction of the navigation error, even for level 2, which implies the integration of only one more frame than the TFE. Every additional level of integration reduces the error even more, although the reduction of the error becomes slower. Figure 7.14 shows some examples of the traveled path, comparing ground truth and estimated ego-position. Figure 7.14(a) shows the estimation with MFE level 6 for the first 15 kilometers traveled. Observe that at the ending position, the estimation error is quite small both in position and orientation. Figure 7.14(b) shows a comparison of

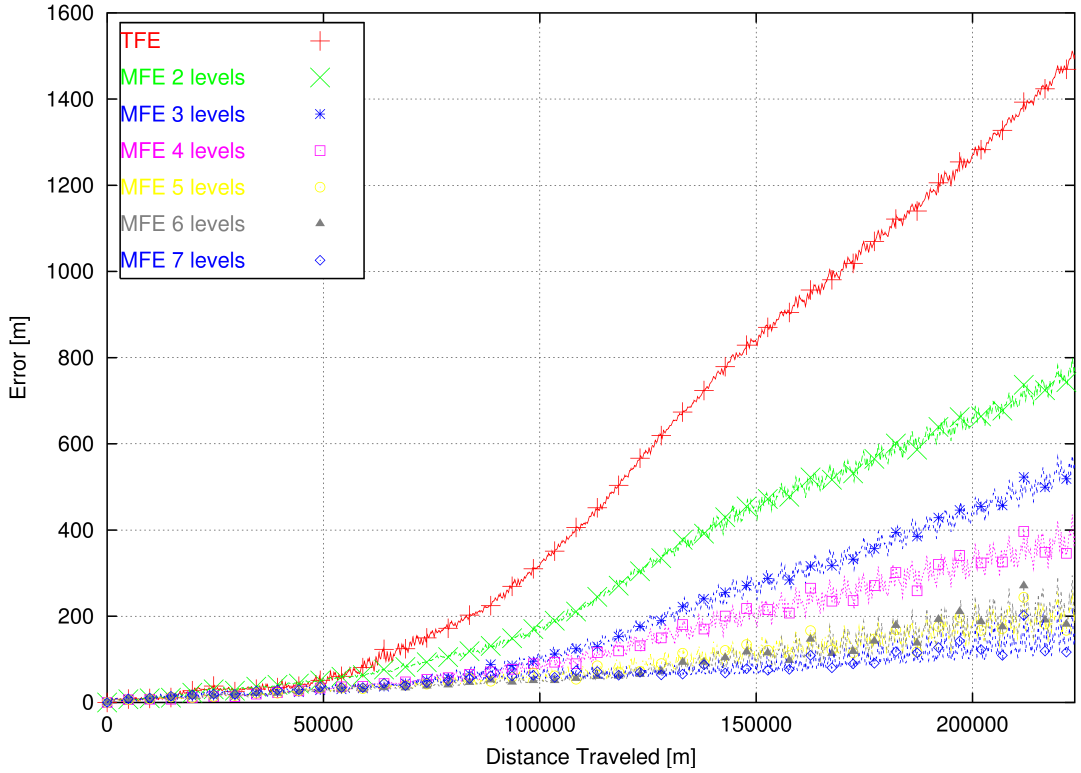


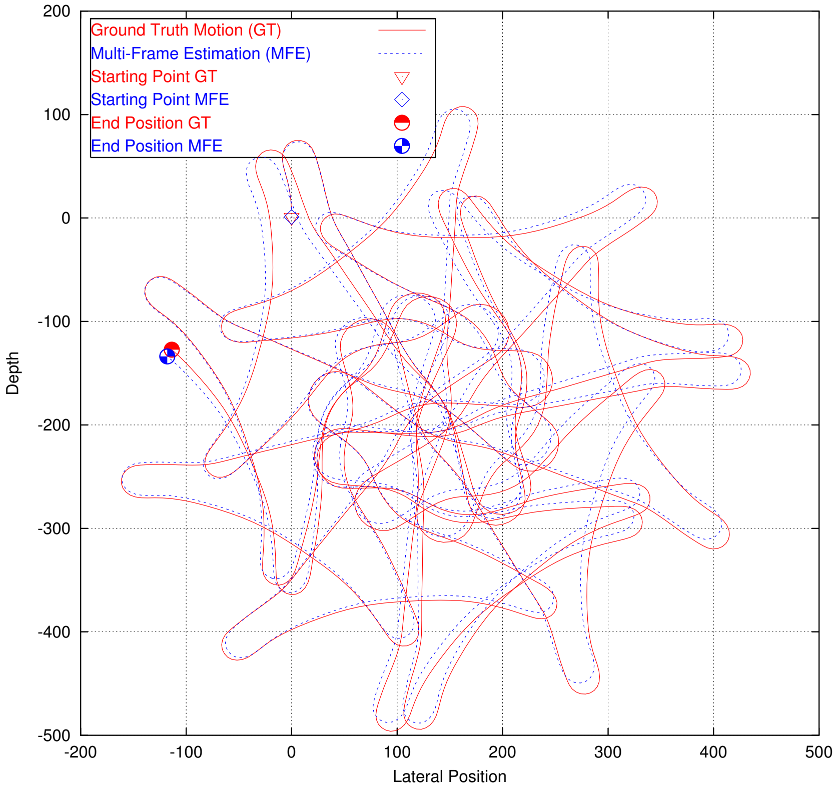
Figure 7.13: Performance of Multi-Frame approach as a function of distance traveled for WLS.

ego-position estimation between the TFE and MFE estimation after a traveled distance of $50\ km$. The error using MFE remains quite small and, although an evident increase in the orientation error is observed, the position error is negligible in comparison to the error made with TFE, which exceeds the 250 meters.

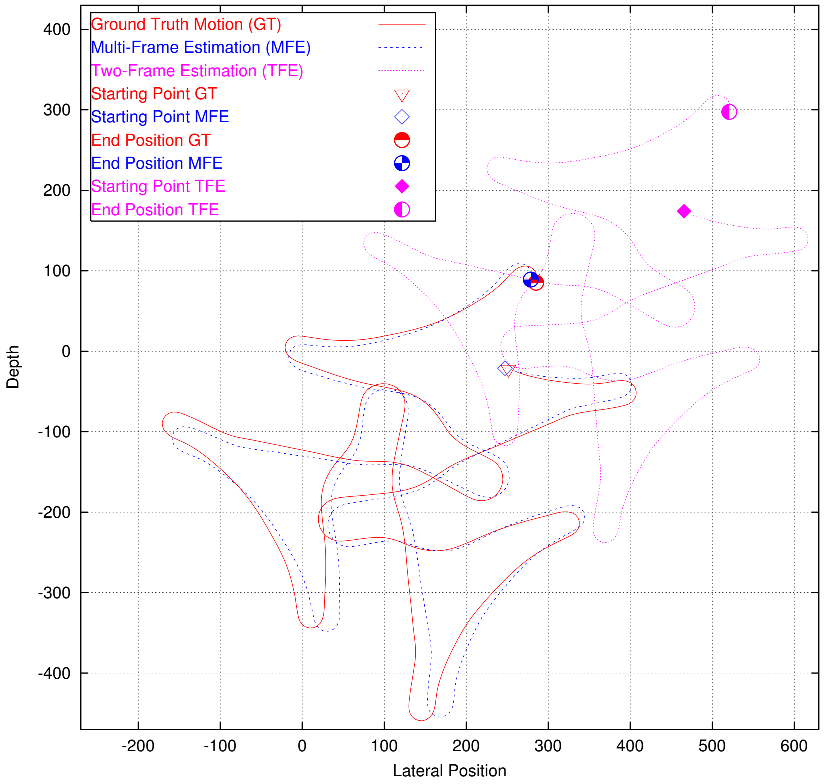
When computing the absolute orientation with TLS, similar results are obtained. Figure 7.15(a) shows the results of MFE using TLS, where it can be seen that same improvements factors, as with WLS, are achieved. Figure 7.15(b) shows the orientation error as a function of traveled distance. The curves also show an improvement with increasing level of integration, but in contrast to the ego-position error, there is almost a constant reduction of the error with every additional level of integration. In fact, it is the reduction of the orientation error that allows almost a linear relationship between traveled distance and ego-position error, as already observed by Olson *et al.* [OMSM03].

7.6 Integration of Filtered Data

Chapter 4 introduced how to optimally estimate the 3D position and 3D velocity of world points fusing optical flow, stereo and the motion parameters of the camera. The simulation results showed that the iterative refinement over time of the position of 3D points with Kalman filters allows a better estimation, than treating the individual measurements as being uncorrelated. It is therefore tempting to use the filter estimates when computing ego-motion and not the triangulated measurements.

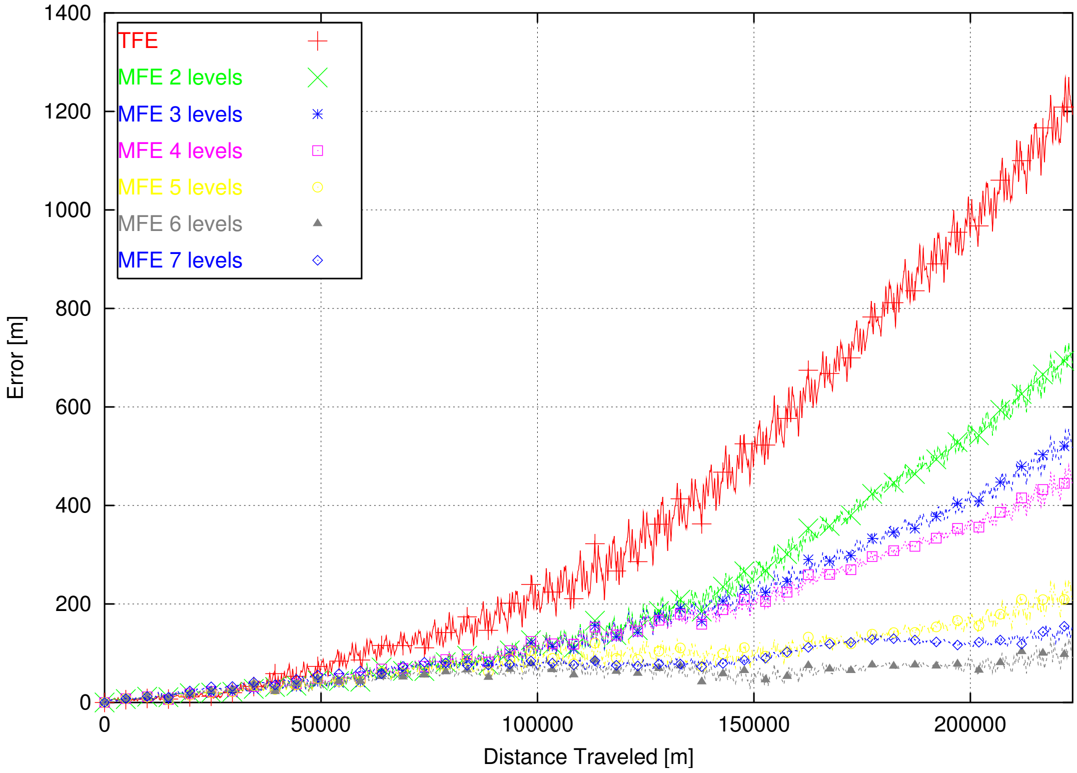


(a) Integrated motion estimation for the first 15 kilometers traveled.

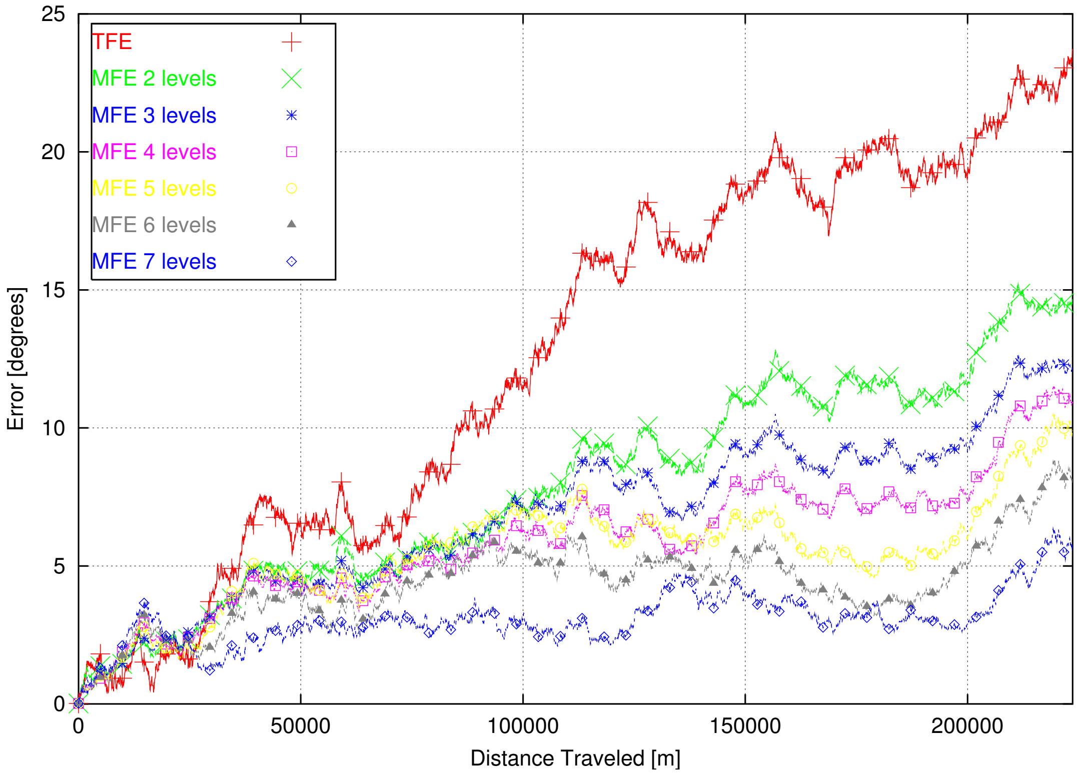


(b) Integrated motion estimation after 100 kilometers traveled.

Figure 7.14: Examples of integrated motion estimation using MFE and TFE.



(a) Euclidean distance position error



(b) Orientation error.

Figure 7.15: Performance of Multi-Frame approach as a function of distance traveled.

However, the filter requires first the motion of the camera in order to be able to tell the optimal *current* position of the points. In other words, at time t_k the motion of the camera between times t_{k-1} and t_k is required in order to obtain the new filtered position of the points, i.e. the required filtered 3D position at time t_k is not available until ego-motion is computed. The ego-motion algorithm cannot, therefore, use the optimal estimation of the point for current time t_k. However, previous filter outputs are available and can be used as measurements for the computation of absolute orientation. This means that when computing the absolute orientation between current and previous time, the points corresponding to the data set $\mathcal{X}_{k-1}$ will not be just the triangulated features points but the estimates provided by the Kalman filters. Observe that not only the immediate previous data set may contain filtered data, but all previous data sets ($\mathcal{X}_{k-2}, \mathcal{X}_{k-3}, \ldots$). This means that not only TFE benefits from the filter outputs, but also MFE.

One important aspect to consider when using the filter outputs for the estimation of ego-motion is the danger of falling into a positive feedback loop. Positive feedback loops enhance or amplify changes, moving a system away from its equilibrium state and make it more and more unstable. Suppose that our system (as shown in Figure 7.1) is in equilibrium. This means the system works stably, computing stereo and optical flow, obtaining the ego-motion of the vehicle from non-filtered (current time) and filtered (previous times) 3D point, and updating 3D point position and velocity of world points. Now let us suppose that a small perturbation is introduced to the system, for example, the tracking or stereo algorithm computes too many outliers (because of repetitive structure in the scene for example). The Ego-motion system will compute the motion of the camera but with less accuracy than normal. The points updated with Kalman Filter will also have a larger error because of the inaccuracy of the ego-motion parameters. The equilibrium is already broken. Every posterior cycle amplifies this effect and the system go into an uncontrollable state.

In order to avoid this we define a *firewall*[4]. A firewall defines a limit for the integration of information. The firewall in our system is achieved through a threshold for the age of the tracked features. The filtered 3D position of feature points that are younger than this threshold, are not stored in the 3D Point List. By doing this we avoid the computation of ego-motion with filtered young features. Instead of the filtered position, the triangulated 3D position of young feature points is used, which is not affected by a previous wrong computation of ego-motion. A relatively old feature point will have a converged state in the KF model and a more stable behavior with respect to wrong estimates.

Figure 7.16 shows the expected root mean square distance error as a function of the point distance, comparing the performance of using the ego-motion results against the results of using the real motion parameters (ground truth). Every curve shows the results for point tracked a specified number of times. The curve "Age 2 Est" shows the RMSE for points with age 2 (tracked 2 times). "Est" means that the ego-motion estimation was used for building the $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$ matrices of the Kalman filter system model. If, instead, the ground truth motion is used for building the matrices, then the curve is labeled "GT". The motion sequence was simulated as constant forward motion. Multi-frame estimation with level 3 was used and all parameters

---

[4] The term *firewall* was taken from Nister *et al.* [NNB04] [NNB06]. In the papers, the authors use this term for defining a method that prevents the system falling into a positive feedback loop.

remain the same as in the previous section, only that now the filtered 3D position of feature points tracked at least 3 times are used. Observe that the ground truth curves are the same as those of Figure 4.9.

The results show that the RMSE of Kalman filtered points, when using the estimated ego-motion parameters, are almost the same as the RMSE of filtered points using the ground truth motion. The error in the ego-motion estimation is propagated to the 3D points and therefore the RMSE is slightly larger than when comparing with ground truth motion. This effect is only visible when the points were tracked several times, as can be seen by points with age 11 and 15. Nevertheless, the difference between corresponding curves is very small and becomes negligible if compared with the RMSE of the unfiltered curve, which is least 3 times larger.

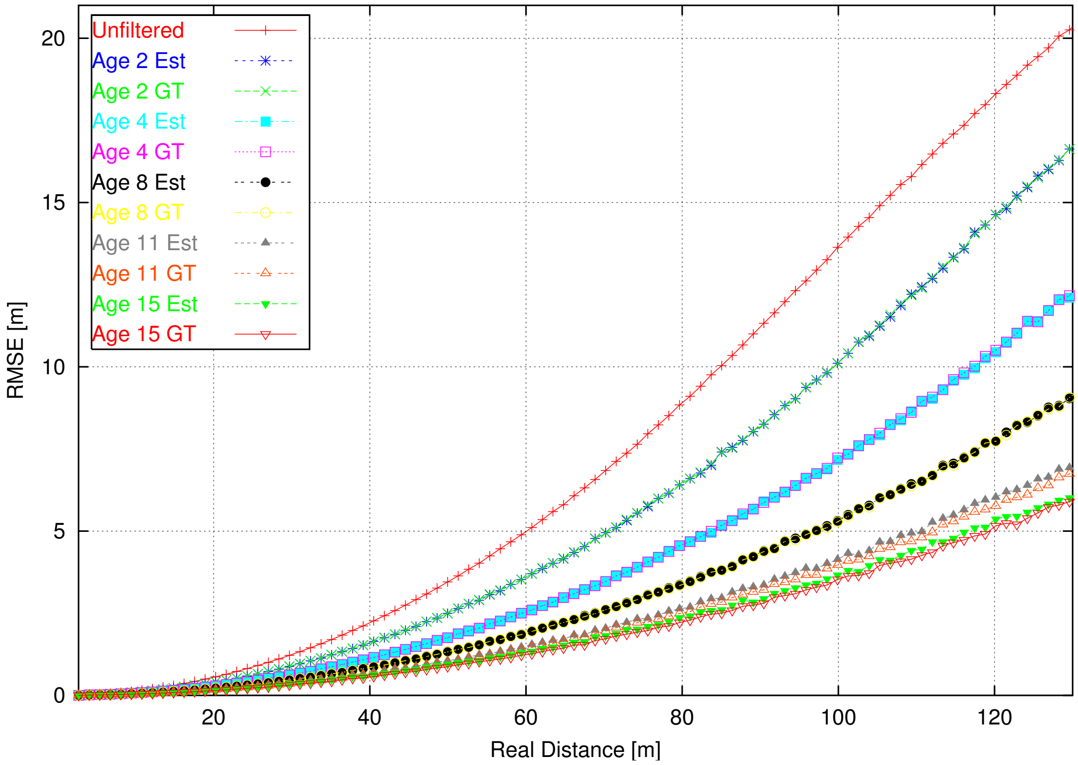


Figure 7.16: Relative Performance using GT and Estimated Ego-Motion.

7.7 Integration with Inertial Sensors

As shown in the next chapter, the ego-motion algorithm proposed above is extremely robust and it is able to compute the ego-motion of the vehicle in typical traffic situations. Nevertheless, such robustness is based on the assumption that the images are correctly acquired and the levels of noise are acceptable, as already mentioned in Section 3.2. The accuracy and correctness of the stereo and tracking algorithm depend highly on the images acquired, and the cameras might provide partially wrong, or no output at all. Some cases where problems are expected are:

- changing lighting conditions (e.g. entering a tunnel);

- partial or total occlusions (e.g. with the windshield wiper);
- distortions (e.g. the distortion produced by the raindrops or dirtiness on the windshield);
- the direct exposure of light or through reflections (e.g. sun light or the headlight of vehicles coming in the opposite direction);
- landscapes with little structure (e.g. flat and even surfaces with no trees or vertical structures).

If the vehicle is provided with additional systems for measuring motion, like a speedometer, a yaw-rate sensor and GPS, then the redundant information can be used to provide a better estimate and to increase robustness. A combined estimate is easily obtained if a confidence for every quantity is available. In the ideal case, the estimate is provided with a corresponding covariance matrix. Let us suppose $\boldsymbol{m}_i = (\phi_i, \psi_i, \theta_i, t_{xi}, t_{yi}, t_{zi})^T$ for $i = 1, 2, \ldots, n$ as the estimates of motion provided by the ith system and $\boldsymbol{C}_i$ their corresponding estimate covariance matrices. Then the combined optimal estimate $\boldsymbol{m}$ is obtained as

$$\boldsymbol{m} = \boldsymbol{C} \,\left(\boldsymbol{C}_1^{-1}\boldsymbol{m}_1 + \boldsymbol{C}_2^{-1}\boldsymbol{m}_2 + \ldots + \boldsymbol{C}_n^{-1}\boldsymbol{m}_n\right) \tag{7.19}$$

where $\boldsymbol{C}$ is the corresponding covariance matrix of the new estimate which is obtained as

$$\boldsymbol{C} = \left(\boldsymbol{C}_1^{-1} + \boldsymbol{C}_2^{-1} + \ldots + \boldsymbol{C}_n^{-1}\right)^{-1}. \tag{7.20}$$

In the experimental results of the next chapter the test vehicle uses a speedometer and a yaw-rate sensor. The speedometer provides a velocity estimate v with variance σ_v^2. The yaw-rate sensor delivers a yaw-rate estimate $\dot{\psi}$ with variance $\sigma_{\dot{\psi}}^2$. The motion model achieved with both estimates is planar. For short interval of time $\Delta\mathrm{t}$, the vector $\boldsymbol{s} = (\dot{\psi}, v)^T$ can be considered constant, and the motion parameters of the vehicle are obtained as a function of $\boldsymbol{s}$, i.e.

$$\boldsymbol{m}_{IS} = \boldsymbol{F}(\boldsymbol{s}) = \frac{1}{\dot{\psi}} \begin{bmatrix} 0 \\ \Delta\mathrm{t}\dot{\psi}^2 \\ 0 \\ \cos(\Delta\mathrm{t}\dot{\psi} - 1)v \\ 0 \\ -\sin(\Delta\mathrm{t}\dot{\psi})v \end{bmatrix} \tag{7.21}$$

The corresponding covariance matrix of $\boldsymbol{m}_{IS}$ can be obtained as

$$\boldsymbol{C}_{IS} = \boldsymbol{J}_{\boldsymbol{F}} \begin{bmatrix} \sigma_{\dot{\psi}}^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix} \boldsymbol{J}_{\boldsymbol{F}}^T \tag{7.22}$$

where $\boldsymbol{J}_{\boldsymbol{F}}$ is the matrix of first partial derivatives of $\boldsymbol{F}(\boldsymbol{s})$, i.e.

$$\boldsymbol{J}_{\boldsymbol{F},ij} = \frac{\partial \boldsymbol{F}_{ij}}{\partial \boldsymbol{s}_j} \tag{7.23}$$

Solving Equation 7.22 results in

$$\boldsymbol{C}_{IS} = \begin{bmatrix} j_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & j_{22} & 0 & j_{24} & 0 & j_{26} \\ 0 & 0 & j_{33} & 0 & 0 & 0 \\ 0 & j_{24} & 0 & j_{44} & 0 & j_{46} \\ 0 & 0 & 0 & 0 & j_{55} & 0 \\ 0 & j_{26} & 0 & j_{46} & 0 & j_{66} \end{bmatrix} \tag{7.24}$$

where

$$j_{22} = \Delta\mathrm{t}^2 \sigma_{\dot\psi}^2 \tag{7.25}$$

$$j_{24} = \frac{\Delta\mathrm{t} v \sigma_{\dot\psi}^2 \big(\cos\psi - 1 + \psi\sin\psi\big)}{\dot\psi^2} \tag{7.26}$$

$$j_{26} = \frac{\Delta\mathrm{t} v \sigma_{\dot\psi}^2 \big(\sin\psi - \psi\cos\psi\big)}{\dot\psi^2} \tag{7.27}$$

$$j_{44} = \sigma_x^2 + \frac{\sigma_v^2 \big(\cos\psi - 1\big)^2}{\dot\psi^2} + \frac{v^2 \sigma_{\dot\psi}^2 \big(\cos\psi - 1 + \psi\sin\psi\big)^2}{\dot\psi^4} \tag{7.28}$$

$$j_{46} = \frac{v^2 \sigma_{\dot\psi}^2 \big(\psi\cos\psi - \sin\psi\big)\big(\cos\psi - 1 + \psi\sin\psi\big)}{\dot\psi^4} - \frac{\sigma_v^2 \big(\cos\psi - 1\big)\sin\psi}{\dot\psi^2} \tag{7.29}$$

$$j_{66} = \frac{\sigma_v^2 \big(\sin\psi\big)^2}{\dot\psi^2} + \frac{v^2 \sigma_{\dot\psi}^2 \big(\sin\psi - \psi\cos\psi\big)^2}{\dot\psi^4} \tag{7.30}$$

$$j_{11} = j_{33} = j_{55} = k_\infty \tag{7.31}$$

where $\psi = \Delta\mathrm{t}\dot\psi$ and k_∞ is a very large number which is used to model the unavailability of the corresponding parameter. The scalar σ_x^2 in j_{44} is also added in order to avoid the singularity of the covariance matrix, otherwise the matrix has only 5 degrees of freedom.

When computing ego-motion with the method described in this chapter, the covariance matrix of the estimated parameters can be obtained as the inverse matrix of second partial derivatives of the objective function of Equation 5.13 evaluated at its minimum, i.e.

$$\boldsymbol{C}_{VO} = \boldsymbol{H}_{VO}^{-1} \tag{7.32}$$

where

$$\boldsymbol{H}_{VO,ij} = \frac{\partial^2 J}{\partial \boldsymbol{m}_i \partial \boldsymbol{m}_j} \tag{7.33}$$

and $\boldsymbol{m}_{VO} = (\phi, \psi, \theta, t_x, t_y, t_z)^T$ is the parameter vector found.

The optimal motion parameter vector fusing inertial sensor and visual odometry is obtained as

$$\boldsymbol{m} = \boldsymbol{H}_{VO}\boldsymbol{m}_{VO} + \boldsymbol{C}_{IS}^{-1}\boldsymbol{m}_{IS} \tag{7.34}$$

and the fused estimation covariance matrix is

$$\boldsymbol{C} = \left(\boldsymbol{H}_{VO} + \boldsymbol{C}_{IS}^{-1}\right)^{-1} \tag{7.35}$$

Observe that Equations 7.35 and 7.35 are only valid if the camera and vehicle coordinate systems have been aligned.

7.8 Summary

The visual ego-motion (or visual odometry) problem implies the extraction of the motion parameters of the camera between two time instances by analyzing the changes of brightness patterns in the acquired images. A plethora of work on ego-motion computation has been published in the last two decades. All ego-motion methods can be classified as belonging to one of two main groups; monocular methods and multi-ocular methods. The main difference between both groups is the type of result obtained. Multi-ocular methods recover the rigid body transformation of the camera platform between two time instances. Monocular methods, instead, can solve the relative orientation problem. The missing scale factor can be recovered by making assumptions about the motion of the camera, or about the structure of the scene. Both groups are classified according to the way in which they integrate the time component: optical flow-based methods, normal flow-based methods, direct methods, and landmark-based methods. This chapter has reviewed some main contributions on ego-motion estimation for each category.

One of the contributions from this chapter is the Smoothness Motion Constraint. The SMC is an effective rejection rule applied to detect outliers (moving points and false correspondences) from the cloud of points. It also helps to reduce the contribution of noisy measurements by assigning weights to the data. Two versions of the SMC are proposed, one version for the weighted least squares and one version for total least squares approach. The WLS version of the SMC determines the weight which defines the contribution of the pair of points as a whole, without discrimination of their components. A weight of zero is applied when the pair of point do not show a coherent motion. Otherwise, the method weighs the pair of points according to the distance between prediction and measurement. The TLS version of the SMC allows to establish a weight matrix for each 3D point. The applied weights also depend on the error between prediction and measurement, but propagating the prediction error from the image to the 3D Euclidean space.

Simulation results are carried out with Gaussian and Slash noise showing the superiority of the SMC version over the standard weighting methods. An important result from the simulations is that, under Gaussian noise the TLS solution for rotation is not necessarilly better than for WLS. This is a consequence of the penalization imposed in the closed form solution of the method, as already addressed in Chapter 5. On all other simulation scenarios, the TLS version outperformed the WLS version, showing a more stable performance. An analysis of the stability of the results when dealing with outliers was also carried out, showing break down points larger than 50%. The decision between choosing WLS and TLS for the ego-motion computation should depend upon; the number of tracks available, the expected noise, the expected number of contaminated data (i.e. outliers), the time requirements and the accuracy required. The simulations shown in this chapter can help in the selection.

The algorithm proposed in this chapter for the computation of ego-motion carries out four main steps: 1) motion prediction; 2) application of the SMC; 3) motion correction; and 4) motion integration. If these four steps are carried out iteratively, a predictor/corrector algorithm is obtained. This chapter proposes a method that carries out the four steps, integrating at each time a new cloud of points into the estimation. This allows to reduce the accumulation of errors when concatenating multiple estimates in a single global estimation. Simulation results are shown and the improvements achieved are presented.

This chapter also describes the integration of the filtered data into the ego-motion estimation, avoiding positive feedback loops using a firewall. Finally, the covariance matrix for the inertial sensor (velocity and yaw-rate sensor) of the vehicle are derived. This allows to fuse inertial sensor information with the visual odometry estimation for a more robust estimation.

Chapter 8

Experimental Results

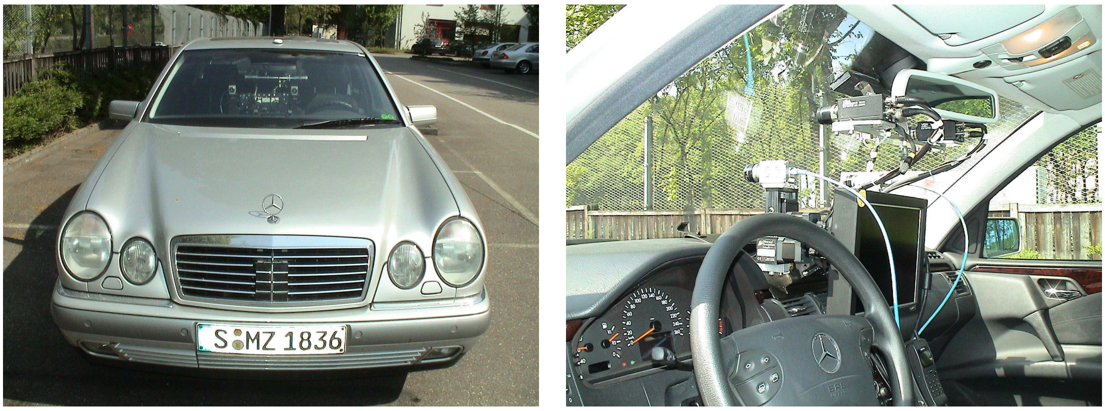
8.1 Introduction

This chapter shows some experimental results of the visual ego-motion estimation presented in the previous chapter and of the estimation of point position and velocity as presented in chapter 4. Although the methods presented in this dissertation were developed for traffic scenarios, the application area can easily be extended to other application, areas such as robotics in indoor environments, Simultaneous Localization And Mapping (SLAM) applications, reconstruction of 3D scenes, or object pose estimation. After a short description of the optical flow and stereo algorithms in Section 8.1.1, Section 8.2 shows some results of ego-motion estimation with two real world sequences of stereo images in typical traffic situations. Section 8.3 applies the method in an off-road scenario, where the vehicle drives over a uneven surface. The results of ego-motion are shown by reconstructing the scene viewed by the cameras over the sequence. Results of an indoor environment are shown in Section 8.4. Finally, Section 8.5 shortly describes how to apply the method for object pose-estimation and shows some results used for crash test analysis.

8.1.1 Optical Flow and Stereo Implementation

The flow and stereo algorithms that can be used for the estimation of ego-motion and point velocity are not constrained to a specific implementation. In fact, our approach was tested with different algorithms obtaining almost identical results. Nevertheless, we describe shortly the stereo and optical flow algorithms used in the experimental results of this chapter.

The stereo algorithm we use in this chapter is described in Franke [Fra00] and Badino [Bad02]. It works based on a coarse-to-fine scheme in which a Gaussian pyramid for left and right images is constructed, with a sampling factor of 2. The search for the best disparity is only performed at the top level of the pyramid and then a translation of the disparity map is made to the next level, where a correction is done within an interval ± 1 of the calculated disparity. We use the sum of squared differences (SSD) as the default correlation function. Different filters and constraints are applied between pyramid translations. The zero-mean normalized cross-correlation



(a) UTA vehicle. (b) Installation of the camera system.

Figure 8.1: Urban Traffic Assistance Research Vehicle (UTA)

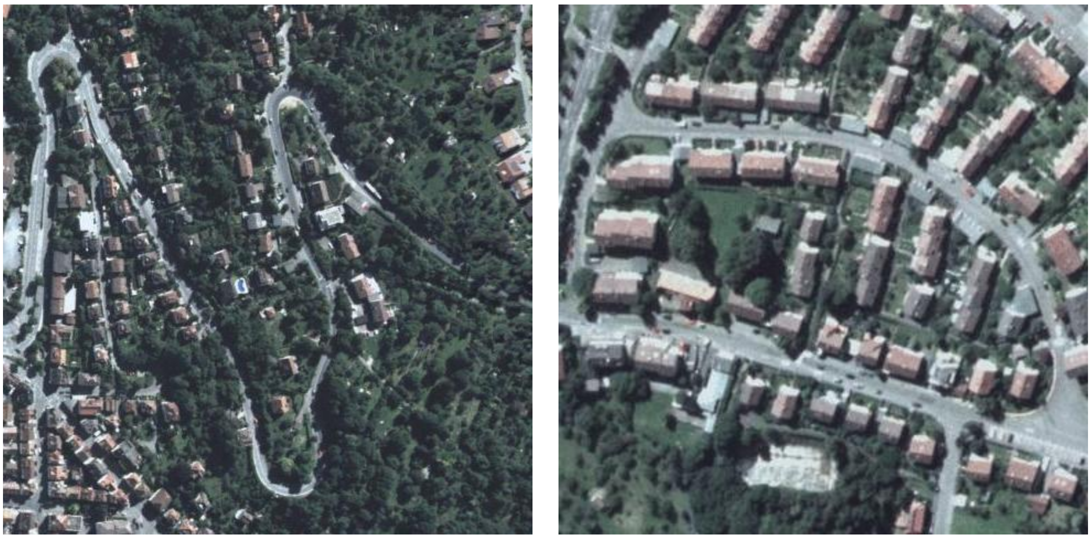
(ZNCC) is used in order to check the confidence of the match. A match is considered reliable if the ZNCC coefficient is larger than a predefined threshold. Dynamic programming can also be applied between pyramid translations, in order to eliminate matches which invalidate the ordering constraint. Finally a sub-pixel disparity map is computed as the last step in the pyramid. This is achieved by fitting a second degree curve to the best match and its neighbors, and finding the sub-pixel disparity where the slope is zero for the quadratic function.

The tracking algorithm we use for the computation of optical flow is the Kanade/Lucas/Tomasi (KLT) tracker. An extended description of the algorithm can be found in [TK91] and [ST94] and therefore we skip the description of this method here. Our experience with different tracker algorithms has shown that the KLT tracker can track feature points with a small error over dozens of frames. In all sequences, when not stated otherwise, a maximal amount of 2000 tracks points were tracked over time. The selection of new features is carried out according to the selection method of the KLT algorithm [ST94], maximizing the quality of tracking. Furthermore, in order to obtain a better distribution of the features over the image, the algorithm constraints new features to have a minimal distance to any other feature in the image.

The first three sequences of images evaluated in this chapter were taken with the research vehicle shown in Figure 8.1.

8.2 Traffic Scenarios

This section shows the application of the ego-motion and Kalman filter methods to two long sequences of images. Aerial views of the path traveled for the two sequences can be seen in Figure 8.2. The frame rate for both sequences was 10 frames per second. The baseline of the stereo camera is 0.35 meters and the images have a standard VGA resolution (640×480 pixels).



(a) *Curves*

(b) *Ring*

Figure 8.2: aerial views of the traveled path for sequences *Curves* and *Ring*.

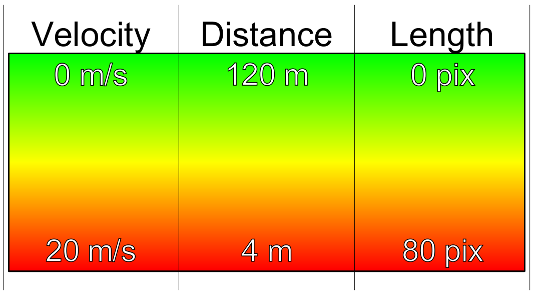


Figure 8.3: Color encoding.

8.2.1 Sequence *Curves*

The sequence *Curves* was taken as the vehicle was traveling the road shown in Figure 8.2(a) at velocities between $20\,km/h$ and $60\,km/h$. The sequence covers a distance of $1.25\,km$ with 1128 stereo images. Ego-motion was computed using only the visual information, i.e. , the inertial sensors of the vehicle were not used for the computation of ego-motion. A Kalman Filter, as described in Chapter 4, is initialized for every new feature and the filter is updated over time as the feature is tracked and a stereo disparity for the feature is computed. This way we obtain an improved position of the point and an estimation of its velocity.

The results for the estimated ego-position of the camera are shown in Figure 8.4, where the ego-position was plotted over the aerial view in order to analyze the accordance of the results with the real path. Some reference points were marked over the map; the sequence starts at reference point 1 and ends at reference point 9. Ego-motion was computed with Weighted Least Squares using a multi-frame estimation of maximal level 5. Although a small deviation from the real path occurs at the end of the sequence, because of error propagation (observe that the last curve with reference points 7 and 8 deviates slightly from the road), the estimated path matches the road almost everywhere, not only in position, but also in orientation.

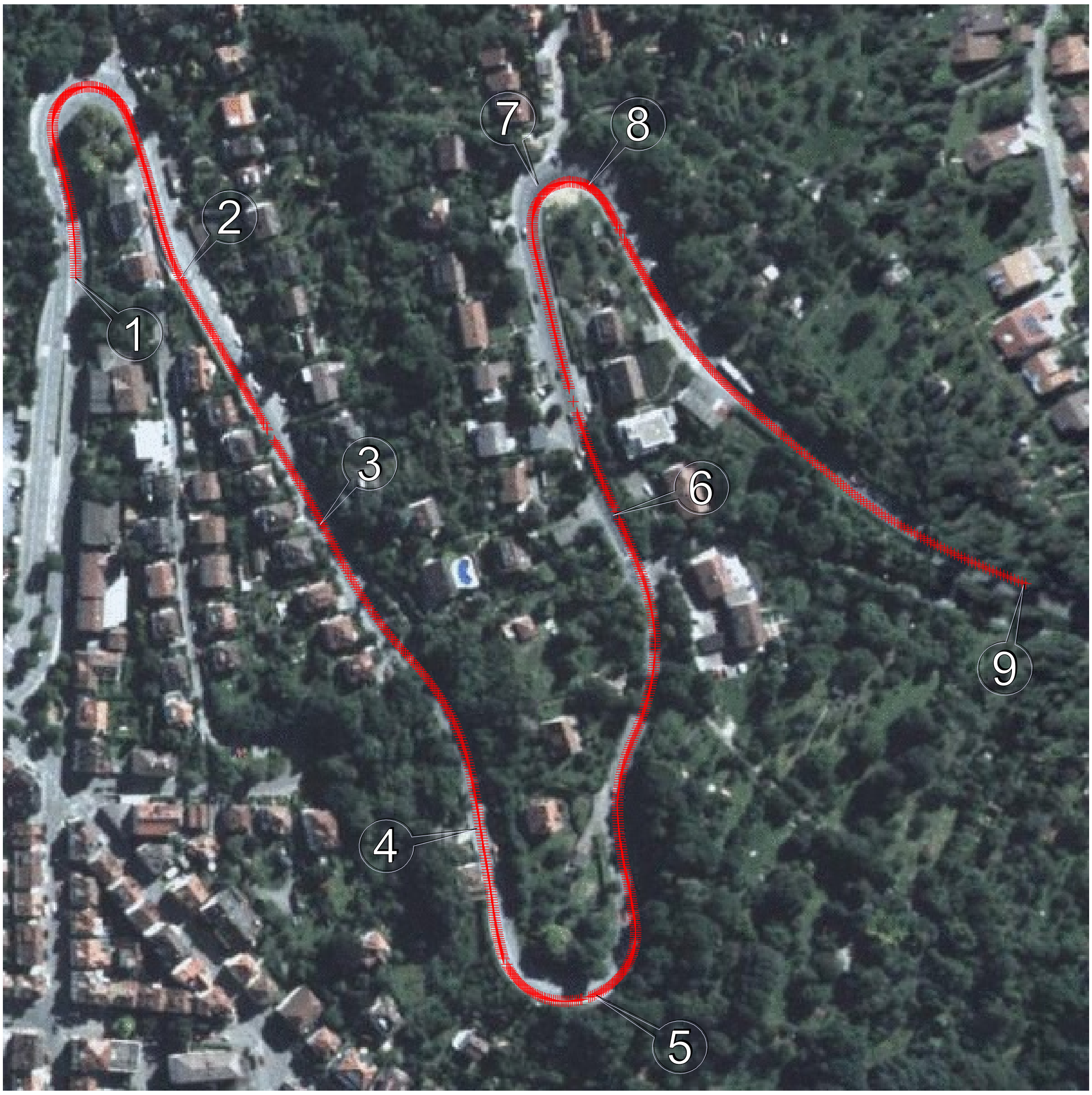


Figure 8.4: Ego-Motion estimation result for the sequence *Curves*.

Figure 8.5 shows snapshots corresponding to the reference points of Figure 8.4. The vectors show the predicted 3D position of each feature in 0.5 seconds back-projected into the image. The color encodes the velocity of the point as Figure 8.3 indicates. Features with an estimated low velocity are shown as green points without an attached vector. Only features which have been tracked at least 2 times are shown, avoiding displaying velocity estimates for false stereo or flow correspondences. Features wrongly tracked tend to disappear in the next two frames (i.e. the tracker is not able to find a correspondence for that feature any more). It can be seen in the snapshots that very few wrong tracked features with an age of 3 images survive. The different coloring of the vectors corresponding to the same object occur mainly because of the different ages of the features. Young features which have not converged, have an estimated velocity smaller than the real one, since the filter is initialized as being static. Those features tends to have a green color in Figure 8.5.

When a feature is incorrectly tracked the feature shows an arrow with a complete wrong direction or length for the corresponding object (an example can be seen in

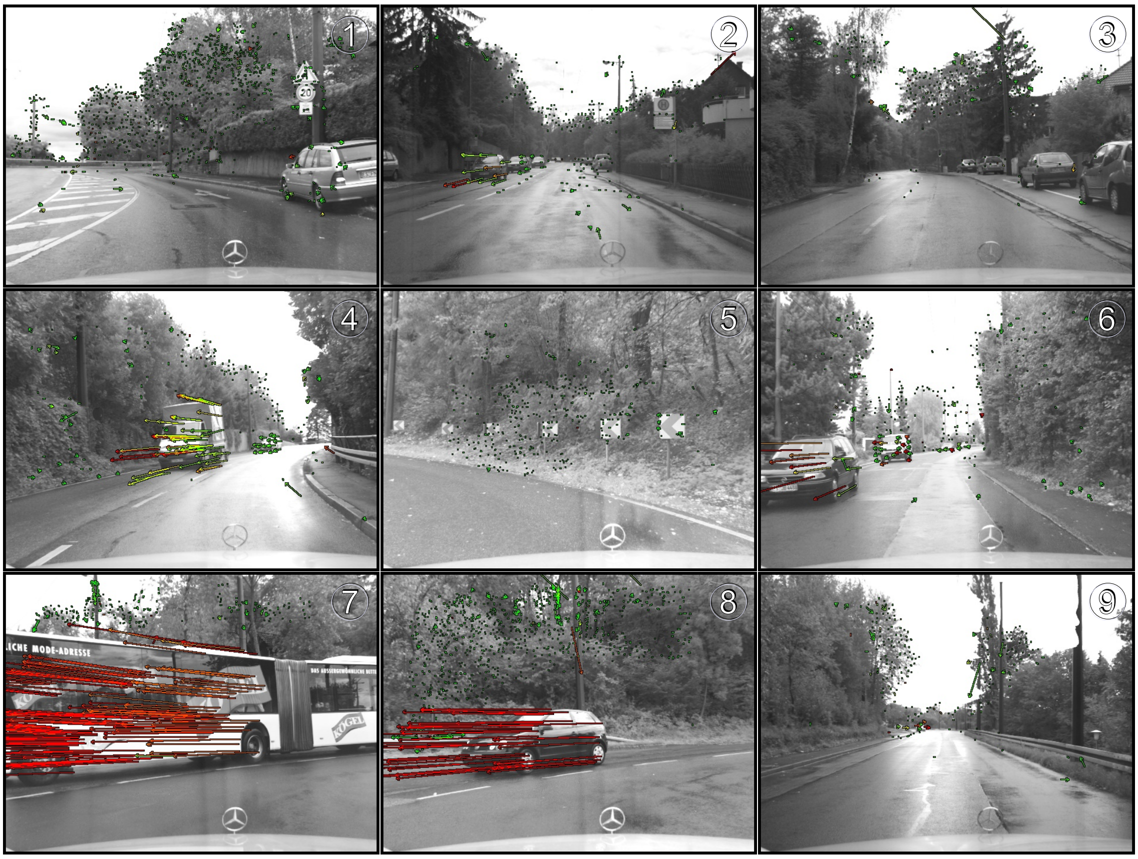


Figure 8.5: Snapshots of the estimated velocity of tracked points for the sequence *Curves*.

snapshot 8 of Figure 8.5 where a feature corresponding to the static background has a large red arrow pointing downwards).

The sequence presents a total of 25 oncoming vehicles, many of them are shown in the snapshots. Even when a very large percentage of the image is occupied with independently moving objects, the algorithm is able to compute robustly the ego-motion of the vehicle. The snapshot 7 shows this case. Ego-motion computation was here possible thanks to the static points of the background.

Figure 8.6 shows some additional information for snapshot number 8. The two top images show optical flow and stereo results. The color encodes the length of the optical flow in left image and the estimated depth of the features in the image at the right (see Figure 8.3). The bottom-left image shows the feature vectors which survived the Smoothness Motion Constraint (i.e. , the inliers). The color encodes the error between prediction and measurement, where green means no error and red means large error. The bottom-right figures shows also the features selected by the Smoothness Motion Constraint but for level 3 of Multi-Step estimation. The last 4 tracked positions for each selected feature are shown. Observe in both cases that all features corresponding to the moving vehicle are avoided and not used for the computation of ego-motion.

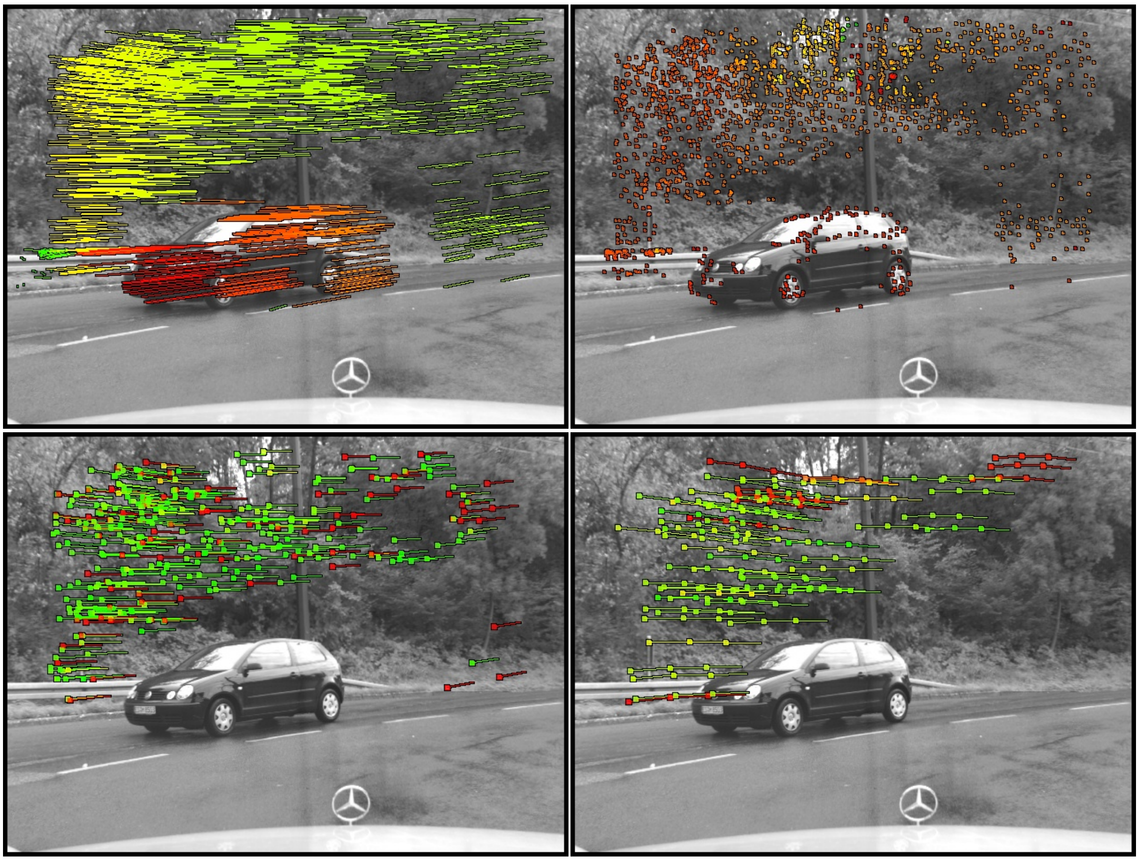


Figure 8.6: Stereo computation, optical flow computation and optical flow vectors used in single and multi-step computation

8.2.2 Sequence *Ring*

The sequence *Ring* corresponds to the road shown in Figure 8.2(b). The velocity of the vehicle varies between $0\,km/h$ and $40\,km/h$. The sequence has a total of 1550 images and covers more than $1.3\,km$. In the sequence, the vehicle travels two times round the block completing $720°$. The amount of features to track was set to 2000. The results of the ego-motion algorithm are shown in Figure 8.7(a), where the ego-position was plotted over the aerial view in order to analyze the accordance of the results with the real path. The first round was plotted in red, while the second round was plotted in green. The speed and yaw rate of the the inertial sensors of the vehicle were used in this sequence, as specified in Section 7.7. Some reference points were marked in the map and Figure 8.7(b) shows the corresponding snapshots. The vehicle started at the position marked by the reference point 1 and finished at reference point 9 after driving twice round the block.

This sequence has only a few independently moving objects, in contrast to the sequence *Curves*. One of them is shown in the snapshot 6 of Figure 8.7(b), where an oncoming vehicle forces the ego-vehicle to stop, since the street is too narrow for the two vehicles. In the next round, the vehicle drives more centered on the street, at the same position as shown in the snapshot 8. Observe that even this difference is reflected in the estimated ego-position, since the first round marked in red in Figure 8.7(a) is displaced a little to the left in comparison to the second round marked in

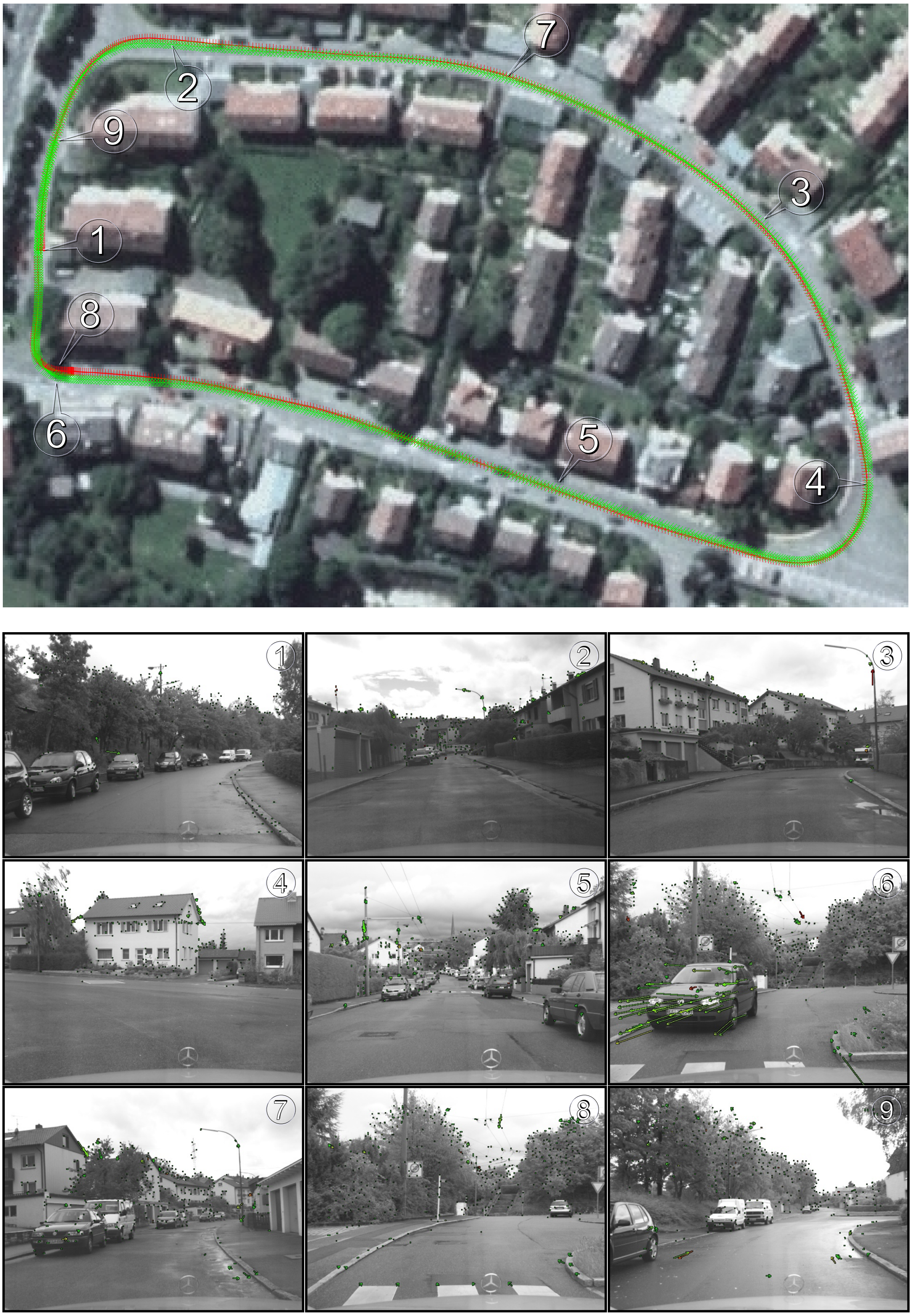


Figure 8.7: Results for the sequence *Ring*.

green (reference points 6 and 8).

8.2.3 Computation Times

8.2.3.1 Absolute Orientation Computation

Optimized implementations of the absolute orientation algorithms of Chapter 5 allow a very fast computation of the optimal rotation and translation between cloud of points. The following table shows the time requirements for each method. All the measurements are in milliseconds. The test were made with a Intel™Pentium™M Processor 2.00 GHz.

Nr Of Points	WLS (quat)	TLS (Weng *et al.*)	TLS Powell
500	0.205	2.491	13.4
1000	0.443	6.924	22.48
1500	0.722	7.304	31.80
2500	1.514	12.841	46.87
5000	4.684	33.155	122.56

Table 8.1: times are measured in milliseconds.

Only time results for the quaternion method are shown, since the SVD and polar decomposition methods have the same computational complexity and the times are equivalent. The closed form solution of Weng *et al.* [WC90] allows an implementation which requires little computational power, as can be seen in the table and makes it a very appealing method for the computation of the TLS. For comparison purposes Equation 5.13 was solved by an iterative method. The average iterations required for obtaining a solution was of 2.04.

8.2.3.2 Full cycle times

The average computation times corresponding to the parameters used in the above two sequences are the following:

Module:	Time [ms]
Ego-Motion:	8.509
Kalman Filter:	9.798
Stereo:	14.643
KLT Tracker:	90.413

8.3 Off-Road and Country Scenarios

Although the application area for which this method was designed for is the field of assistance driver systems in typical traffic situations, it can also be used in other

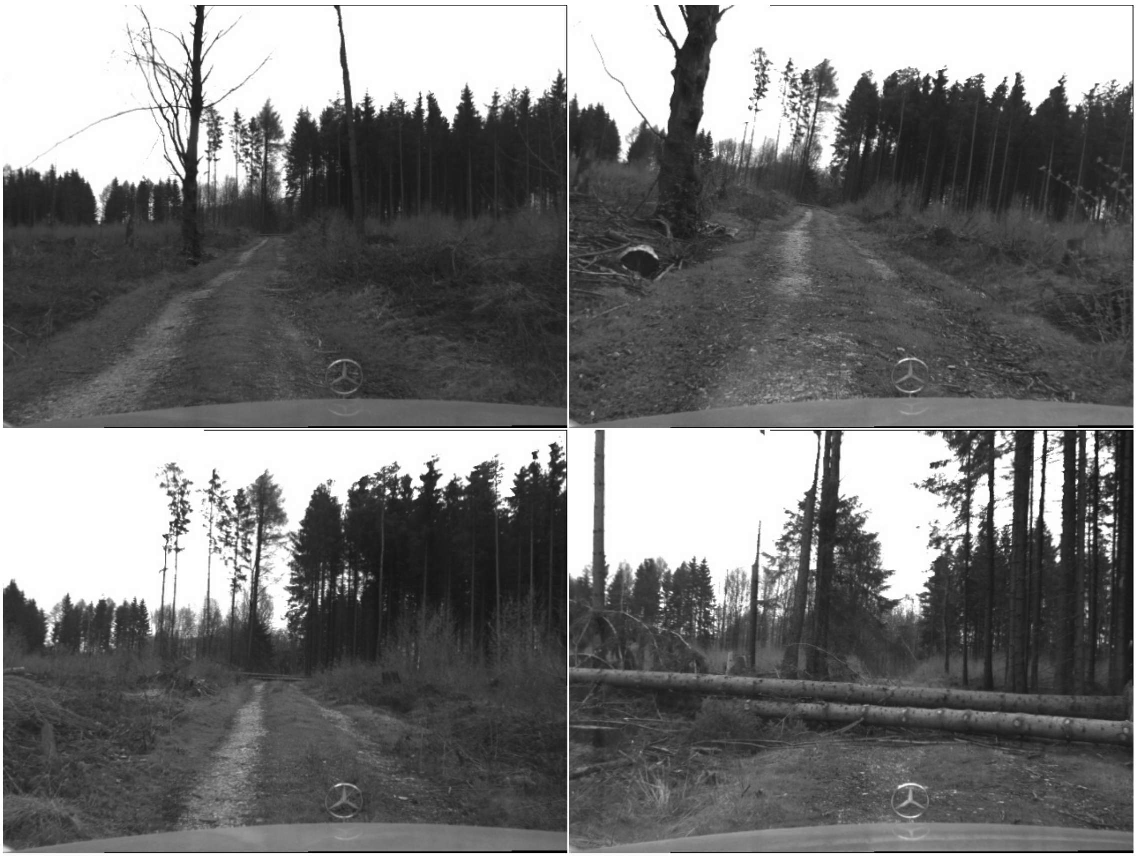


Figure 8.8: Snapshots of the sequence *Forest*.

environments such as country scenarios. The application area can also be used for 3D scene reconstruction, as this section shows.

The differences between a typical downtown driving scenario and that of driving on a forest/field/dessert are given by:

- The type of object expected to be found, e.g. buildings, trees, etc.
- The behavior of the objects, e.g. moving or static.
- The textureness, e.g. low textureness on walls, high textureness on trees or bushes.
- The type of motion expected, e.g. planar, vibrating, etc.
- Expectation of repetitive structures; cities and forest contain many vertical repetitive structures like windows and trunks which might lead to false correspondences; on the other hand a sand desert might contain no vertical structures at all.
- Object materials; in populated areas some surfaces might reflect light rays into the camera, in natural environments such reflections are less probable.

A method might become useless if applied to a different area than that for which it was designed for.

The ego-motion method presented in this dissertation can be applied to different application areas since it does not make any assumptions about the type of motion expected. The main requirement is that the optical flow and stereo algorithm deliver a high enough number of point correspondences. As already shown in the previous chapter, the condition of at least 50% of non-contaminated data is not even required, since the method is able to work relatively well, even when more than half of the data are outliers.

This section shows the ego-motion results with a sequence taken as the vehicle drove on a unpaved roadway in a forest. The road is quite uneven and the vehicle motion is different to that expected on a road. In order to compensate for the rough motion of the vehicle, the frame rate was set to an average of $24\,frames/sec$. Figure 8.8 shows some snapshots of the sequence. Observe that the top-right image shows a rotation of about 9° around the optical axis, which is normally not expected to happen in normal roads. In sequences *Curves* and *Ring* the maximal rotation around the optical axis was about 2.5°. The sequence is composed of 800 stereo images and the vehicle drives at slow speed a distance of approx. 120 meters.

The reconstruction of the scene in this section is achieved by just plotting the 3D points observed through the sequence. Since every observed 3D point is actually measured and estimated in a camera coordinate system (i.e. the camera position is always at the origin), the direct plotting of 3D points will not regenerate the structure of the scene (unless the camera is static). Instead, the clouds of points obtained at each frame are translated and rotated according to the corresponding position and orientation of the camera (i.e. , its ego-motion) obtaining continuity of the scene. The points to plot are not those directly obtained with stereo. Instead, the Kalman filtered 3D positions of the points are the optimal candidates to plot. A point is selected for plotting (together with its grey value) if its estimated velocity is below a threshold. This way, we avoid the plotting of points with large error terms. This basic information might be used by more efficient reconstruction methods (e.g. [KESFK05] [JL03] [GS04] [Mor02]), all of which are out of the scope of this dissertation.

Snapshots of the reconstruction are shown in Figure 8.9. The maximal number of tracks was 3000. Ego-motion was computed with a maximal multi-step integration level of 15. A total of approx. 1.7×10^6 3D points with corresponding grey value were collected trough the sequence. The path traveled by the camera is shown in red.

8.4 Indoor Environment

The main difference between outdoor and indoor environments is the mean and variance distance to the observed objects. In indoor environments both variables are quite a bit smaller than in outdoor environments. For stereo applications this means, mainly, that the noise affecting the 3D position will be smaller. On the other hand, images of indoor environments might contain more and bigger untextured areas, like walls, and therefore less information. As long as the images contain enough texture to allow the computation of optical flow and stereo, the method presented in this dissertation can be used to compute the ego-motion of a freely moving camera in

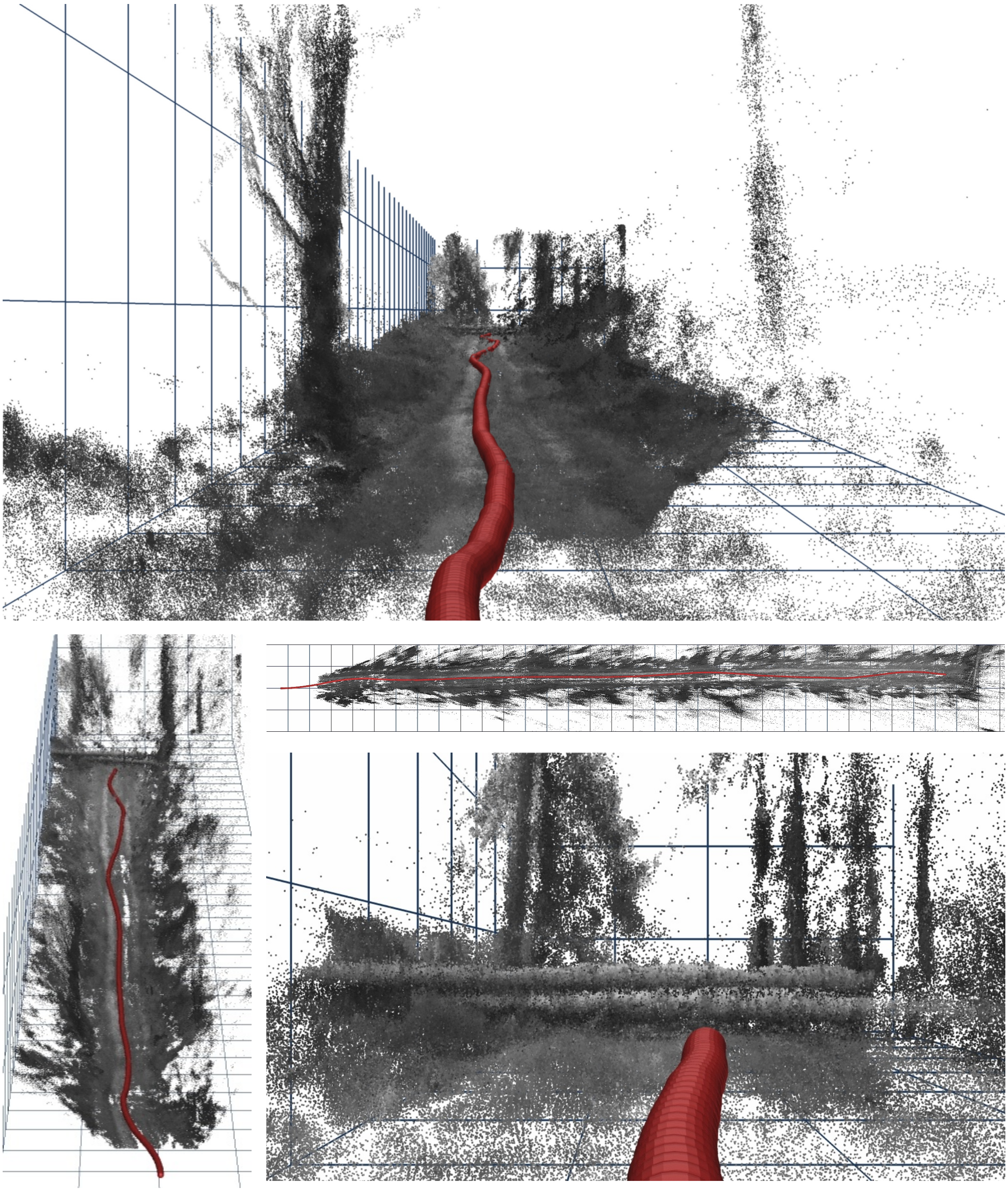


Figure 8.9: scene reconstruction for the sequence *Forest*

indoor environments, as this section shows.

In order to obtain some ground truth information about the motion of the camera, a special sequence was constructed. A normal sequence of about 468 images was recorded in an office while a person was moving. A second sequence was generated from the first sequence by just reversing the playing order of each image. Finally the latter was attached to the former in order to obtained a complete sequence of 936 images. The start and end position of the camera for the whole sequence is exactly the same and therefore the complete integration of motion must be zero at the end of the sequence. The camera configuration has a baseline of $12cm$ and a focal length

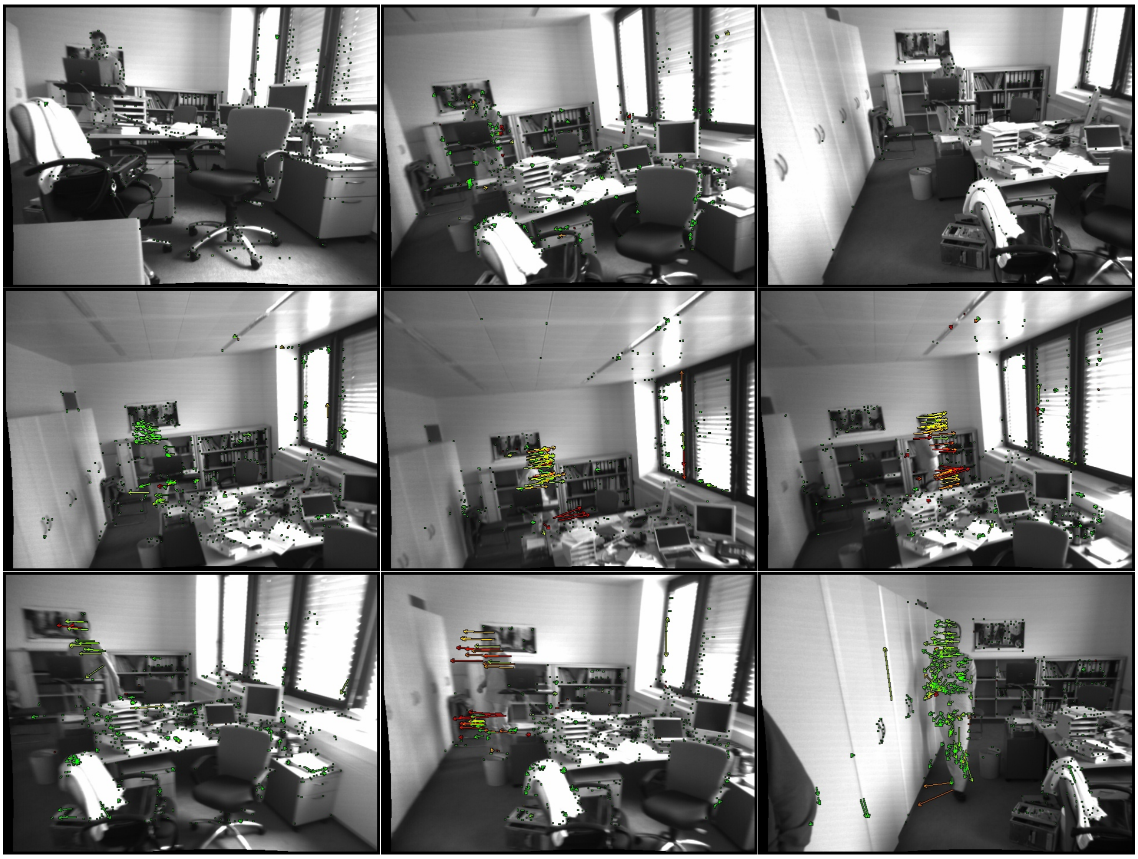
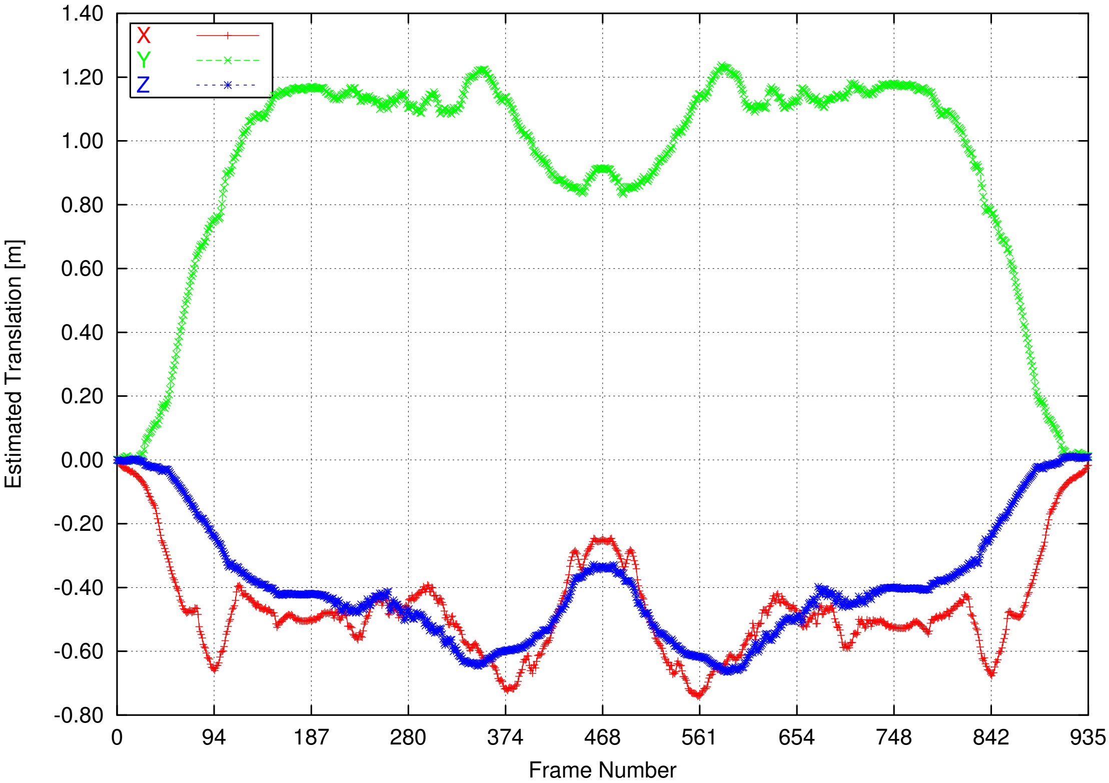


Figure 8.10: Snapshots of the sequence *Office*.

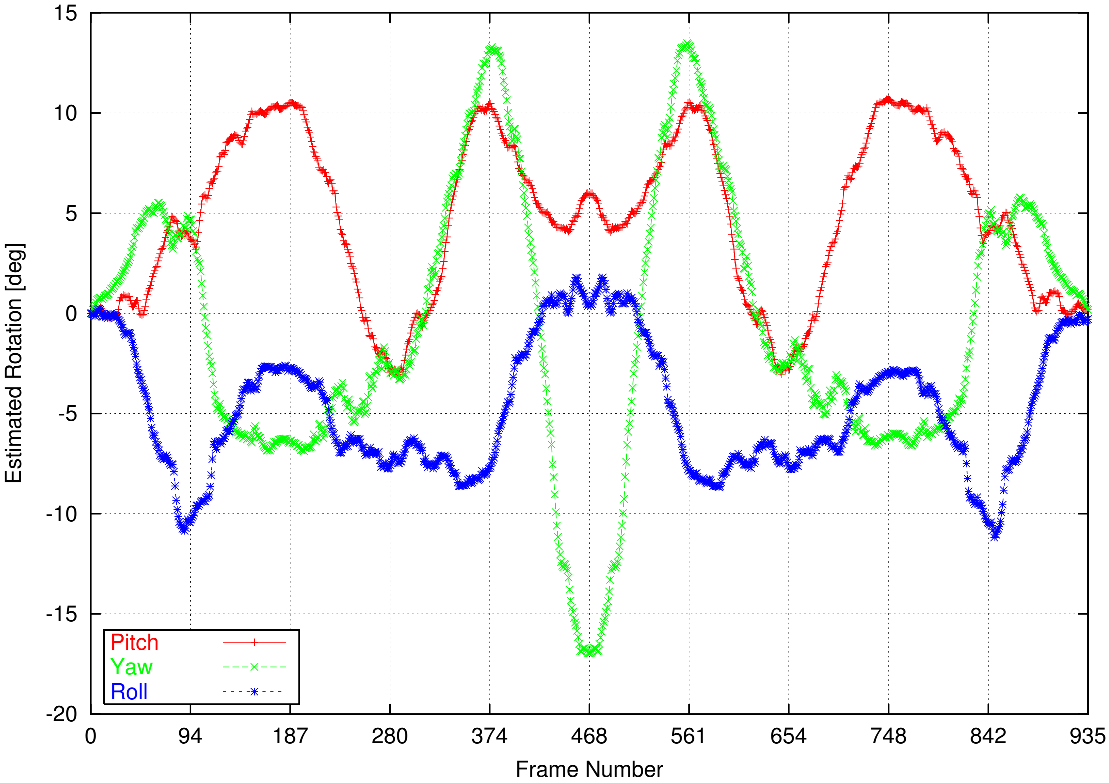
of $0.4cm$. The images have a standard VGA resolution. A maximal integration depth of 80 frames was used for this sequence. Large integration levels are achieved only when the feature can be tracked over so many frames.

Figure 8.10 shows some snapshots of the sequence *Office*. A person holds the camera and moves it to different position and orientations throughout the sequence. The sequence shows a person, who stays still and partially occluded at the beginning of the sequence (first three top images of Figure 8.10. The person then moves to the left, then to the right, once again to the left and finally towards the camera. At this point (last image of Figure 8.10) the sequence is repeated in backward order. In the snapshots, it can be seen that there are some velocity vectors of the points corresponding to the moving person. The color encoding is the same as shown in Figure 8.3, but where red means $1.75\,m/s$. The different colors of the vectors obey principally to the different velocities of the body-parts. Since the scene contain many repetitive structures (e.g. , the folders in the filling cabinet at the background and the blinds at the left), some velocity vectors for static objects are incorrectly estimated, but without major consequences.

Observe that as a consequence of the special construction of the sequence, the results for the total camera motion over time must be symmetric. Figure 8.11 shows the estimation of the total rotation and translation as a function of image number. Observe the high symmetry in the motion estimation for all motion parameters. A consequence of this symmetry is that the motion estimation at the end of the sequence should be the same as in the beginning of the sequence, i.e. zero. Any



(a) Total estimated translation



(b) Total estimated rotation.

Figure 8.11: Results for the sequence *Office*

	Component	**Error**	**Total**
Rotation	Pitch	$0.209064\,°$	$0.4012\,°$
	Yaw	$0.180416\,°$	
	Roll	$-0.291064\,°$	
Translation	X	$-0.016799\,m$	$0.02245\,m$
	Y	$0.01174\,m$	
	Z	$0.009168\,m$	

Table 8.2: Total accumulated motion error for sequence *Office* over 936 images.

deviation from that shows the total accumulated error, which is shown in Table 8.2.

8.5 Pose-Estimation for Crash Test Analysis

Computing the ego-motion of the camera requires the definition of a reference frame. Until now, the reference frame was implicitly given by the static scene (the world), with origin and orientation given by the position of the camera in the first image of the sequence. When the motion to estimate is caused by the change in position and/or orientation of the camera over time, with respect to its reference frame, we refer to it as computing the *ego-motion* of the camera. If, instead, the camera remains static and the motion to estimate is given by the change in position and/or orientation of some observed object, then this is referred to as *object pose-estimation*. Ego-motion and pose estimation are related problems and can be solved using the same mathematical framework (see e.g. [LHM98]). Pose estimation is a problem that has undergone much research in Computer Vision. For an overview of robotic camera pose estimation techniques refer to [GKA98]. Most pose estimation algorithms rely on models of the observed object (see e.g. [RPS05]). This section shows how to perform a model-free pose estimation applied to crash analysis. The results have been initially published in [Geh06].

In our pose estimation scenario, the stereo camera system remains static and the observed object moves. The whole method remains the same, since the relative motion is identical. However, the Smoothness Motion Constraint is not related to the motion of the camera any more, but instead to the observed object. We obtain the initial object position with the initial stereo computation. From there, we transform the motion of the object back into the static reference frame and compare the predicted position via Smoothness Motion Constraint to the measured position. This way, tracks that are not on the observed object can be eliminated for pose estimation.

The main difference with previous applications is the reduced angular resolution of the object. When the region of interest is the whole image, as in the previous examples, the angular resolution is approx. $42° \times 32°$ for a typical image of 640×480 pixels and focal length of $8mm$[1]. On the other hand, in crash analysis the FOV of the object of interest can be as small as $2°$. Investigations on pose estimation of

---

[1] In the sequence *Office* the FOV after rectification was approx. $61° \times 45°$ because the focal length of the camera was $4mm$.

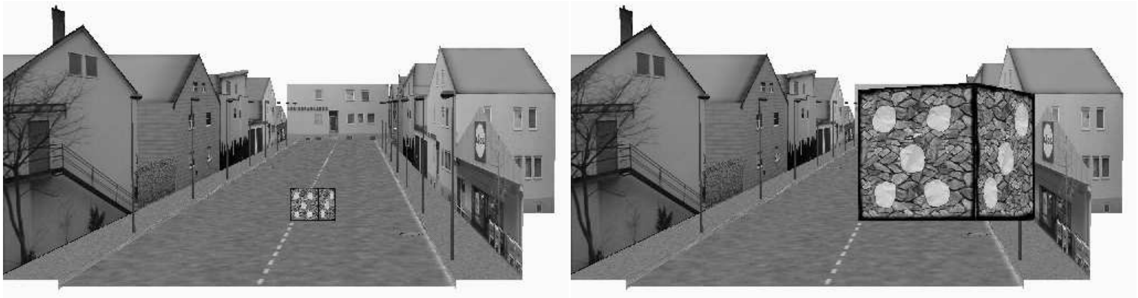


Figure 8.12: Smallest and largest simulated dice of the synthetic sequence.

Object size (10m distance)	$0.25m$ $70[pix]$	$0.50m$ $140[pix]$	$0.75m$ $210[pix]$	$1.00m$ $280[pix]$
Horizontal Angular Resolution	$\approx 2°$	$\approx 4°$	$\approx 6°$	$\approx 8°$
Number of tracks	2000	8000	11000	12000
Mean Error (± std deviation)				
Δ pitch[$°/frame$]	0.04 ± 0.07	0.00 ± 0.02	0.00 ± 0.01	0.00 ± 0.003
Δ yaw[$°/frame$]	0.12 ± 0.25	0.01 ± 0.08	0.01 ± 0.02	0.00 ± 0.02
Δ roll[$°/frame$]	0.01 ± 0.01	0.00 ± 0.003	0.00 ± 0.001	0.00 ± 0.001
integrated pitch[°]	0.52 ± 0.19	0.06 ± 0.09	0.06 ± 0.02	0.01 ± 0.01
slope of integrated yaw[$°/frame$]	0.08 ± 0.30	0.00 ± 0.07	0.00 ± 0.05	0.00 ± 0.03
integrated roll[°]	0.50 ± 0.18	-0.06 ± 0.09	-0.03 ± 0.02	0.03 ± 0.01
Maximum Error				
Δ pitch[$°/frame$]	1.9	0.42	0.19	0.11
Δ yaw[$°/frame$]	4.3	2.9	1.1	0.46
Δ roll[$°/frame$]	0.3	0.13	0.04	0.04
integrated pitch [°]	6.9	1.6	0.4	0.2
integrated yaw [°]	10.0	2.3	0.8	0.4
integrated roll [°]	6.7	0.7	0.38	0.22

Table 8.3: Accuracy of pose estimation for varying object sizes. Motions are described in the camera coordinate system, pitch motion around the horizontal axis, yaw around the vertical axis and roll in the image plane.

small objects covering only a small field of view, are rare in the literature. In order to verify the obtainable accuracy of the algorithm, a simulation scene with a dice of variable size is used. Image dimensions are 1600×1200 pixels (comparable to high speed/high resolution cameras used in crash analysis), and the dice is set at 10m distance (see Figure 8.5). We varied the size of the dice and let it rotate around the vertical axis to obtain a yaw motion of $2°/frame$ from the camera perspective. The obtained errors are listed in Table 8.3, and show that even small objects allow an accurate pose estimation. The maximum errors, including the integrated pose errors over 100 frames, stay within $10°$, the average errors stay well below $1°$. Multi-frame estimation up to 20 frames was used. Other publications rarely show data on pose estimation algorithms for small objects, so no reasonable comparison data can be provided.

We illustrate the performance with real images showing results of an offset crash.

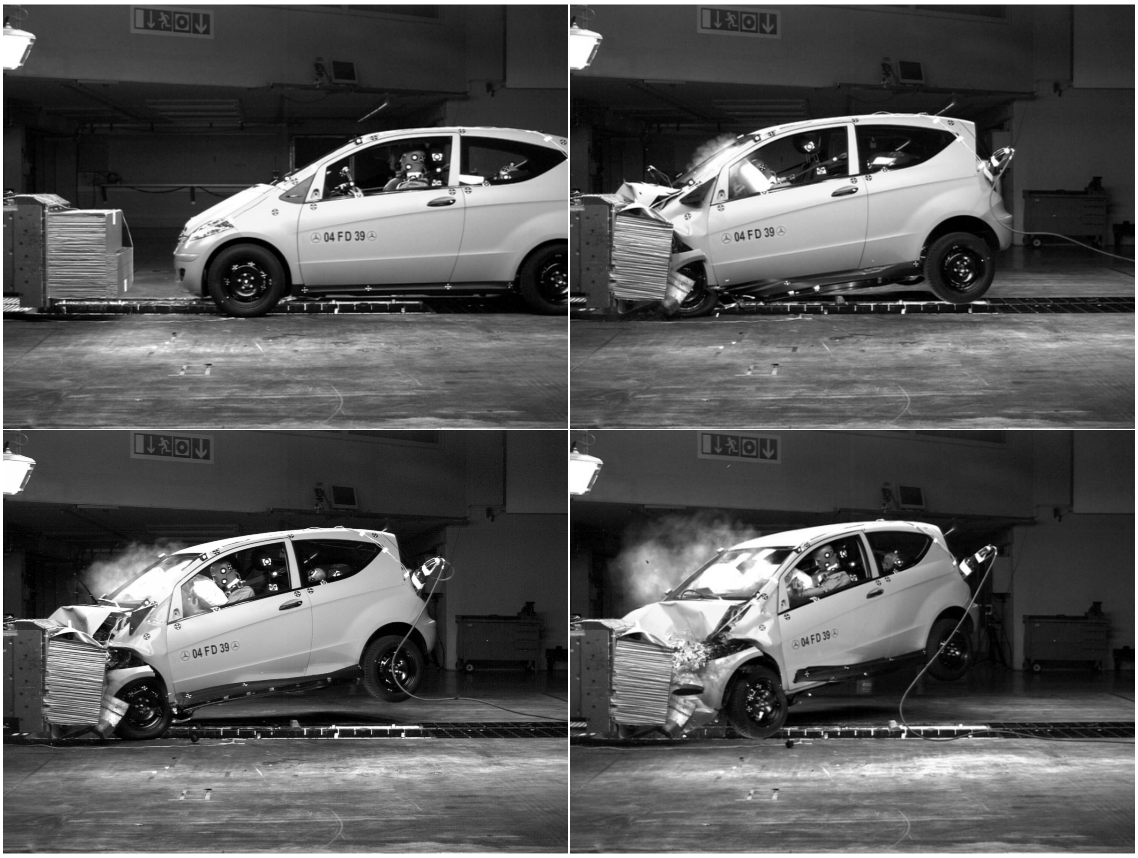


Figure 8.13: Snapshots of the sequence *Crash Test*. Top-Left and bottom-right images show the first and last frame the sequence. The top-right image shows the moment of deepest airbag penetration. The bottom-left image shows the image at which the head is totally visible once again.

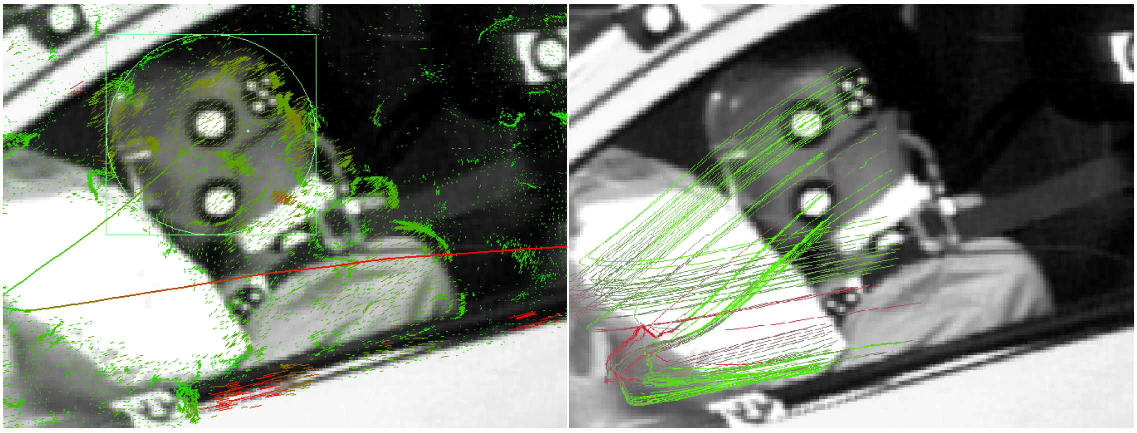
The object of interest is the head of a dummy, which almost disappears in the airbag at one moment in the sequence. Figure 8.5 shows some snapshots of the sequence. The sequence is recorded at $1000Hz$ and consists of 300 images. Here, we used multi-frame estimation up to 100 frames in order to cope with the massive occlusion.

The object of interest is selected manually in the scene. From there on, the object is tracked automatically using the average motion vector of the observed object. To obtain the motion vector, all tracks that originate within the selected region of interest are considered for pose estimation. But only tracks that pass the Smoothness Motion Constraint are used for average motion computation. This way, partial occlusion is handled easily, since these flow vectors are not consistent with the current pose. Therefore, the manual selection does by no means require being precise, as long as the majority of the flow vectors in the selected region corresponds to the object of interest.

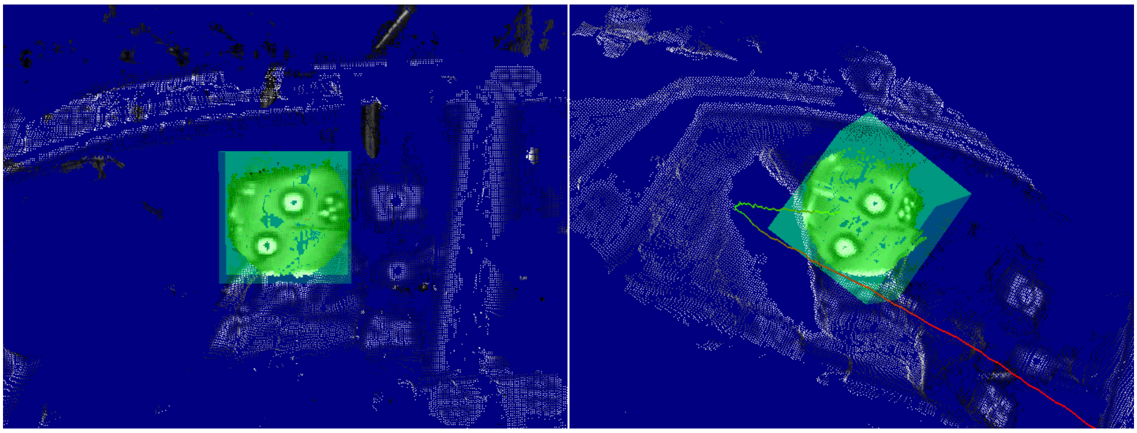
To check the plausibility of the measurements, we also tracked the region of the driver door, specifically the letters. Knowing the vehicle dynamics of offset crashes, we expect a yaw motion away from the barrier and a slight camera roll motion due to jumping. Both behaviors can be clearly seen in the plot of the integrated angles of the estimated poses in Figure 8.14(c). The pitch motion is expected to remain around

$0°$, and the deviations from that illustrate the noise level of the measurements.

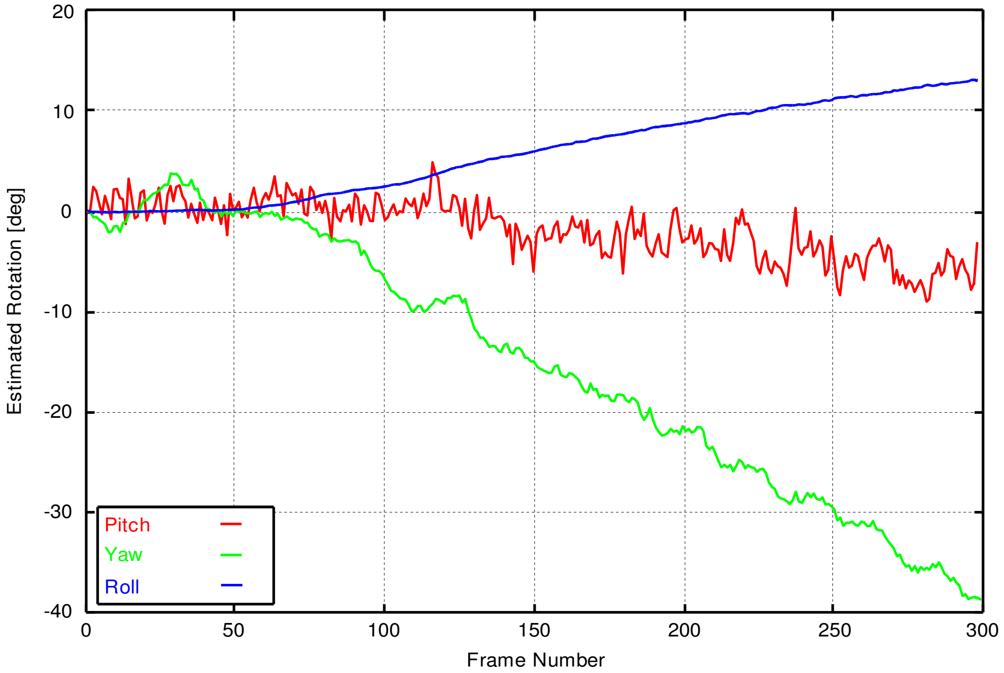
Figure 8.14(a) shows the optical flow and multi-steps tracks for image number 200 of the sequence. In the left image the circle shows the tracked region of interest. Most good tracks stay at the five-dot-marker at the right part of the head that stays visible throughout the sequence. Figure 8.14(b)shows a 3D view of the scene. The position of the camera is based on the obtained pose, and transforms the head to its initial orientation, viewed from $2.5m$ distance. The right image depicts the transformed pose of image 200 after reappearing from the airbag. Note the good agreement to the initial pose shown in the left image.



(a) Optical flow (left image) and tracks of multi-level 100 (right image) in frame 200.



(b) 3D view of the scene



(c) Results of the pose-estimation algorithm.

Figure 8.14: Optical flow, Tracks and aligned 3D Views.

Chapter 9

Conclusions and Outlook

9.1 Summary

This dissertation presents a complete framework for the real time estimation of the scene structure, the detection and estimation of the independently moving points, and the accurate estimation of the 6 degrees of freedom of ego-motion. These are basic components required for building many intelligent automotive applications, which support the driver in traffic situations. The complete framework works exclusively with the information provided by a binocular platform. Furthermore, the presented approach is a passive technique, in the sense that the method does not call for any data acquisition strategy, i.e. the stereo images are just processed as they are provided. This avoid any interference with other vision applications that could make use of the cameras (e.g. lane detection) and which control their functionality.

The proposed framework is divided in three main blocks.

- The *Registration of Image Features* is the block that receives, as input, the rectified images and provides, as output, a list of tracked image features with their corresponding 3D position. The method presented in this dissertation is independent of the actual implementation used for the registration of image features.
- The *Ego-Motion* block carries out four mains steps in an iterative way: motion prediction, application of the Smoothness Motion Constraint, absolute orientation computation, and motion integration. The SMC is a powerful constraint for the rejection of outliers and for the assignment of weights to the measured 3D points. The absolute orientation is computed with a weighted or a total least squares approach in closed form. Each iteration provides a new motion hypothesis, which is integrated into the current motion estimation. We call this approach, Multi-Frame Estimation (MFE), in contrast to the Two-Frame Estimation (TFE), which considers only the current and previous frames for the computation of ego-motion.
- The *Kalman Filter* represents the third block. By means of Kalman Filters, an iterative refinement of the 3D point position is achieved. The motion of a static

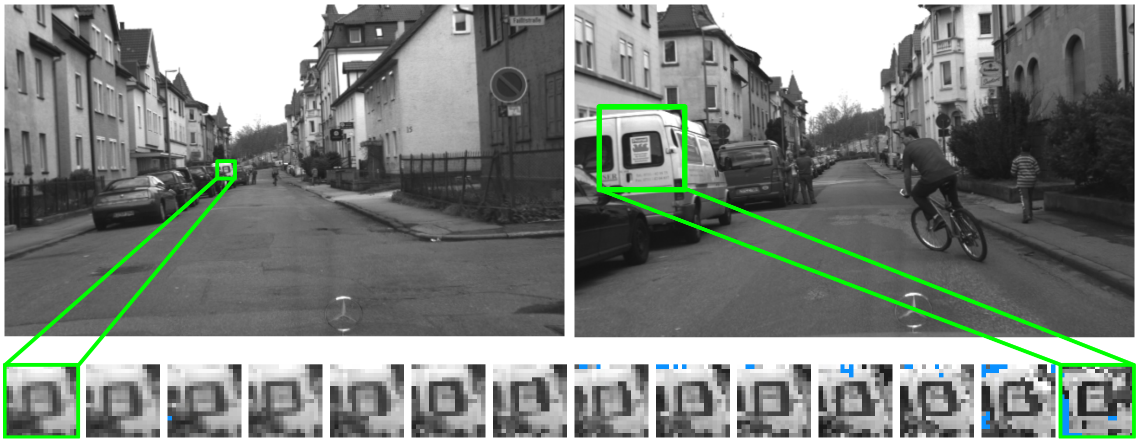


Figure 9.1: Homography computed with ego-motion parameters.

> world point is given by the rigid body motion of the background. Any deviation from this movement implies an independent motion, which is estimated by the differentiation of the point position over time.

The second block, i.e. the computation of the ego-motion of the camera constitutes the core of this dissertation. A variety of simulations have validated the implementation of the proposed methods. The experimental results have shown the robustness and accuracy of the method using large sequences of stereo images.

9.2 Conclusions

The robust and real-time computation of the 6 degrees of freedom of ego-motion from stereo sequences has shown to be possible. This is true if the image sequence fulfills some minimal requirements, as presented in Section 3.2. When this requirements are not fulfilled, the inertial sensor data, if available, can be used to increase robustness. The method has shown to be adaptable to many scenarios and application areas. The experimental results have shown the potential applicability of this method to the Simultaneous Localization and Mapping problem, by building a 3D cartography of the scene. Indoor applications have been tested with positive results. Finally, a model-free pose-estimation of small objects has also been performed.

9.3 Future Work

The information provided by this framework can be used by advanced intelligent vehicle driving assistance systems, like obstacle detection, autonomous navigation and self localization. Furthermore, within the framework much can still be done, especially referring to the block *Registration of Image Features*. In the dissertation, this block was not addressed in detail, allowing some flexibility in the actual implementation. Nevertheless this block takes more than $2/3$ of the total time requirements. Improving the quality of the output of this block will automatically improve the rest of the estimates. Decreasing its computation time will be beneficial too, not only because of the larger resolution time, but also because the correspondence problem becomes simpler for smaller motions.

One way to increase speed and quality is using the prediction stage of the Kalman Filter. If the prediction happens to be accurate, the time required for finding a match reduces and the probability of finding a correct match increases. Another way to increase accuracy is by integrating the ego-motion information in the matching process. Thanks to the computation of ego-motion the complete transformation of the scene between any two times is known. This allows for the computation of the homography of image patches. An example is given in Figure 9.1, where the first and last images of a sequence are shown. A patch image was tracked over time and the corresponding homography was computed with the ego-motion information at regular intervals. A tracking method can make use of this information to increase the performance of the algorithm.

9.4 Outlook

The computational power of computers increases day by day and new and better computer vision algorithms and methods are continuously developed. Open competitions like the DARPA Grand Challenge (http://www.grandchallenge.org) and research initiatives like INVENT (http://invent-online.de) are helping to accelerate research and to promote the development of more advanced systems. Some applications, like autonomous navigation, have already proven to work well in controlled environments (http://www.darpa.mil/grandchallenge05/GC05winnerv2.pdf). Much more is expected in the near future.

Intelligent vehicles of the future will contain a redundant set of sensors and sources of information; such as cameras, infra-red-sensors, lasers, radars, geographical data bases and digital maps, GPS data and even the exchange of data between different vehicles. The redundancy obtained by multiple sensors provides a considerable improvement in overall data quality, compared to the information from individual sensors. A camera is the sensor which provides more information and its use has not been completely exploited. In this thesis, the use of sequences of images to estimate ego-motion information has been proven.

Appendix A

Quaternions

Quaternions are denoted by the set $\mathbb{H}$ and are a non-commutative extension of the set of complex numbers $\mathbb{C}$. Being the set $\mathbb{C} = \mathbb{R} + \mathbb{R}i$, the set of quaternions is defined as

$$\mathbb{H} = \mathbb{C} + \mathbb{C}j\,, \quad \text{with } j^2 = -1 \text{ and } ij = -ji \tag{A.1}$$

By expanding Equation A.1, a quaternion $\mathring{\boldsymbol{q}}$ is defined as

$$\begin{aligned} \mathring{\boldsymbol{q}} &= (q_0 + q_x i) + (q_y + q_z i)j \\ &= q_0 + q_x i + q_y j + q_z ij\ , \quad q_0, q_x, q_y, q_z \in \mathbb{R} \end{aligned} \tag{A.2}$$

The complex term ij is usually replaced with k for simplicity. The set of quaternions $\mathbb{H}$ is isomorphic to the Euclidean set $\mathbb{E}^4$ and has a structure of non-commutative algebra. Observing that

$$\begin{array}{lllllllll} i^2 &=& -1, & j^2 &=& -1, & k^2 &=& -1 \\ ij &=& k, & jk &=& i, & ik &=& -j \end{array} \tag{A.3}$$

and the anti-commutative properties

$$ji = -k, \quad kj = -i, \quad ki = j \tag{A.4}$$

the product between a quaternion $\mathring{\boldsymbol{p}} = p_0 + p_x i + p_y j + p_z k$ and the quaternion $\mathring{\boldsymbol{q}}$ is

$$\begin{aligned} \mathring{\boldsymbol{p}}\mathring{\boldsymbol{q}} &= (p_0q_0 - p_xq_x - p_yq_y - p_zq_z) \\ &+ (p_0q_x + p_xq_0 + p_yq_z - p_zq_y)i \\ &+ (p_0q_y - p_xq_z + p_yq_0 + p_zq_x)j \\ &+ (p_0q_z + p_xq_y - p_yq_x + p_zq_0)k. \end{aligned}$$

The product $\mathring{\boldsymbol{q}}\mathring{\boldsymbol{p}}$ has similar structure but with some signs changed, i.e.

$$\begin{aligned}\mathring{\boldsymbol{q}}\mathring{\boldsymbol{p}} &= (p_0q_0 - p_xq_x - p_yq_y - p_zq_z)\\ &+ (p_0q_x + p_xq_0 - p_yq_z + p_zq_y)i\\ &+ (p_0q_y + p_xq_z + p_yq_0 - p_zq_x)j\\ &+ (p_0q_z - p_xq_y + p_yq_x + p_zq_0)k\end{aligned}$$

and therefore, in general, the product of two quaternions is not commutative, i.e. $\mathring{\boldsymbol{p}}\mathring{\boldsymbol{q}} \neq \mathring{\boldsymbol{q}}\mathring{\boldsymbol{p}}$. When considering a quaternion as a 4 dimensional column vector, the product of quaternions can be expressed as the product of a 4×4 matrix and the vector, i.e.

$$\mathring{\boldsymbol{p}}\mathring{\boldsymbol{q}} = \begin{bmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & -p_z & p_y \\ p_y & p_z & p_0 & -p_x \\ p_z & -p_y & p_x & p_0 \end{bmatrix} \mathring{\boldsymbol{q}} = \boldsymbol{P}\mathring{\boldsymbol{q}}. \tag{A.5}$$

One can choose also to associate the matrix to the second quaternion, i.e.

$$\mathring{\boldsymbol{q}}\mathring{\boldsymbol{p}} = \begin{bmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & p_z & -p_y \\ p_y & -p_z & p_0 & p_x \\ p_z & p_y & -p_x & p_0 \end{bmatrix} \mathring{\boldsymbol{q}} = \bar{\boldsymbol{P}}\mathring{\boldsymbol{q}}. \tag{A.6}$$

noting that $\boldsymbol{P}$ and $\bar{\boldsymbol{P}}$ differ only on lower-right hand 3×3 sub-matrix, which is transposed.

The dot products of quaternions is the scalar obtained as the sum of the products of the corresponding components:

$$\mathring{\boldsymbol{p}} \cdot \mathring{\boldsymbol{q}} = p_0q_0 + p_xq_x + p_yq_y + p_zq_z \tag{A.7}$$

and the dot product of a quaternion with itself defines the square of the norm of the associated vector, i.e.

$$||\mathring{\boldsymbol{q}}||^2 = \mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}} \tag{A.8}$$

Observing the anti-commutative rules of Equation A.4 the conjugate of a quaternion is obtained negating its imaginary part, i.e.

$$\mathring{\boldsymbol{q}}^* = q_0 - q_xi - q_yj - q_zk \tag{A.9}$$

In order to obtain the associated matrices of the conjugate of a quaternion, the associated matrices for the original quaternion must be transposed. Observe also that the associated matrices are orthogonal and therefore $\boldsymbol{P}\boldsymbol{P}^T = \mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}}\, I_{4\times4}$ (and also $\bar{\boldsymbol{P}}\bar{\boldsymbol{P}}^T = \mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}}\, I_{4\times4}$). The product of a quaternion with its conjugate is real:

$$\mathring{\boldsymbol{q}}\mathring{\boldsymbol{q}}^* = q_0^2 + q_x^2 + q_y^2 + q_z^2 = \mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}} \tag{A.10}$$

The inverse of a nonzero quaternion is defined as

$$\mathring{\boldsymbol{q}}^{-1} = \frac{\mathring{\boldsymbol{q}}^*}{\mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}}} \tag{A.11}$$

A subgroup of $\mathbb{H}$ is the set of *unit quaternions*, i.e. the group of quaternions with unity norm. They are denoted with $\mathbb{S}^3$ and defined as

$$\mathbb{S}^3 = \{\mathring{\boldsymbol{q}} \in \mathbb{H} \mid \|q\|^2 = 1\} \tag{A.12}$$

The set of all unit quaternions is simply the unit sphere in $\mathbb{R}^4$. The rotation group $SO(3)$ is embedded into the group $\mathbb{S}^3$. To see this, first observe that dot products are preserved, i.e.

$$\begin{aligned}(\mathring{\boldsymbol{q}}\mathring{\boldsymbol{p}}) \cdot (\mathring{\boldsymbol{q}}\mathring{\boldsymbol{r}}) &= (\boldsymbol{Q}\mathring{\boldsymbol{p}}) \cdot (\boldsymbol{Q}\mathring{\boldsymbol{r}}) \\ &= (\boldsymbol{Q}\mathring{\boldsymbol{p}})^T(\boldsymbol{Q}\mathring{\boldsymbol{r}}) \\ &= \mathring{\boldsymbol{p}}^T\boldsymbol{Q}^T\boldsymbol{Q}\mathring{\boldsymbol{r}} \\ &= \mathring{\boldsymbol{p}}^T(\mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}})\boldsymbol{I}_{4\times4}\mathring{\boldsymbol{r}}\end{aligned} \tag{A.13}$$

since the matrices associated with quaternions are orthogonal. Being $\mathring{\boldsymbol{q}}$ a unit quaternion we can conclude that

$$(\mathring{\boldsymbol{q}}\mathring{\boldsymbol{p}}) \cdot (\mathring{\boldsymbol{q}}\mathring{\boldsymbol{r}}) = \mathring{\boldsymbol{p}} \cdot \mathring{\boldsymbol{r}} \tag{A.14}$$

A quaternion can be seen as the pair $(q_0, \boldsymbol{q})$ where $q_0 \in \mathbb{R}$ and $\boldsymbol{q} \in \mathbb{E}^3$, i.e. a pure imaginary quaternion represents a vector in Euclidean space, i.e. $\mathring{\boldsymbol{q}} = 0 + q_x + q_y + q_z$ represents the vector $\boldsymbol{q} = q_x + q_y + q_z$ (at the same time, scalars in $\mathbb{R}$ can be represented by real quaternions with zero imaginary part). The rotation of a pure imaginary quaternion $\mathring{\boldsymbol{r}}$ with a rotation unit quaternion $\mathring{\boldsymbol{q}}$ is obtained as the composite product:

$$\mathring{\boldsymbol{r}}' = \mathring{\boldsymbol{q}}\mathring{\boldsymbol{r}}\mathring{\boldsymbol{q}}^* \tag{A.15}$$

where $\mathring{\boldsymbol{r}}'$ is also pure imaginary. Expanding Equation A.15 we obtain:

$$\mathring{\boldsymbol{r}}' = \mathring{\boldsymbol{q}}\mathring{\boldsymbol{r}}\mathring{\boldsymbol{q}}^* = (\boldsymbol{Q}\mathring{\boldsymbol{r}})\mathring{\boldsymbol{q}}^* = \bar{\boldsymbol{Q}}^T\boldsymbol{Q}\mathring{\boldsymbol{r}} \tag{A.16}$$

$\bar{\boldsymbol{Q}}^T\boldsymbol{Q}$ being equal to

$$\bar{\boldsymbol{Q}}^T\boldsymbol{Q} = \begin{bmatrix} \mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}} & 0 & 0 & 0 \\ 0 & (q_0^2 + q_x^2 - q_y^2 - q_z^2) & 2(q_xq_y - q_0q_z) & 2(q_xq_z + q_0q_y) \\ 0 & 2(q_yq_x + q_0q_z) & (q_0^2 - q_x^2 + q_y^2 - q_z^2) & 2(q_yq_z - q_0q_x) \\ 0 & 2(q_zq_x + q_0q_y) & 2(q_zq_y - q_0q_x) & (q_0^2 - q_x^2 - q_y^2 + q_z^2) \end{bmatrix}.$$

Since $\mathring{\boldsymbol{q}}$ is a unit quaternion, $\boldsymbol{Q}$ and $\bar{\boldsymbol{Q}}$ are both orthonormal and $\mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{q}} = 1$ and the lower right hand 3×3 sub-matrix is also orthonormal. It can also be demonstrated that the determinant of the sub-matrix is $+1$. Indeed the sub-matrix corresponds to the orthonormal rotation matrix of the group $SO(3)$.

The inverse process, i.e. obtaining a unit rotation quaternion from a orthonormal rotation matrix can be accomplished using the following combinations of the diagonal elements of Equation A.17:

$$\begin{aligned} 1 + r_{11} + r_{22} + r_{33} &= 4q_0^2 \\ 1 + r_{11} - r_{22} - r_{33} &= 4q_x^2 \\ 1 - r_{11} + r_{22} - r_{33} &= 4q_y^2 \\ 1 - r_{11} - r_{22} + r_{33} &= 4q_z^2 \end{aligned}$$

where the first quaternion component to extract is the largest (in order to ensure numerical accuracy). The other 3 components of the quaternion are obtained from the following off-diagonal sums and differences:

$$\begin{aligned} r_{13} - r_{31} &= 4q_0 q_y & \qquad r_{13} + r_{31} &= 4q_z q_x \\ r_{21} - r_{12} &= 4q_0 q_z & \qquad r_{21} + r_{12} &= 4q_x q_y \\ r_{32} - r_{23} &= 4q_0 q_x & \qquad r_{32} + r_{23} &= 4q_y q_z \end{aligned}$$

replacing the first component found and solving for remaining components.

Unit quaternions can be used to interpolate rotations. The direct linear interpolation of unit quaternions would not produce a rotation since quaternions live on the unit hyper-sphere. Instead, a spherical linear interpolation is performed. The spherical linear interpolation is usually defined as a function Slerp$(\mathring{\boldsymbol{q}}, \mathring{\boldsymbol{p}}, t)$ where $0 \leq t \leq 1$ and $\mathring{\boldsymbol{q}}$ and $\mathring{\boldsymbol{p}}$ are the quaternions to be interpolated. The spherical linear interpolation is defined as

$$\text{Slerp}(\mathring{\boldsymbol{q}}, \mathring{\boldsymbol{p}}, t) = \frac{\sin((1-t)\Omega)}{\sin \Omega}\mathring{\boldsymbol{q}} + \frac{\sin(t\Omega)}{\sin \Omega}\mathring{\boldsymbol{p}} \tag{A.17}$$

where $\Omega = \arccos(\mathring{\boldsymbol{q}} \cdot \mathring{\boldsymbol{p}})$ is half the angle between $\mathring{\boldsymbol{q}}$ and $\mathring{\boldsymbol{p}}$. From Equation A.17 it can be easily be seen that

$$\text{Slerp}(\mathring{\boldsymbol{q}}, \mathring{\boldsymbol{p}}, 0) = \mathring{\boldsymbol{q}} \quad \text{and} \quad \text{Slerp}(\mathring{\boldsymbol{q}}, \mathring{\boldsymbol{p}}, 1) = \mathring{\boldsymbol{p}} \tag{A.18}$$

and the corresponding symmetry of interpolation

$$\text{Slerp}(\mathring{\boldsymbol{q}}, \mathring{\boldsymbol{p}}, t) = \text{Slerp}(\mathring{\boldsymbol{p}}, \mathring{\boldsymbol{q}}, 1-t). \tag{A.19}$$

Appendix B

Abbreviations and Symbols

The following symbols, abbreviations and notation conventions are used throughout the thesis.

Notation

Scalars are denoted by Roman or Greek lower-case letters in italic form, e.g. a, β.
Vectors are denoted with lower case bold-italic symbols, e.g. $\boldsymbol{p}$, $\boldsymbol{z}$.
Matrices are denoted with upper case bold-italic symbols, e.g. $\boldsymbol{M}$, $\boldsymbol{W}$.
A vector joining two points is denoted with right arrow over the points, e.g. $\overrightarrow{PQ}$.
An homogeneous vector is denoted as a vector with a point at the left, e.g. ${}^{\cdot}\boldsymbol{p}$, ${}^{\cdot}\boldsymbol{m}$.
The real unavailable value of a variable is denoted with a bar, e.g. $\bar{\boldsymbol{p}}$, $\bar{\boldsymbol{R}}$.
Functions are denoted according to the type of variable returned, e.g. $\boldsymbol{g}(a)$ or $f(\boldsymbol{s})$.
Quaternions are denoted with a ring, e.g. $\mathring{\boldsymbol{p}}$, $\mathring{\boldsymbol{q}}$, $\mathring{\boldsymbol{r}}$.
The transposed of a matrix $\boldsymbol{A}$ is denoted as $\boldsymbol{A}^T$.
The identity $n \times n$ matrix is denoted as $\boldsymbol{I}_{n\times n}$.
The null $n \times m$ matrix is denoted as $\mathbf{0}_{n\times m}$.

Abbreviations

CRLB	Cramér-Rao Lower Bound.
CV	Computer Vision.
EKF	Extended Kalman Filter.
KF	Kalman Filter.
LS	Least Squares.
LMedS	Least Median of Squares
MFE	Multi-Frame Estimation.
RANSAC	Random Sample Consensus.
SAD	Singular of Absolute Differences.
SLAM	Simultaneous Localization and Mapping.

SMC	Smoothness Motion Constraint.
SSD	Singular of Squared Differences.
SVD	Singular Value Decomposition.
TFE	Two-Frame Estimation.
TLS	Total Least Squares.
p.d.f.	Probability density function.
d.o.f.	Degrees of freedom.

Symbols

$\boldsymbol{R}$	Rotation Matrix.
$\boldsymbol{t}$	Translation Vector.
$\boldsymbol{M}$	Homogeneous Transformation Matrix.
t_k	Cycle time k.

Bibliography

[Adm05] Federal Highway Administration. Nationwide differential global positioning system program fact sheet. http://www.tfhrc.gov/its/ndgps/02072.htm. Technical report, U.S. Department of Transportation, December 2005.

[Adm06] Federal Aviation Administration. Wide area augmentation system fact sheets. http://gps.faa.gov/Library/waas-f-text.htm. Technical report, Federal Aviation Administration, June 2006.

[AG92] P. Aschwanden and W. Guggenbühl. Experimental results from a comparative study on correlation-type registration algorithms. *Robust Computer Vision*, pages 268–289., 1992.

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):698–700, March 1987.

[Ait35] A. C. Aitken. On least squares and linear combination of observations. In *Proceedings of the Royal Society of Edinburgh*, volume 55, pages 42–68, March 1935.

[AKI05] M. Agrawal, K. Konolige, and L. Iocchi. Real-time detection of independent motion using stereo. In *IEEE Workshop on Motion and Video Computing (WACV/MOTION 2005)*, pages 672–677, Breckenridge, CO, USA, January 2005.

[AO97] Antonis A. Argyros and Stelios C. Orphanoudakis. Independent 3d motion detection based on depth elimination in normal flow fields. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pages 672–677, November 1997.

[ATB94] Y. Altunbasak, A.M. Tekalp, and G. Bozdagi. Simultaneous motion-disparity estimation and segmentation from stereo. In *Proceedings 1994 International Conference on Image Processing*, pages 73–77, Austin, Texas, USA, 1994.

[ATB95] Y. Altunbasak, A.M. Tekalp, and G. Bozdagi. Simultaneous stereo-motion fusion and 3d motion tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP-95.*, volume 4, pages 2277–2280, May 1995.

[ATO98] Antonis A. Argyros, P.E. Trahanias, and Stelios C. Orphanoudakis. Robust regression for the detection of independent 3d motion by a binocular observer. *Journal of Real Time Imaging*, 4:125–141, April 1998.

[Bad02] Hernán Badino. Comparative research of stereo vision methods. Technical report, DaimlerChrysler AG, Stuttgart, Germany, 2002.

[Bad04] Hernán Badino. A robust approach for ego-motion estimation using a mobile stereo platform. In 1^{st} *Internation Workshop on Complex Motion (IWCM04)*, Günzgburg, Germany, October 12 - 14 2004.

[Bad05] Hernán Badino. Volume of uncertainty in 3d-recovery with stereo. Technical report, DaimlerChrysler AG, Stuttgart, Germany, 2005.

[BE95] J. L. Barron and R. Eagleson. Binocular estimation of motion and structure from long sequences using optical flow without correspondence. In *International Conference on Image Processing*, pages 193–196, 1995.

[BFRG06] Hernán Badino, Uwe Franke, Clemens Rabe, and Stefan Gehrig. Stereovision based detection of moving objects under strong camera motion. In *International Conference on Computer Vision Theory and Applications VISAPP*, Setúbal, Portugal, February 2006.

[BH81] Anna R. Bruss and Berthold K.P. Horn. Passive navigation. In *A.I. Memo No. 662, MIT*, November 1981.

[Bie77] Gerald J. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Dover Pubn Inc, 1977.

[Bla72] J. A. R. Blais. Three-dimensional similarity. *The Canadian Surveyor*, 1:71–76, 1972.

[BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[BNB04] E. Bandari, I. Nesnas, and M. Bajracharya. Efficient calculation of absolute orientation with outlier rejection. In *Spatiotemporal Image Processing. One Day BMVA Symposium at the Royal Statistical Society*, London, UK, March 2004.

[Bou00] J. Y. Bouguet. Camera Calibration Toolbox For Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc, 2000.

[BS94] K. Brammer and G. Siffling. *Kalman-Bucy Filter: deterministische Beobachtung und stochastische Filterung*. Oldenbourg, 1994.

[BSL93] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation And Tracking: Principles, Techniques and Software*. Artech House, Inc., 1993.

[BSM01] Alireza Behrad, Ali Shahrokni, and Seyed Ahmad Motamedi. A robust vision-based moving target detection and tracking system. In *Proceeding of Image and Vision Computing conference (IVCNZ2001), University of Otago, Dunedin, New Zealand.*, November 2001.

[Car91] S. Carlsson. Recursive estimation of ego-motion and scene structure from a moving platform. In *Proc. 7:th Scand. Conf. on Image Analysis*, pages 958–965, Aalborg, 1991.

[CYSZ99] Tang CY, Hung YP, Shih SW, and Chen Z. A 3d feature-based tracker for multiple object tracking. In *Proceedings of the National Science Council, ROC(A)*, volume 23-1, pages 151–168, 1999.

[Dav02] P. L. Davies. Statistical procedures and robust statistics. Technical Report 54-02, Fachbereich Mathematik und Informatik, Universität Essen, Essen, Germany, November 2002.

[DD01] D. Demirdjian and T. Darrel. Motion estimation from disparity images. Technical Report AI Memo 2001-009, MIT Artificial Intelligence Laboratory, May 2001.

[DD02] D. Demirdjian and T. Darrel. Using multiple-hypothesis disparity maps and image velocity for 3-d motion estimation. *International Journal of Computer Vision*, 47(1-3):219–228, April 2002.

[DH00] D. Demirdjian and R. Horaud. Motion-egomotion discrimination and motion segmentation from image pair streams. In *Computer Vision, Graphics and Image Processing*, volume 78(1), pages 53–68, April 2000.

[DHS02] Thao Dang, Christian Hoffmann, and Christoph Stiller. Fusing optical flow and stereo disparity for object tracking. In *Proceedings of the IEEE V. International Conference on Intelligent Transportation Systems*, pages 112–117, Singapore, September 2002.

[DN93] K. Daniilidis and H.-H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 188–193, New York, NY, June 15-17, 1993.

[DNC$^+$01] M. Dissanayaka, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building problem. *IEEE Transactions On Robotics and Automation*, 17(3):229–241, June 2001.

[EBW04] R. S. J. Estépar, A. Brun, and C.-F. Westin. Robust generalized total least squares iterative closest point registration. In *Seventh International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'04*, Rennes - Saint Malo, France, September 2004.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FH83] O. D. Faugeras and M. Hebert. A 3-d recognition and positioning algorithm using geometric matching between primitive surfaces. In *International Joint Conference on Articial Intelligence*, volume 8, pages 996–1002, August 1983.

[FL01] Olivier Faugeras and Quang-Tuan Luong. *The Geometry of Multiple Images*. The MIT Press, 2001.

[Föl94] Otto Föllinger. *Regelungstechnik: Einführung in die Methoden und ihre Anwendung*. Hüthig, 1994.

[FP03] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[Fra00] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *IEEE Conference on Intelligent Vehicles*, Dearborn, October 2000.

[FRBG05] U. Franke, C. Rabe, H. Badino, and S. Gehrig. 6d-vision: Fusion of stereo and motion for robust environment perception. In *DAGM '05*, Vienna, October 2005.

[GA93] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice*. Prentice Hall, 1^{st} edition, 1993.

[Geh00] S. Gehrig. *Design, Simulation and Implementation of a Vision-Based Vehicle-Following System*. PhD thesis, Eberhard-Karls-Universität, Tübingen, 2000.

[Geh06] Stefan K. Gehrig. Accurate and model-free pose estimation of small objects for crash video analysis. In *Proc. British Machine Vision Conference, BMVC06*, Edinburgh, September 2006.

[Ger99] Neil Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[GKA98] B. Grinstead, A. Koschan, and M. A. Abidi. A comparison of pose estimation techniques: Hardware vs. video. In *SPIE Unmanned Ground Vehicle Technology*, pages 166–173, Orlando, FL, 1998.

[GN98] Joshua Gluckman and Shree K. Nayar. Ego-motion and omnidirectional cameras. In *Sixth International Conference on Computer Vision*, pages 999–1005, Bombay, India, January 1998.

[GS04] M. A. Garcia and A. Solanas. 3d simultaneous localization and modeling from stereo vision. In *IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[HC97] P. K. Ho and R. Chung. Stereo-motion that complements stereo and motion analyses. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pages 213–218, Washington, DC, USA, 1997.

[Hei02] Stefan Heinrich. Fast obstacle detection using flow/detph constraint. In *IEEE Intelligent Vehicle Symposium (IV'2002)*, Versailles, France, June 2002.

[HHN88] Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonomal matrices. In *Journal of the Optical Society of America A*, volume 5(4), pages 1127–1135, July 1988.

[HIG02] Heiko Hirschmüller, Peter R. Innocent, and Jon Garibaldi. Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In *7th International Conference on Control, Automation, Robotics and Vision*, Singapore, 2 - 5 December 2002.

[HJL$^+$87] Robert M. Haralick, Hyonam Joo, Chung-Nan Lee, Xinhua Zhuang, V. G. Vaidya, and M. Kim. Pose estimation from corresponding point data. *IEEE Computer Society Workshop on Computer Vision*, pages 258–263, November 1987.

[HJL$^+$89] Robert M. Haralick, Hyonam Joo, Chung-Nan Lee, Xinhua Zhuang, Vinay G. Vaidya, and Man Bae Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446, November 1989.

[Hor87] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. In *Journal of the Optical Society of America A*, volume 4(4), pages 629–642, April 1987.

[HP96] A. Y.-K. Ho and T.-C. Pong. Cooperative fusion of stereo and motion. In *Pattern Recognition*, volume 29(1), pages 121–130, 1996.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4'th Alvey Vision Conference*, pages 147–151, Manchester, June 1988.

[HS97] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[JB96] Javier González Jiménez and Anibal Ollero Baturone. Estimación de la posición de un robot móvil. In *Automática*, number 29 in 4, pages 3–18, 1996.

[JH92] A. Jepson and D. Heeger. Linear subspace methods for recovering translational direction. Technical Report RBCV-TR-92-40, University of Toronto, Departament of Computer Science, Toronto, Ontario, 1992.

[JL03] Il-Kyun Jung and Simon Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *9th IEEE International Conference on Computer Vision (ICCV'2003)*, pages 946–951, Nice, France, October 2003.

[JT86] Michael Jenkin and John K. Tsotsos. Applying temporal constraints to the dynamic stereo problem. *Computer Vision, Graphics and Image Processing*, 33(1):16–32, 1986.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[Kan93] Kenichi Kanatani. 3-d interpretation of optical flow by renormalization. *International Journal of Computer Vision*, 11(3):267–282, 1993.

[Kan94] Kenichi Kanatani. Analysis of 3-d rotation fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):543–549, May 1994.

[Kel94] Alonzo Kelly. A 3d state space formulation of a navigation kalman filter for autonomous vehicles. Technical Report CMU-RI-TR-94-19, The Robotics Institute, Carnegie Mellon University, Pittsburgh, May 1994.

[KESFK05] Reinhard Koch, Jan-Friso Evers-Senne, Jan-Michael Frahm, and Kevin Koeser. 3d reconstruction and rendering from image sequences. In *WIAMIS 2005*, Montreux, Switzerland, April 2005.

[KK95] P.J. Kellman and M.K. Kaiser. Extracting object motion during observer motion: Combining constraints from optic flow and binocular disparity. *JOSA-A*, 12(3):623–625, March 1995.

[KK98] Yoshio Kimura and Hitoshi Kondo. A note on the generalised gauss-markov theorem on least squares estimation. In *Studies in Regional Science*, volume 28-2, pages 67–75, December 1998.

[LD93] Lingxiao Li and James H. Duncan. 3-d translational motion and structure from binocular image flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):657–667, July 1993.

[LEF95] A. Lorusso, D. Eggert, and R. B. Fisher. A comparison of four algorithms for estimating 3-d rigid transformations. In *Proc. British Machine Vision Conference, BMVC95*, Birmingham, 1995.

[Lev44] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2(7):164–168, 1944.

[LHFP80] H. C. Longuet-Higgins, F.R.S., and K. Prazdny. The interpretation of a moving retinal image. In *Proc. of the Royal Soc. London*, volume 208. B, pages 385–397, July 1980.

[LHM98] C. P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 22(6):610–622, 1998.

[LK81] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th IJCAI*, pages 674–679, Vancouver, Canada, December 1981.

[LK91] Sukhan Lee and Youngchul Kay. A kalman filter approach for accurate 3-d motion estimation from a sequence of stereo images. *Computer Vision, Graphics and Image Processing: Image Understading*, 54(2):244–258, 1991.

[LMT^{+}85] Mobile Robot Laboratory, Hans Moravec, Chuck Thorpe, Gregg Podnar, and Patrick Muir. Autonomous mobile robots, annual report. Technical Report CMU-RI-TR-86-04, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 1985.

[Lof90] Otmar Loffeld. *Estimationstheorie, Bd.2, Anwendungen, Kalman-Filter (Sondereinband)*. Oldenbourg, 1990.

[LS93] J. Li and R. Skerjanc. Stereo and motion correspondence in a sequence of stereo images. In *Signal Processing: Image Communication*, volume 5(4), pages 305–318, 1993.

[Lux00] Slobodanka Lux. Schätzung der Position Stationärer Objekte mit Hilfe eines aktiven Sehesystems. Master's thesis, FH Stuttgart, 2000.

[Mat92] L. Matthies. Toward stochastic modeling of obstacle detectability in passive stereo range imagery. In *In Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 765–768, Champaign, IL (USA), December 1992.

[May79] Peter S. Maybeck. *Stochastic Models, Estimation, and Control, Volume 1*. Academic Press, Inc, 1979.

[MB00] N. Molton and M. Brady. Practical structure and motion from stereo when motion is unconstrained. *International Journal of Computer Vision*, 39(1):5–23, August 2000.

[MD02] Louis-Philippe Morency and Trevor Darrell. Stereo tracking using icp and normal flow constraint. In *Proceedings of International Conference on Pattern Recognition*, Quebec, Canada, August 2002.

[MG03] Louis-Philippe Morency and Rakesh Gupta. Robust real-time egomotion from stereo images. In *Proceedings International Conference on Image Processing, 2003.*, 2003.

[Mil97] Steven Mills. Stereo-motion analysis of image sequences. In *Proceedings of the first joint Australia and New Zealand conference on Digital Image and Vision Computing: Techniques and Applications, DICTA'97 / IVCNZ'97, Albany, Auckland, NZ*, December 1997.

[MKS89] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

[MLG00] Anthony Mallet, Simon Lacroix, and Laurent Gallo. Position estimation in outdoor environments using pixel tracking and stereovision. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000*, pages 3519–3524, San Francisco, April 2000.

[MM96] Martin C. Martin and Hans Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[MM98] Matthias Mühlich and Rudolf Mester. The role of total least squares in motion analysis. In *Proc. European Conference on Computer Vision*, june 1998.

[MM99] Bogdan Matei and Peter Meer. Optimal rigid motion estimation and performance evaluation with bootstrap. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 339–347, September 1999.

[Mor80] Hans Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover (Doctoral Dissertation))*. PhD thesis, Stanford University, September 1980.

[Mor02] H. Moravec. Robust navigation by probabilistic volumetric sensing. Technical Report http:// www.frc.ri.cmu.edu / hpm / project.archive / robot.papers / 2002 / ARPA.MARS / Report.0202.html, Carnegie Mellon University, February 2002.

[MS87] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. In *IEEE Journal of Robotics and Automation*, volume RA-3(3), pages 239–248, June 1987.

[MSS99] R. Mandelbaum, G. Slagian, and H. Sawhney. Correlation-based estimation of ego-motion and structure from motion and stereo. In *Proceedings of the Internation Conference on Computer Vision (ICCV'98)*, pages 544–550, Kerkyra, Greece, September 1999.

[MT84] L. H. Matthies and Chuck Thorpe. Experience with visual robot navigation. In *Proceedings of IEEE Oceans-84*, volume 16, pages 594–597, September 1984.

[Müh05] Matthias Mühlich. *Estimation in Projective Spaces and Applications in Computer Vision*. Johann Wolfgang Goethe-Universität, Ph.D. Thesis, 2005.

[Müh06] Matthias Mühlich. The absolute orientation problem revisited. Technical Report IC-TR-D-0601, Johann Wolfgang Goethe-Universität, Frankfurt am Main, Germany, February 2006.

[NH87] Shahriar Negahdaripour and Berthold K. P. Horn. Direct passive navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):168–176, January 1987.

[NNB04] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, 2004.

[NNB06] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, January 2006.

[OK98] N. Ohta and K. Kanatani. Optimal estimation of three-dimensional rotation and reliability evaluation. In *Proc. 5th Euro. Conf. Computer Vision (ECCV'98)*, volume 1, pages 175–187, Freiburg, Germany, June 1998.

[OMSM03] Clark F. Olson, Larry H. Matthies, Marcel Schoppers, and Mark W. Maimone. Rover navigation using stereo ego-motion. In *Robotics and Autonomous Systems*, volume 43(4), pages 215–229, June 2003.

[Pra80] K. Prazdny. Egomotion and relative depth map from optical flow. In *Biological Cybernetics*, volume 36, pages 87–102, 1980.

[RA90] Jeffrey J. Rodríguez and J. K. Aggarwal. Stochastic analysis of stereo quantization error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):467–470, May 1990.

[Rab05] Clemens Rabe. Detektion von Hindernissen vor Fahrzeugen durch Bewegungsanalyse. Master's thesis, Fachhochschule Würzburg-Schweinfurt, 2005.

[RBE87] Patrick Rives, E. Breuil, and Bernard Espiau. Recursive estimation of 3d features using optical flow and camera motion. In *Intelligent Autonomous Systems, An International Conference*, pages 522–532. North-Holland, 1987.

[RPS05] B. Rosenhahn, C. Perwass, and G. Sommer. Pose estimation of 3d free-form contours. *International Journal of Computer Vision*, 62(3):267–289, 2005.

[San73] Fernando Sansò. An exact solution of the roto-translational problem. *Photogrammetria*, 29:203–216, 1973.

[SC70] Peter H. Schönemann and Robert M. Carrol. Fitting one matrix to another under choice of a central dilation and a rigid motion. *Psychometrika*, 35(2), June 1970.

[Sim06] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006.

[SK99] T. Suzuki and T. Kanade. Measurements of vehicle motion using optical flow. In *IEEE Conference on Intelligent Transportation Systems, Tokyo, Japan*, October 1999.

[SKLL05] N. Sünderhauf, K. Konolidge, T. Lemaire, and S. Lacroix. Comparison of stereovision odometry approaches. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2005*, 2005.

[SMS00] G. P. Stein, O. Mano, and A. Shashua. A robust method for computing vehicle ego-motion. In *IEEE Intelligent Vehicles Symposium (IV2000), Dearborn, MI*, October 2000.

[SMS05] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Bias reduction filter convergence for long range stereo. In *12th International Symposium of Robotics Research (ISRR'05)*, October 2005.

[SSH01] Alexander Suppes, Franke Suhling, and Michael Hötter. Robust obstacle detection from stereoscopic image sequences using kalman filtering. In 23^{rd} *DAGM Symposium, Munich, Germany*, pages 385–391, September 2001.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, June 1994.

[SWB04] R. Sekuler, S. N. J. Watamaniuk, and R. Blake. *Stevens' Handbook of Experimental Psychology*, volume 1, Sensation and Perception, chapter 4, Perception of Visual Motion. Wiley, 3^{rd} edition, January 2004.

[TK91] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method – 3. detection and tracking of point features. Technical Report CMU-CS-91-132, School of CS – CMU, April 1991.

[TK92] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method – full report on the orthographic case. Technical Report CMU-CS-92-104, School of CS – CMU, March 1992.

[TM04] A. Talukder and L. Matthies. Real-time detection of moving objects from moving vehicles using dense stereo and optical flow. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 315–320, Sendai, Japan, September 2004.

[TMM85] Chuck Thorpe, L. Matthies, and Hans Moravec. Experiments and thoughts on visual navigation. In *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 830 – 835, March 1985.

[TS93] C. Tomasi and J. Shi. Direction of heading from image deformations. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 422–427, April 1993.

[TTH96] T. Tian, C. Tomasi, and D. Heeger. Comparison of approaches to egomotion computation. In *Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 315–320, San Francisco, Ca., June 1996.

[Ume91] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[VSVS02] Raquel Frizera Vasallo, José Santos-Victor, and Hans Jörg Schneebeli. A general approach for egomotion estimation with omnidirectional images. In *Proceedings of the Workshop on Omni-directional Vision (held with ECCV'02)*, Copenhagen, Denmark, 2002.

[WB99] John Williams and Mohammed Bennamoun. Multiple view 3d registration: A review and a new technique. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 497–502, October 1999.

[WC90] Juyang Weng and Paul Cohen. Robust motion and structure estimation using stereo vision. In *Proceedings of the First International Conference on Robust Computer Vision*, pages 367–388, Seattle, Washington, USA, October 1990.

[WD86] A. M. Waxman and J. H. Duncan. Binocular image flows: Steps toward stereo-motion fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):715–729, November 1986.

[WK04] Felix Woelk and Reinhard Koch. Robust monocular detection of independent motion by a moving observer. In 1^{st} *International Workshop on Complex Motion (IWCM'04)*, Lecture Notes in Computer Science, Günzburg, Germany, October 2004. Springer.

[WS91] Michael W. Walker and Lejun Shao. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54(3):358–367, November 1991.

[WW03] A. Würz-Wessel. *Free-formed Surface Mirrors in Computer Vision Systems*. PhD thesis, Eberhard-Karls-Universität, Tübingen, 2003.

[YC94] Y. S. Yao and R. Chellappa. Dynamic feature point tracking in an image sequence. In *Proceedings 1994 International Conference on Pattern Recognition*, volume A, pages 654–657, Jerusalem, Israel, October 1994.

[YCS$^+$95] Hung YP, Tang CY, Shih SW, Chen Z, and Lin WS. A 3d predictive visual tracker for tracking multiple moving objects with a stereo vision system. In *Lecture Notes in Computer Science*, volume 1024, pages 25–32, 1995.

[YSJS04] Ma Y., Soatto S., Kosecká J., and Sastry S.S. *An Invitation to 3-D Vision. From Images to Geometric Models*, volume 26 of *Interdisciplinary Applied Mathematics*. Springer, 2004.

[ZF91] Z. Zhang and O. D. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo images. Technical Report RR-1438, Inria, July 1991.

[Zha99] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, pages 666–673, 1999.

LEBENSLAUF

ZUR PERSON

Hernán Badino
geboren am 20.05.1977 in Oncativo
Córdoba, Argentinien

Sindelfinger Strasse 118
Maichingen - Sindelfingen
D71069, Deutschland
E-mail: hernan.badino@gmail.com
Homepage: http://www.lelaps.de/

SCHULBILDUNG UND STUDIUM

Goethe Universität, Frankfurt am Main, Deutschland

Doktorand, Januar 2003 - Mai 2005
Thema der Dissertation: "Binocular Ego-Motion Estimation for Automotive Applications"
Betreuern: Prof. Dr. Rudolf Mester, Goethe Universität und Dr. Uwe Franke, Daimler AG.

National Technological University, Córdoba, Argentinien

Abschluss als Diplom-Ingenieur, Februar 2002.

Hipólito Yrigoyen College, Comodoro Rivadavia, Chubut, Argentinien

Sekundarschullabschluss, Dezember 1994.

ARBEITSERFAHRUNG

Goethe Universität, Frankfurt am Main, Deutschland.

Wissenschaftlicher Mitarbeiter **Oktober 2006 - z.Z.**

in der Forschungsgruppe „Visuelle Sensorik und Informationsverarbeitung".

Daimler Chrysler A.G., Stuttgart, Deutschland.

Doktorand **Januar 2003 - Mai, 2006**

Forschung und Implementierung von Echtzeitalgorithmen für visuelle Fahrerassistenzsysteme.

Wissenschaftlicher Mitarbeiter **Februar - Dezember, 2002**

Forschung und Implementierung von Echtzeitalgorithmen für Stereosehen.

Praktikum **September, 1999 - Februar, 2000**

Programmierer von Stereoalgorithmen.

Oracle Argentina S.A., Buenos Aires, Argentina.

Praktikum **Januar - März, 1998**

PL/SQL Programmierer für Datenbankensysteme.