Motion and Structure Estimation From Video

Johan Hedborg
Dedicated to Ida and Edvin, my family
Abstract

Digital camera equipped cell phones were introduced in Japan in 2001, they quickly became popular and by 2003 outsold the entire stand-alone digital camera market. In 2010 sales passed one billion units and the market is still growing. Another trend is the rising popularity of smartphones which has led to a rapid development of the processing power on a phone, and many units sold today bear close resemblance to a personal computer. The combination of a powerful processor and a camera which is easily carried in your pocket, opens up a large field of interesting computer vision applications.

The core contribution of this thesis is the development of methods that allow an imaging device such as the cell phone camera to estimates its own motion and to capture the observed scene structure. One of the main focuses of this thesis is real-time performance, where a real-time constraint does not only result in shorter processing times, but also allows for user interaction.

In computer vision, structure from motion refers to the process of estimating camera motion and 3D structure by exploring the motion in the image plane caused by the moving camera. This thesis presents several methods for estimating camera motion. Given the assumption that a set of images has known camera poses associated to them, we train a system to solve the camera pose very fast for a new image. For the cases where no a priori information is available a fast minimal case solver is developed. The solver uses five points in two camera views to estimate the cameras relative position and orientation. This type of minimal case solver is usually used within a RANSAC framework. In order to increase accuracy and performance a refinement to the random sampling strategy of RANSAC is proposed. It is shown that the new scheme doubles the performance for the five point solver used on video data. For larger systems of cameras a new Bundle Adjustment method is developed which are able to handle video from cell phones.

Demands for reduction in size, power consumption and price has led to a redesign of the image sensor. As a consequence the sensors have changed from a global shutter to a rolling shutter, where a rolling shutter image is acquired row by row. Classical structure from motion methods are modeled on the assumption of a global shutter and a rolling shutter can severely degrade their performance. One of the main contributions of this thesis is a new Bundle Adjustment method for cameras with a rolling shutter. The method accurately models the camera motion during image exposure with an interpolation scheme for both position and orientation.

The developed methods are not restricted to cellphones only, but is rather applicable to any type of mobile platform that is equipped with cameras, such as a autonomous car or a robot. The domestic robot comes in many flavors, everything from vacuum cleaners to service and pet robots. A robot equipped with a camera that is capable of estimating its own motion while sensing its environment, like the human eye, can provide an effective means of navigation for the robot. Many of the presented methods are well suited of robots, where low latency and real-time constraints are crucial in order to allow them to interact with their environment.
Populärvetenskaplig sammanfattning


Huvudbidraget i denna avhandling är metoder som möjliggör att en kamerautrustad enhet, såsom en mobiltelefon, kan beräkna sin egenrörelse och genom detta återskapa tre-dimensionell strukturer av vad kameran ser. Med hjälp av dessa tekniker skulle man kunna komplettera sina semesterbilder med 3D modeller eller t.ex. statyer och byggnader. Då man kan återskapa tre-dimensionella information finns det också möjlighet att skapa stereobilder från sin mobiltelefonkamera, som sedan kan visas på t.ex. en 3D TV. Ett tredje exempel är så kallad förstärkt verklighet där virtuella objekt kan placeras i kamerabilden. Med denna teknik skulle man kunna göra datorspel som man spelar i ”verkligheten” eller ha en navi-gator som placerar ut skyltar och pilar på vägen eller på en fasad i den verkliga bilden.


Krav på lägre strömförbrukning, minskad storlek, och ett lägre pris har lett till en designförändring hos bildsensorerna för nästan alla nya typer av kameror. Denna designförändring har medfört att bildexponeringen ändrats från en global slutare till en rullande slutare, där den rullande slutaren exponerar bilden rad för rad. Klassiska struktur från rörelse metoder är baserade på ett antagande om en global slutare och om de används på en rullande slutare kan resultatet kraftigt försämras. Ett viktigt bidrag i denna avhandling är en bundle adjustment metod för kameror med rullande slutare. Metoden modellerar noggrant rörelsen hos kameran för varje bildrad med både position och orientering.

De utvecklade metoderna är inte enbart tillämpbar på mobiltelefoner, utan för alla typer av mobila plattformar som är utrustade med kameror, t.ex. en robot eller en autonom bil. En kamerautrustad robot eller bil kan navigera och interagera i sin omgivning, och i likhet med en människa kan en robot få en bättre uppskattning av sin omgivning genom att röra sig och ändra sin vy.
Acknowledgments

I would like to thank all current and former members of the Computer Vision Laboratory in Linköping, contributing to a very friendly and inspiring working environment, and a special thanks goes to:

- **Michael Felsberg** for sharing his wisdom, expertise, giving the very best possible support, and finally for showing tolerance and keeping confidence in me.

- **Per-Erik Forssén** for invaluable discussions and insights, whiteout it this work would never have been the same.

- **Gösta Granlund, Eirk Jonsson, Fredrik Larsson, Klas Nordberg, Björn Johansson, Erik Ringaby** for many and very interesting discussions.

- **Johan Wiklund** for mediating between me and my hardware/software.

- **Per-Erik Forssén, Klas Nordberg** and **Ida Friedleiff** for proofreading this thesis.

I would also like to thank my family and friends, most notably:

- My mother and **Peter** for all love, support and encouragement throughout my life, whithout this support this work would not have been possible

- Last but certainly not least my family, **Ida** for love and massive support and great patience during this period and **Edvin** for always greeting me with a smile after a hard day at work.

Acknowledged goes out to the funders of this research: European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 215078 DIIPLECS, the European Community’s Sixth Framework Programme (FP6/2003-2007) under grant agreement n° 004176 COSPAL, and ELLIT, the Strategic Area for ICT research, funded by the Swedish Government

_Johan Hedborg  May 2012_
# Contents

1 Introduction 1
   1.1 Motivation 1
   1.2 Outline 2
      1.2.1 Included Publications 2

I Background Theory 9

2 Least Squares Problems 11
   2.1 Problem Formulation 11
   2.2 Linear Least Squares 12
   2.3 Non-Linear Least Squares 12
      2.3.1 The Gauss-Newton Method 13
      2.3.2 The Levenberg-Marquardt Method 14

3 Image Correspondences 17
   3.1 Blob Detection 17
   3.2 Corner Detection 19
   3.3 Similarity Metrics 19
   3.4 Correspondence Search 21

4 Pose Estimation 23
   4.1 Overview 23
   4.2 The Pinhole Camera and its Calibration 24
   4.3 Five Point Solution 25
   4.4 Perspective-N-Point Pose Estimation 28
   4.5 Costs 28
   4.6 Pose Search 29

5 Bundle Adjustment 31
   5.1 Overview 31
   5.2 Rolling Shutter 32
   5.3 Bundle Adjustment Problem Formulation 33
   5.4 The Jacobian Structure 33
   5.5 Rolling Shutter Camera 34
CONTENTS

5.6 Rolling Shutter Rotation Rectification . . . . . . . . . . . . . . . . 36
5.7 Rolling Shutter Bundle Adjustment . . . . . . . . . . . . . . . . . . 36

6 Concluding Remarks 39
6.1 Result . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
6.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

II Publications 43
A Real-time view-based pose recognition and interpolation for tracking initialization 45
B KLT Tracking Implementation on the GPU 69
C Fast and Accurate Structure and Motion Estimation 79
D Fast Iterative Five point Relative Pose Estimation 95
E Structure and Motion Estimation from Rolling Shutter Video 113
F Rolling Shutter Bundle Adjustment 131
Chapter 1

Introduction

1.1 Motivation

Digital camera equipped cell phones were introduced in Japan in 2001, they quickly became popular and by 2003 outsold the entire stand-alone digital camera market. In 2010 sales passed one billion units and the market is still growing. This development has led to a very capable imagine device that is small, power efficient and has a low cost. For example, the iPhone 4 camera has a 5Mpix 1/3.2”, back-illuminated CMOS sensor with autofocus, is no bigger than half of a cubic cm, and at a price of only 9 Euro.

The core contribution of this thesis is the development of methods that allow an imaging device such as the cell phone camera to estimate its own motion and to capture the observed scene structure. A portable device such as the cell phone could be used not only for taking pictures, but also to capture the geometry. Besides a collection of holiday photos, whole scenes or statues could be captured and could be viewed as realistic textured 3D models. Another aspect of being able to capture geometry is that, if the geometry is known there is a possibility to change from where an object is viewed. This gives a cell phone camera the ability to create 3D stereoscopic for a 3D capable TV.

Augmented reality has recently captured the interest of many, here a cell phone can be used to augment virtual objects to the real-world environment when looking through the cell phone screen. There are quite a few applications that show this ability, most of them however use a predefined pattern, usually a paper with a special pattern printed on it. In this thesis we investigate markerless methods, which allows us to use the observed scene as “markers” and no special pattern is needed.

A mobile platform such as an autonomous car or a robot, equipped with cameras is capable of estimating its own motion while sensing its environment. However using cameras for navigation is a challenging task, as was shown in one of the largest competitions for unmanned cars in urban environments, the Darpa Urban Challenge 2007. Here all vision sensors where turned off during the final. The vision sensor gave too much noise and were simply too unreliable, however when
CHAPTER 1. INTRODUCTION

a human drives a car more or less all data comes via their vision. This thesis have a focus towards real-time applications, which is well suited for navigation where low latency and real-time constraints are crucial in order to allow fast response. Pose estimation methods presented in this thesis have high accuracy, this can be crucial for mobile platforms e.g. when moving forward.

A challenge that has recently appeared is the introduction of a rolling shutter on cameras. A rolling shutter camera captures the image row by row, and is especially challenging for computer vision methods that handles moving cameras and are based on the assumption of a static scene. Small cheap cameras has had a rolling shutter for a longer time now, but they are also starting to appear in more expensive cameras, even the ones used by professionals (e.g. RED or Canon C300). In this thesis we aim to solve some of the problems associated with a rolling shutter and structure from motion.

1.2 Outline

This thesis is divided into two parts, one introductory part and one publication part consisting of six previous published papers. The introductory part serves as a brief overview of the area of structure from motion which is a research field within computer vision. Underlying theory and concepts are presented, which hopefully will help the reader to better understand the papers in the second part. Additionally, in the introduction the author highlights common parts of the papers and positions them in a larger context.

The papers can be divided into two sub categories, where the first four papers are concerned with pose estimation with a focus on real-time applications. The last two papers contribute with methods used for solving the structure from motion problem on a CMOS rolling shutter sensor, a sensor type which is used in nearly all new consumer imaging devices.

1.2.1 Included Publications

Here is a brief description of the six included papers, followed by an author list and abstract for each of the papers.

Paper A uses a set of \textit{a priori} known camera poses to train a system to fast retrieve a camera pose for a new image. The method is based on P-Channels which is a kind of feature vector and is compared with SIFT.

Paper B is concerned with the task of finding points in a sequence of images that corresponds to one and the same 3D point in the world. The method is known as the KLT-tracker [23] and the papers core contribution is way of accelerating the search using the GPU.

Paper C presents an extension to the RANSAC framework, which lowers the number of iteration by half, and hence doubles performance. The contribution is a strategy for how the minimal set is chosen.

Paper D contains work on a new five point relative pose estimation method, the method is based on the Powell’s dog leg method which is a nonlinear least squares solver. Two possible ways of initialize the solver is also presented.
Paper E focuses on the problem of estimating structure from motion from a camera with a rolling shutter. The paper assumes small translation between frames, and uses a rotation only rectification scheme to rectify points in the image. The points are then used with classical structure from motion methods to reconstruct camera trajectory and structure.

Paper F extends the work of doing structure from motion on rolling shutter images, however in this paper a tailor made bundle adjustment method is presented which fully models the rolling shutter of the camera.

Paper A: Real-Time View-Based Pose Recognition and Interpolation for Tracking Initialization


**Abstract:** In this paper we propose a new approach to real-time view-based pose recognition and interpolation. Pose recognition is particularly useful for identifying camera views in databases, video sequences, video streams, and live recordings. All of these applications require a fast pose recognition process, in many cases video real-time. It should further be possible to extend the database with new material, i.e., to update the recognition system online. The method that we propose is based on P-channels, a special kind of information representation which combines advantages of histograms and local linear models. Our approach is motivated by its similarity to information representation in biological systems but its main advantage is its robustness against common distortions such as clutter and occlusion. The recognition algorithm consists of three steps: (1) low-level image features for color and local orientation are extracted in each point of the image; (2) these features are encoded into P-channels by combining similar features within local image regions; (3) the query P-channels are compared to a set of prototype P-channels in a database using a least-squares approach. The algorithm is applied in two scene registration experiments with fisheye camera data, one for pose interpolation from synthetic images and one for finding the nearest view in a set of real images. The method compares favorable to SIFT-based methods, in particular concerning interpolation. The method can be used for initializing pose-tracking systems, either when starting the tracking or when the tracking has failed and the system needs to re-initialize. Due to its real-time performance, the method can also be embedded directly into the tracking system, allowing a sensor fusion unit choosing dynamically between the frame-by-frame tracking and the pose recognition.

**Contributions:** The author contributed with implementation, partly theory and partly writing, while Felsberg contributed with the idea, theory and writing.
CHAPTER 1. INTRODUCTION

Paper B: KLT Tracking Implementation on the GPU


Abstract: The GPU is the main processing unit on a graphics card. A modern GPU typically provides more than ten times the computational power of an ordinary PC processor. This is a result of the high demands for speed and image quality in computer games.

This paper investigates the possibility of exploiting this computational power for tracking points in image sequences. Tracking points is used in many computer vision tasks, such as tracking moving objects, structure from motion, face tracking etc. The algorithm was successfully implemented on the GPU and a large speed up was achieved.

Contributions: The author contributed with the idea, theory, implementation and writing. The co-authors contributed with writing and partly theory.

Paper C: Fast and Accurate Structure and Motion Estimation


Abstract: This paper describes a system for structure-and-motion estimation for real-time navigation and obstacle avoidance. We demonstrate a technique to increase the efficiency of the 5-point solution to the relative pose problem. This is achieved by a novel sampling scheme, where we add a distance constraint on the sampled points inside the RANSAC loop, before calculating the 5-point solution.
1.2. OUTLINE

Our setup uses the KLT tracker to establish point correspondences across time in live video. We also demonstrate how an early outlier rejection in the tracker improves performance in scenes with plenty of occlusions. This outlier rejection scheme is well suited to implementation on graphics hardware. We evaluate the proposed algorithms using real camera sequences with fine-tuned bundle adjusted data as ground truth. To strengthen our results we also evaluate using sequences generated by a state-of-the-art rendering software. On average we are able to reduce the number of RANSAC iterations by half and thereby double the speed.

Contributions: The author contributed with the idea, theory, implementation and writing. The co-authors contributed with writing and partly theory.

Paper D: Fast Iterative Five point Relative Pose Estimation


Abstract: Robust estimation of the relative pose between two cameras is a fundamental part of Structure and Motion methods. For calibrated cameras, the five point method together with a robust estimator such as RANSAC gives the best result in most cases. The current state-of-the-art method for solving the relative pose problem from five points is due to [28], because it is faster than other methods and in the RANSAC scheme one can improve precision by increasing the number of iterations.

In this paper, we propose a new iterative method, which is based on Powell’s Dog Leg algorithm. The new method has the same precision and is approximately twice as fast as Nistér’s algorithm. The proposed algorithm is systematically evaluated on two types of datasets with known ground truth.

Contributions: The author contributed with the idea, theory, implementation and writing. The co-author contributed with writing and partly theory.

Paper E: Structure and Motion Estimation from Rolling Shutter Video


Abstract: The majority of consumer quality cameras sold today have CMOS sensors with rolling shutters. In a rolling-shutter camera, images are read out row by row, and thus each row is exposed during a different time interval. A rolling-shutter exposure causes geometric image distortions when either the camera or the scene is moving, and this causes state-of-the-art structure and motion algorithms to fail. We demonstrate a novel method for solving the structure and motion problem for rolling-shutter video. The method relies on exploiting the continuity of the camera motion, both between frames, and across a frame. We demonstrate
the effectiveness of our method by controlled experiments on real video sequences. We show, both visually and quantitatively, that our method outperforms standard structure and motion, and is more accurate and efficient than a two-step approach, doing image rectification and structure and motion.

**Contributions:** The author contributed with the idea, theory, implementation and writing. The co-authors contributed with writing and partly theory.

**Paper F: Rolling Shutter Bundle Adjustment**

Abstract: This paper introduces a bundle adjustment (BA) method that obtains accurate structure and motion from rolling shutter (RS) video sequences: RSBA. When a classical BA algorithm processes a rolling shutter video, the resultant camera trajectory is brittle, and complete failures are not uncommon. We exploit the temporal continuity of the camera motion to define residuals of image point trajectories with respect to the camera trajectory. We compare the camera trajectories from RSBA to those from classical BA, and from classical BA on rectified videos. The comparisons are done on real video sequences from an iPhone 4, with ground truth obtained from a global shutter camera, rigidly mounted to the iPhone 4. Compared to classical BA, the rolling shutter model requires just six extra parameters. It also degrades the sparsity of the system Jacobian slightly, but as we demonstrate, the increase in computation time is moderate. Decisive advantages are that RSBA succeeds in cases where competing methods diverge, and consistently produces more accurate results.

Contributions: The author contributed with the idea, theory, implementation and writing. The co-authors contributed with writing and partly theory.
Part I

Background Theory
Chapter 2

Least Squares Problems

A large set of computer vision problems can be formulated within some minimization framework. Methods developed in this thesis are, to a large extent, fitted into a least squares optimization framework. As indicated by the name, a least squares problem has a cost function (or error function) which is a sum of a set of squared costs, and although not being robust against data which has a large deviation from the model (outliers), the least squares methods are usually efficient and have a high rate of convergence. This chapter is a brief introduction to some of the least squares methods used in this thesis. For a more in-depth presentation, the reader is referred to the report [24].

2.1 Problem Formulation

A least squares problem aims to find the $n$ dimensional vector $x$ that minimizes

$$ F(x) = \frac{1}{2} \sum_{i=1}^{I} f_i(x)^2, \quad (2.1) $$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, ..., I$ are given cost functions, and $I \geq n$ is needed to keep the system from not being underdetermined. The global minimizer of $F$, denoted $x^+$, is defined as

$$ x^+ = \text{argmin}_x \{ F(x) \}, \quad \text{where } F : \mathbb{R}^n \rightarrow \mathbb{R}, \quad (2.2) $$

and yields the overall minimum value for (2.1).

To find the global minimizer is in general a hard problem, and in this thesis methods are used that instead aims to find a local minimizer for $F(x)$. A local minimizer is a vector $x^*$ that gives a minimal value for $F(x)$ inside a region with radius $\delta$ around $x^*$:

$$ F(x^*) \leq F(x) \text{ for } \|x - x^*\|_2 < \delta, \quad (2.3) $$

where $\delta$ is some positive number.
Finding a local minimizer is done with the aim of it also being the global minimizer, or at least close to it. For the current discussion, $F()$ is assumed to be differentiable which allows for the following Taylor expansion to be made

$$F(x + h) = F(x) + h^T F'(x) + O(||h||^2), \quad (2.4)$$

where

$$F'(x) = \left( \frac{\partial F}{\partial x_1} \cdots \frac{\partial F}{\partial x_n} \right)^T. \quad (2.5)$$

A necessary condition for $x^*$ to be a local minimizer is

$$F'(x^*) = 0. \quad (2.6)$$

Note that, even if $F'(x) = 0$, $x$ is not guaranteed to be a local minimizer, the stationary point $x$ might instead be either a local maximizer or a saddle point.

### 2.2 Linear Least Squares

If every $f_i$ is a linear function $(a_i^T x - b_i)$ then (2.1) can be formulated as

$$F(x) = \frac{1}{2} \sum_{i=1}^{I} (a_i^T x - b_i)^2 = \frac{1}{2} (Ax - b)^T (Ax - b), \quad (2.7)$$

where $A$ is a matrix with $I$ column vectors given by $a_i$.

The solution to $x$ can be found by first taking the derivative of (2.7) and then setting it to 0:

$$\frac{1}{2} A^T (Ax - b) + (x^T A^T - b^T) A = A^T Ax - A^T b = 0. \quad (2.8)$$

The equation system (2.8) is known as the normal equations and is solved as a standard linear equation system.

The solution for the normal equations yields a global minimizer because linear least squares problems are convex. We are using a linear least squares approach in paper A and it is also used as a part of a non-linear solver for paper E and F.

### 2.3 Non-Linear Least Squares

All non-linear optimization methods used in this thesis are descent methods. A descent method is an iterative method which starts with some initial guess of the solution $x_0$ and iterates through a series of vectors $(x_1, x_2, \ldots)$. This hopefully converges to a local minimizer $x^*$ and if $x_0$ is chosen carefully enough the local minimizer has a good probability of also being the global minimizer. As the name hints, a descent method enforces a descent condition $F(x_{k+1}) < F(x_k)$, which prevents convergence to a local maximizer and makes convergence towards a saddle point less likely.
2.3. NON-LINEAR LEAST SQUARES

2.3.1 The Gauss-Newton Method

In similarity with all other non-linear least squares methods, the Gauss-Newton method also uses an iteration scheme. The Gauss-Newton scheme can be summarized as

\begin{verbatim}
begin
x = x_0
found = false
while (not found)
    h = find_Gauss_step(x)
    if (no such h exists)
        found = true
    else
        x = x + h
end
\end{verbatim}

The method iterates until no more update step can be found or the number of iterations has exceeded some upper limit. An important part of the method is how the next update step \( h \) is found, i.e. how the \texttt{find\_Gauss\_step()} function is defined, which we now will show how to solve.

By defining the vector \( f(x) \) as
\[
f(x) = \begin{bmatrix} f_1(x) & \cdots & f_I(x) \end{bmatrix},
\]
(2.1) can be formulated as
\[
F(x) = \frac{1}{2} f(x)^T f(x) \quad (2.9)
\]

Taylor expanding \( f(x) \) around the point \( x \) gives:
\[
f(x + h) \approx l(h) = f(x) + J(x)h
\]
(2.10)

where \( J(x) \) is the Jacobian matrix, where each element \((i,j)\) in the matrix contains the first partial derivative \( \partial f_i(x)/\partial x_j \). An approximation to \( F(x + h) \) is then given by
\[
F(x + h) \approx L(h) = \frac{1}{2} l(h)^T l(h),
\]
(2.11)

and inserting (2.10) into (2.11) results in
\[
L(h) = \frac{1}{2} f^T f + h^T J^T f + \frac{1}{2} h^T J^T J h
\]
(2.12)

with \( f = f(x) \) and \( J = J(x) \).

The goal is to find the argument \( h \) which minimizes \( L \):
\[
\arg\min_h \{ L(x) \}
\]
(2.13)

Similar to the linear least squares approach, we take the derivatives w.r.t. \( h \) and set the resulting equation to \( 0 \), resulting in
\[
J^T f + J^T J h = 0.
\]
(2.14)
The vector $h$ is now calculated by solving the linear equation system $(J^T J)h = -J^T f$ (called the normal equations as in section 2.2).

The current solution vector $x$ is updated with the update step $h$ under the condition that it exists a better solution and that $h$ is not too small, else the iterations are stopped.

### 2.3.2 The Levenberg-Marquardt Method

Levenberg and later Marquardt came up with a damped version of the Gauss-Newton method, by introducing a damping term $1/2 \lambda h^T h$ into (2.12). The damping term is used to penalize large steps, taken by the vectors $h$.

The introduction of $\lambda$ is mainly done for two reasons. First, with $\lambda > 0$ we always get a positive definite coefficient matrix when solving the normal equations, which ensures a descent direction. Second, if $x$ is close to a local minimizer, a quadratic model of the cost surface fits well, a small $\lambda$ results in a type of Gauss-Newton method which has close to quadratic convergence near a local minimizer. Or if the quadratic model does not fit well, it is in many cases better to be more careful and take shorter steps which is what a larger $\lambda$ will do. The strategy is hence, to start with some larger $\lambda$ and let it decrease as $x$ is getting closer to the minimizer.

For deriving the update function for $h$ (denoted find\_LM\_step()) consider (2.12) but now with the additional damping term:

$$L(h) + \frac{1}{2} \lambda h^T h = \frac{1}{2} (f^T f + 2h^T J^T f + h^T J^T Jh + \lambda h^T h).$$

(2.15)

Find the argument $h$ which minimizes

$$\arg\min_h \{L(h) + \frac{1}{2} \lambda h^T h\},$$

(2.16)

which, as before is done by taking the derivative of (2.15) w.r.t. $h$ and then setting it to 0:

$$J^T f + J^T Jh + \lambda h = 0$$

(2.17)

This results in the normal equations for the Levenberg-Marquardt method:

$$(J^T J + \lambda I)h = -J^T f.$$
2.3. NON-LINEAR LEAST SQUARES

begin
x = x0, λ = λ0
v = 2
while not converged
    h = find_LM_step(x)
    xnew = x + h
    ϱ = (F(x) - F(xnew))/(L(0) - L(h))
    if(ϱ > 0)
        x = xnew
        λ = λ · max(\{\frac{1}{3}, 1 - (2ϱ - 1)^3\})
        v = 2
    else
        λ = λv
        v = 2v
end

The variation of λ is decided by the quality measure ϱ. A large value of ϱ indicates that L(h) is a close approximation to F(x + h) and for the next iteration we should decrease λ, which will make the next step closer to a Gauss-Newton step. If ϱ is small the approximation is not good and λ will be increased. There are several techniques for choosing by which amount to increase or decrease λ, here we use the one presented in [27].
Chapter 3

Image Correspondences

The majority of image processing algorithms presented in this thesis are based on multiple images as input. This usually involves an initial processing step which finds where image regions in one image are located in others, which is commonly referred to as finding image correspondences. In order for the localization to work, the image regions must contain some local image structure. Without this it is not possible to find any correspondences and it would be like trying to localize a part of a white wall only looking at a local all-white region. The structure should preferably have at least two separate directions within the region. In order to re-localize it in a second view, the structure also needs to be coherent enough over the view change. The methods in this thesis uses a point representation of the region (the center point) and region correspondences are denoted as point correspondences.

3.1 Blob Detection

A blob detector is a method which detects parts of the image with elliptical characteristics. Various types of blob detectors exits There are different types of blob detectors [21, 25, 7], where one of the most commonly used is the Difference of Gaussians (DoG) detector. In short the DoG works by using a band-pass filter on the image which isolates a frequency range, in this frequency range a search for local maxima is done to find blobs. This is applied for several frequency bands allowing the detector to detect blobs of different sizes. Two band-pass kernels and their response images together with a local max are illustrated in figure 3.1, upper row.

A band-pass filter can efficiently be implemented by subtracting two images that have been blurred by Gaussians with different variances, which effectively suppresses all lower frequencies. The efficiency comes from using lower resolution for images with low frequency content. Although not used in this thesis, we choose to briefly describe it because of the wide usage of the SIFT detector and descriptor [22] in the field of structure from motion, where the DoG is essentially the SIFT detector, apart from some small tricks to avoid the SIFT detector to firing on
CHAPTER 3. IMAGE CORRESPONDENCES

Figure 3.1: Blob and corner detection

Difference of Gaussians

Shi-Tomasi

FAST
3.2 Corner Detection

Besides blobs, corner-like regions are also of potential interest. A corner can be defined as the intersection of two edges and can be detected in different ways. In this thesis we use the Shi-Tomasi corner detector [31] and the FAST detector [30].

The structure tensor is a 2x2 matrix that summarizes the predominant directions of the image gradient around a point. The Shi-Tomasi detector calculates the corner response as the minimal eigenvalue of the structure tensor. The corner response is an image where each pixel represents a measurement of how much image structure there is in the direction with the lowest amount of variation. The corners are found by performing a search for local maxima in the filter response image. The middle row of figure 3.1 shows a part of a checkerboard image, its Shi-Tomasi response image and the detected local maximum. It is common to define the Shi-Tomasi detector as a corner detector, but it actually gives maximal response when the edges go all across the surrounding region, e.g. a cross or chess pattern. This method is utilized in paper C and D.

The FAST detector uses a different approach than the Shi-Tomasi one. Here the corner detection is done by examining pixels in a circle around the point. If sufficiently many consecutive pixels on the circle have a large enough intensity difference relative to the center pixel, then the center pixel is classified as a corner. The consecutive pixels need to all be smaller or all larger than the center. A sufficient number of consecutive pixels is usually defined as more than the amount it takes to create a half circle. In contrast to the Shi-Tomasi detector, FAST will give maximal response for a corner. One advantage with this method is that it is fast, as implied by the name. The speed is due to certain properties of the detector, no edge detection is needed, the number of pixel lookups is relatively low compared to competing detectors and finally by testing only four points evenly distributed on the circle a primary selection can be done to sort out potential candidates. An image containing a corner, the partial test pattern and the full test pattern are illustrated in figure 3.1. We have used the FAST detector in paper E and F.

3.3 Similarity Metrics

Projections of one and the same real world scene can look very different, things like illumination differences and viewing angle can alter the appearance drastically. The choice of similarity metric is dependent of the type of data being considered. Preferably, the amount of invariance in a metric should be enough to deal with the appearance changes, but too much invariance can make precision worse and correspondence search more difficult. With low differences in appearance, the Sum of Squared pixel intensity Differences (SSD) over a region is a good choice. It achieves high position accuracy and has a continuous derivative that allows for a fast and accurate search. The low appearance difference property is typically
Figure 3.2: Correspondence search for the different metrics. P-Channel image, courtesy of Erik Johnson
3.4. CORRESPONDENCE SEARCH

met in video sequences, where neighboring frames have large similarities. The upper row in figure 3.2 shows two images for a video stream and a cost surface that is constructed by evaluating the SSD for different displacements along the two different dimensions of the image.

If the illumination changes are larger and if the viewing angle is larger, the SSD metric becomes less discriminative. In these cases, it is possible to make the metric invariant to differences in illumination by using the gradient of the image instead of intensity. Additionally, large viewing differences can be handled by a local histogram approach which creates coarse statistics about edge content, this is less dependent on the exact location of the edge. In SIFT, a region is divided into 4x4 sub-regions and from each region an 8-bin large edge histogram is estimated, forming a 128-valued vector. These steps are illustrated in figure 3.2. The Euclidean distance (or any other suitable distance) between two of these vectors can be later used as a metric between two image regions.

Color information is completely discarded using SSD or SIFT metric, however this kind of information is quite discriminative for some type of image regions. In paper A, we are using a descriptor vector that, like SIFT, uses a type of local histogram of edges. Additionally, the descriptor is extended with color and intensity information. Instead of standard local histograms, we use P-Channels, which allows for a more accurate representation [4]. The P-Channels add an offset on each histogram bin which represents the bin’s “center of mass”. As with the SIFT descriptor, any suitable metric can be used to measure the distance between two descriptor vectors. An overview and evaluation of similarity metrics can be found in [19].

3.4 Correspondence Search

Once a metric is defined, it is possible to do a search for corresponding regions. If the metric/cost function is squared as in the case of SSD, the correspondence search can be formulated as a least squares problem. The cost can then be minimized by using a suitable method from chapter 2. The KLT-tracker uses an SSD cost function and a Gauss-Newton method, introduced in section 2.3.1, in order to search for correspondences [23]. Derivation and a GPU-accelerated implementation of the KLT-tracker is described in paper B. The KLT tracker is also used in paper C-F, because they mainly focus on image sequences from video.

For other metrics it is less straightforward to make such a search. An alternative approach is to limit the search to the discrete positions around the interest points. The SIFT descriptor makes it fairly inexpensive to compare two regions by only comparing the two 128-valued vectors. Having a fast way to compare two regions and a limited set of points to compare, it is possible to do an exhaustive search within a reasonable time frame. Alternatively, the comparison can be made more efficiently by replacing the exhaustive search with a Kd-tree search [26]. This kind of search process is usually referred to as descriptor matching.

In the case where we have access to a set of images with known poses, it is possible to directly find a mapping between the descriptor and the pose by super-
vised learning. This implies finding a mapping from a set of given poses to their associated descriptors. This can be posed as a linear least squares problem from section 2.2 and can be solved efficiently with a standard linear algebra package. With the learning done, the process of finding a pose from a descriptor vector is very fast, the method is described in more detail in paper A.
Chapter 4

Pose Estimation

4.1 Overview

With camera pose estimation we refer to the task of establishing the location from where an image was taken. A camera pose is a composition of a direction and a position, where the camera direction is represented by a rotation on the unit sphere and the position is a 3D coordinate. The pose can be found from image content alone and there are numerous examples where this has been applied, e.g. robotics, car safety, movie making, measuring and inspection.

There are other sensors for estimation of position and rotation such as the GPS (Global Positioning System) and the IMU (Inertial Measurement Unit). When compared to a normal GPS, the camera is more accurate in position over shorter distances but the GPS is georeferenced and is not subject to error accumulation as with the camera, which effectively gives it better accuracy over larger distances. Compared to an IMU, a pose estimated with a camera tends to have considerably less drift during normal motion. For fast motions the IMU tends to be a better choice due to motion blur and large field of view differences.

The camera, IMU, and GPS can be seen as complementary sensors. Where the IMU is good in a short time interval, the camera is better at longer time intervals but still subject to drift, and for large distances the GPS has the best accuracy. The camera and IMU combination is investigated in [17].

A strength of cameras compared to IMU and GPS is the ability to sense scene structure. This is possible while or after a pose has been established and builds upon the principles of triangulation. Having a set of sparse point correspondences this is fairly straightforward [10], and in the case of a dense estimation it is more complicated [16].

A common approach is to establish the relative pose between two or more cameras in an initial calibration step using a calibration pattern with known geometry. Because the relative pose must remain unchanged during usage, this approach requires a setup with multiple rigidly fixed cameras that are synchronized with respect to image acquisition. This type of setup is typically referred to as a stereo camera setup and is common for depth estimation, which essentially is
a type of scene geometry estimation. With the use of known calibration patterns, the relative pose between the cameras in the stereo rig can be very accurately estimated.

Alternately the pose can be estimated from the given image content without the help of a special pattern. This is a more challenging task due to the unknown geometry and image correspondences that are less accurately localized than the ones from a specialized calibration pattern. There are however gains to be made from this approach. By moving a single camera it is possible to create a set of stereo images by taking multiple exposures. This approach eliminates the need for camera rigs and synchronization but the scene geometry which is observed needs to be static. The ability to move the camera more freely gives the flexibility to create larger baselines between images, resulting in better geometry estimation. Even if the fixed base-line stereo rig has a better estimated relative pose, the wider baselines compensate for this and the resulting geometry is generally better.

If there is no structure with known dimensions in the scene there is no way of knowing the absolute scale of the scene, much like the way the film industry fools us by using miniature models and scenes. This is referred to as the scale ambiguity and it is a source of error which will accumulate over the sequence. These errors can have a large impact on the final estimate and a partial solution to this problem will be discussed in chapter 5.

There is also the option to fuse the two approaches, where the pose of a stereo camera is estimated and larger baselines can be utilized as well. Because the baseline is fixed over the sequence, there is a known distance between pairs of cameras, and this can be used to resolve the scale ambiguity.

4.2 The Pinhole Camera and its Calibration

Let \((u, v)^T\) be the projection of a 3D scene point \(X = (x, y, z)^T\) to an image plane which lies in the xy-plane and passes through the point \(0, 0, f)^T\). This can be expressed as

\[
u = \frac{x f}{z}, \quad v = \frac{y f}{z}.
\] (4.1)
4.3. FIVE POINT SOLUTION

The origin in the image coordinate system is usually not in \((0, 0, f)^T\) as in (4.1) but rather with an offset \((u_0, v_0)\). Equation 4.1 with offset can be expressed in a matrix representation as

\[
\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}
\]

(4.2)

where \(\lambda = z\).

Pixels on the image sensor might not be perfect squares and to compensate for this an aspect ratio parameter \(\eta\) can be introduced into the model. In the case where the sensor is a bit skewed \(\gamma\) can compensate for this, see figure 4.1. This results in a calibration matrix \(K\) with five degrees of freedom,

\[
\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & \gamma & u_0 \\ 0 & \eta f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = KX.
\]

(4.3)

This is known as the pinhole camera model due to the projection through a single point (the focal point). If the camera’s focal point is not at the origin, a coordinate transformation consisting of a rotation \(R\) and a translation \(t\) is applied

\[
\lambda \begin{pmatrix} p \\ 1 \end{pmatrix} = C \begin{pmatrix} X \\ 1 \end{pmatrix}, \text{ where } C = KR^T[I - t].
\]

(4.4)

Here \(\lambda\) is no longer equal to \(z\) as in (4.2), but also dependent on \(R\) and \(t\). \(C\) is called the camera matrix, and consists of five intrinsic parameters (in \(K\)) and six extrinsic parameters (in \(R\) and \(t\)). The extrinsic parameters are also referred to as the camera pose.

Lacking in the pinhole camera model is the distortion that results from the light first passing through a set of optical lenses. This results in a number of optical distortions like radial distortion, spherical aberration, coma and astigmatism. However the non-linear geometrical distortion resulting from the radial distortion is usually the one which affects pose estimation the most, see figure 4.1.

In paper C-F calibrated cameras are used, where a calibration step is done to estimate \(K\) and the radial distortion parameters in advance. The implementation is from OpenCV and is based on the method from [33]. Once the camera parameters are found all image points can be rectified and the camera matrix \(C\) is reduced to \([R^T] - R^Tt\). This is not to be confused with a calibration step for a stereo rig, where in addition to the intrinsic parameters the extrinsic parameters are also estimated.

### 4.3 Five Point Solution

In the case of two calibrated cameras viewing a rigid scene it is possible to estimate their relative rotation and translation, only using five image point correspondences. This will now be shown.
Figure 4.2: An object viewed from two cameras
Consider the two cameras, one of them is at the origin and the other has a relative rotation \( \mathbf{R} \) and position \( \mathbf{t} \) as illustrated in figure 4.2. Let \( \mathbf{X}_i \) be a 3D point with index \( i \), its projection in camera 1 is \( \mathbf{p}_i \) and in camera 2 it is \( \mathbf{q}_i \). This can be expressed as

\[
\mathbf{p}_i = \frac{1}{\lambda_p} \mathbf{X}_i \\
\mathbf{q}_i = \frac{1}{\lambda_q} \mathbf{R}^T (\mathbf{X}_i - \mathbf{t}).
\]

(4.5)

Denote the two camera centers \( \mathbf{c}_1 \) and \( \mathbf{c}_2 \) then the vectors between \( \mathbf{X}_i \) and the two camera centers be \((\mathbf{c}_1 - \mathbf{X}_i)\) and \((\mathbf{c}_2 - \mathbf{X}_i)\). Because all vectors lie in the same plane, the cross product between \( \mathbf{t} \) and \((\mathbf{c}_2 - \mathbf{X}_i)\) (the normal of the plane) is perpendicular to \((\mathbf{c}_1 - \mathbf{X}_i)\):

\[
(\mathbf{c}_1 - \mathbf{X}_i)^T (\mathbf{t} \times (\mathbf{c}_2 - \mathbf{X}_i)) = 0.
\]

(4.6)

By using the knowledge that \( \mathbf{p}_i \) is parallel to \((\mathbf{c}_1 - \mathbf{X}_i)\) and \( \mathbf{Rq}_i \) is parallel to \((\mathbf{c}_2 - \mathbf{X}_i)\) it is possible to rewrite equation 4.6 as

\[
\mathbf{p}_i^T (\mathbf{t} \times \mathbf{Rq}_i) = 0.
\]

(4.7)

Introducing the antisymmetric matrix \( \mathbf{T} \)

\[
\mathbf{T} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}
\]

(4.8)

for the cross product yields \( \mathbf{t} \times \mathbf{a} = \mathbf{T} \mathbf{a} \). Plugging (4.8) into (4.7) gives

\[
\mathbf{p}_i^T \mathbf{TRq}_i = 0,
\]

(4.9)

where \( \mathbf{TR} \) is called the essential matrix, denoted here as \( \mathbf{E} \). The rotation matrix \( \mathbf{R} \) has three degrees of freedom (DOF). The translation can only be recovered up to an unknown scale factor. Setting the length of the translation vector to 1 gives two DOF for \( \mathbf{T} \), resulting in five DOF for \( \mathbf{E} \). For each point correspondence \((\mathbf{p}_i, \mathbf{q}_i)\) equation 4.9 gives one constraint, which means that five point correspondences are needed to solve for \( \mathbf{E} \).

In summary the relative five point pose problem can be formulated as

\[
\text{Given } \mathbf{K} \text{ and five corresponding point pairs } \\
\{ (\mathbf{p}_i, \mathbf{q}_i) \}_{i=1}^5 \text{, find } \mathbf{E} \text{ from which } \mathbf{R} \text{ and } \mathbf{t} \text{ (up to scale) can be extracted.}
\]

We use the five point closed form solver proposed by Nistér [28] in paper C. Paper D shows an iterative solver for the five point problem using the Powell’s dog leg method. In similarity with the Levenberg-Marquardt method (section 2.3.2), Powell’s dog leg method is based on the Gauss-Newton method for least squares optimization. Our five point solver is shown to be around twice the speed of the method of Nistér. A five point solver is also used to initialize the bundle adjustment solvers in paper E and F.
4.4 Perspective-N-Point Pose Estimation

The five point solver yields a solution to the essential matrix from which the pose is extracted. This solves the pose problem for image correspondences only; there are however cases where reliable geometrical data in the form of corresponding 3D points is available. In these cases it can be faster and more stable to solve for a new pose from the 3D point to image point correspondences.

The problem is known as the perspective-N-point problem and can be formulated as

\[
\text{Given } K \text{ and } N \text{ corresponding point pairs } \{(p_i, X_i)\}_{i=1}^{N}, \\
\text{find } R \text{ and } t.
\]

Without going in to further details, the minimal case for the perspective-N-point problem is three point correspondences and is denoted as the perspective-3-point problem \([6, 9]\). In this thesis we are using the perspective-3-point solver in conjunction with the five point solver to estimate the relative pose between three cameras. First a pose is estimated between two cameras for five points. Then a third camera is added by first triangulating three 3D points from the first two images and their poses. The 3D points and their corresponding points in the third image are used with the perspective-3-point method to estimate the last camera pose. This allows for a relative pose estimation between three views from five points \([28]\), and is used in paper C, E and F. In E and F it is only used as an initial step because we later estimate a structure that is good enough to use with a Perspective-N-Point method. In paper F a special Rolling shutter version of the Perspective-N-Point method is developed and used, more on rolling shutter in section 5.2.

4.5 Costs

The minimal solvers give exact solutions, meaning that for the minimal set the only source of error is from errors introduced by the limited precision of the floating point arithmetics used. Due to errors in the image correspondence estimation, a solution from a minimal set of points is not likely to yield a good solution to the pose. To increase precision, the pose estimations are done for several sets of points and in order to measure the quality of different poses, each pose is evaluated against the rest of the point correspondences. A quality measure or cost can be assigned by looking at the point to epipolar line distance in image space, let \(p_i\) be point \(i\) in the first image and \(q_i\) is the corresponding point in the second image. Then \(p_i^T E\) represents a line \(l_i\) in the second image called the epipolar line. The distance between the epipolar line and \(q_i\) gives an indication of how much the point correspondence deviates for the pose model. To summarize, the cost \(C_i^E\) for point correspondence \(i\) is

\[
C_i^E = ||\hat{l}i q_i||_2, \quad \text{where} \quad \hat{l} = 1/\sqrt{l_1^2 + l_2^2}
\]  

(4.10)
4.6. **POSE SEARCH**

Quality measurements that are based on a point-to-line distances do not account for error along the line. In the case of Perspective-N-Point problem, the known structure can be re-projected by the estimated pose to the image plane and a cost can be formulated based on a point-to-point distance instead. Define the projection mapping \( \text{proj}() : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \) as

\[
\text{proj}(y) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}^T.
\]

(4.11)

The re-projection error or cost \( C^R \) is then defined as

\[
C^R_i = ||p_i - \text{proj}(C[X_i^T 1]^T)||_2.
\]

(4.12)

### 4.6 Pose Search

Minimal solvers play an important role when the pose estimation should be done in a robust way. To achieve a robust estimate, the solver is run for several iterations, where in each iteration a new minimal set of points is used to solve for the pose. Each pose is scored with a robust scoring method. Finally the pose with the highest score is chosen.

The original scheme is known as RANSAC (RANdom SAMple Consensus) [6] and scores a pose by counting the number of inliers, where an inlier is defined as a point correspondences which cost falls below some threshold \( \tau \). The inlier count \( n \) can be expressed as

\[
n = \sum_{i=1}^{I} \alpha(C_i), \quad \text{where} \quad \alpha(C) = \begin{cases} 
1 & \text{if } C < \tau \\
0 & \text{if } C \geq \tau
\end{cases}
\]

(4.13)

for \( I \) number of points. In RANSAC the points in the minimal set are randomly chosen. Although the original paper uses RANSAC with a perspective-N-points solver, the framework has been used with a large variety of solvers.

A common property for the minimal solvers is that they give multiple solutions which are all valid for the minimal set. However the camera can only be in one pose and only one of the solutions will fit more than the minimal set, implying that in order to single out the right solution more points are needed. In the case of the five point method there are up to 10 solutions [28], and in the case of the perspective-3-point solver there are up to four solutions [9].

In paper C RANSAC is used together with the five point solver. Here we propose to extend the random choice with a constraint on the minimal distance between the chosen points in image space. The five point solver developed in paper D is used within a RANSAC framework, and Paper E and F use a RANSAC framework together with a five point solver to initialize the structure from motion scheme.

An alternative to estimate pose estimation is to use a supervised learning methodology. With supervised learning we refer to the task of inferring a function from labeled training data. In paper A the training data consists of images with an associated pose. The function argument is a P-Chanel feature vector and the output is a pose.
CHAPTER 4. POSE ESTIMATION
Chapter 5

Bundle Adjustment

5.1 Overview

In the field of structure from motion, bundle adjustment is the process of refining both the camera poses and the 3D structure. Optimizing over all parameters results in a large optimization problem and that even for moderately sized problems would take a very long time to solve with a standard nonlinear least squares solver. There are however some problem-specific properties of bundle adjustment, that allows the optimization to be done significantly faster.

As previously discussed in chapter 4, using the time dimension to get multiple views from a single camera results in a scale ambiguity. Both the relative five point method in three views and the perspective-N-view method can be used to solve the structure from motion problem for more than three views. With the relative pose between three views, two views can be overlapped and relative scale is retrieved. The perspective-N-point method can be alternated with triangulation to retrieve the camera poses. Both approaches deal with the scale ambiguity but are highly susceptible to error accumulation, leading to fast deterioration of the solution and total failure is not uncommon [3].

By refining the cameras and 3D points with bundle adjustment after a new camera (or a few) is added to the system, it is possible to significantly reduce the error accumulation. This is the typical application for bundle adjustment and is referred to as incremental or sequential bundle adjustment. In such an approach, bundle adjustment is not only a refinement step, it actually enables the solution of large sets of cameras to be found.

Recent development in bundle adjustment allows a solver to optimize over a large amount of cameras and 3D points. By the use of conjugate gradient methods the memory footprint has been reduced, enabling large systems of over tens of thousands of cameras to be solved [2, 1]. Large speedups have been achieved by adapting part of the bundle adjustment solver to run on multi-core processors and GPUs [18, 32].

Bundle adjustment has been shown to work well in real-time applications. Engels et al. [3] use a 20 views windowed approach where for each new view they
solve the bundle adjustment problem for the last 20 views. By keeping the system this small they are able to achieve real-time performance. It is further shown that, despite the quite small window size, this prevents failures and enables the system to deliver significantly more long-term stability. Klein et al. [20] showed a stable real-time augmented reality approach where two bundle adjustment systems where run simultaneously, one smaller (local) that is updated more frequently and one larger (global) that is less frequently updated. The system is called PTAM and is still up to this date considered to be state-of-the-art in augmented reality.

5.2 Rolling Shutter

A key modeling assumption in bundle adjustment is that each image is associated with one single camera pose. In order to satisfy the assumption for a moving camera, the whole image has to be captured at one time instance, referred to as the camera having a global shutter.

This is the case for the CCD (Charge-Coupled Device) sensor which has been the dominant sensor in digital imaging. Due to cost and energy efficiency the CCDs are gradually being replaced by the CMOS (Complementary Metal-Oxide-Semiconductor) sensor. With very few exceptions all cell-phones, consumer- and prosumer-camcorders, and movie capable DLSRs have CMOS sensors with a rolling shutter. Contrary to a global shutter the rolling shutter captures each image row at different time instance, usually in a sequential order, from top to bottom.

Rolling shutter results in geometric distortions in the image if there is camera motion or moving objects in the scene; here follows a short illustrative example for an iPhone 4 camera. The rolling shutter camera on the iPhone takes 30ms to capture a complete image. In video mode the lens has a horizontal angle of view of 46.7° and the video format is 720p (1280x720). For a horizontal panning motion where the camera is turned 90°, the following table of rotation speeds and pixel displacements can be composed:

<table>
<thead>
<tr>
<th>Rotation speed</th>
<th>Motion type</th>
<th>Rotation, one image</th>
<th>Image displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>90° in 2s</td>
<td>rapid motion</td>
<td>1.35°</td>
<td>37 pixels</td>
</tr>
<tr>
<td>90° in 4s</td>
<td>normal motion</td>
<td>0.68°</td>
<td>18.5 pixels</td>
</tr>
<tr>
<td>90° in 8s</td>
<td>slow panning</td>
<td>0.34°</td>
<td>9.3 pixels</td>
</tr>
</tbody>
</table>

Although being simplified, the example clearly shows the large amount of distortion that can be expected within one image due to the rolling shutter.

For normal use the distortion poses no big problem, in worst case things look a bit wobbly from camera shake or things become skewed when panning. The problem arises when using methods that assume a global shutter and, as in the case of structure from motion, are sometimes unstable even with a global shutter. In our experience the global shutter re-projection errors, when doing structure from motion on video sequences, are normally below one pixel. Solutions to the rolling shutter problem are discussed in papers E and F.
5.3 BUNDLE ADJUSTMENT PROBLEM FORMULATION

Let us consider a system where \( J \) cameras \( C_j, j = 1 \ldots J \) depict a static scene which is represented with a set of \( K \) 3D points \( X_k, k = 1 \ldots K \). \( p_{j,k} \) is an observation (a tracked or matched image point) of 3D point \( k \) in camera \( j \). Similar to (4.12) the estimated cost for one image point then becomes

\[
||p_{j,k} - \text{proj}(C_j[X_k^T 1]^T)||_2^2.
\]

(5.1)

If all 3D points are visible in all cameras there would be a total of \( J \times K \) observed image points. However, in general the point projections are fewer due to viewport changes, scene occlusions and other things that cause the point correspondence search to fail. Taking this into consideration, a subset \( V_j \) of the 3D points are defined for each camera \( j \) which consists of all points that are visible in that particular camera. A least squares cost function can now be formulated as

\[
F(y) = \frac{1}{2} \sum_{j=1}^{J} \sum_{k \in V_j} ||p_{j,k} - \text{proj}(C(w_j)[X_k^T 1]^T)||_2^2.
\]

(5.2)

where \( y = [w_1 \ldots w_J \; X_1 \ldots X_K]^T \) is a vector containing all parameters for the cameras and the 3D points, \( w_j \) is a vector containing the six extrinsic parameters of camera \( C_j \) and \( X_k \) is the coordinate of 3D point \( k \).

Equation 5.2 could potentially be minimized with any nonlinear least squares method from chapter 2. However the usual approach is to use the Levenberg-Marquardt method from section 2.3.2 and this is the approach we have chosen for papers E and F.

5.4 The Jacobian Structure

In order to use Levenberg-Marquardt for solving the bundle adjustment problem, the Jacobian \( J = \partial f/\partial y \) needs to be evaluated. On the left side of figure 5.1 the structure of the Jacobian is illustrated. This small system uses four calibrated cameras, 10 3D points and 26 image projections. The matrix consists of two types of blocks, a 2x6 type that is spread sparsely over the left side of \( J \) and a 2x3 type that is here gathered per 3D point on the right side of \( J \). The 2x6 block structure is a result of the camera parameters being differentiated w.r.t. the point projection, and similar reasoning results in the 2x3 block size for the 3D points. Because every image point only depends on one camera pose, the elements corresponding to the other camera’s pose parameters will be 0. The same is true for the 3D points; each image point is only dependent upon one 3D point. This results in a sparse matrix where only 6+3 elements per row are not equal to 0. The structure of \( J \) is shown to the left in figure 5.1. On the right side the symmetric approximate Hessian \( J^TJ \) matrix is shown. Due to the sparse structure properties of \( J \), the multiplication \( J^TJ \) also results in a sparse structure.

Handling the structural properties of the Jacobian in an efficient way is the key difference between a normal Levenberg-Marquardt solver and a bundle adjustment
CHAPTER 5. BUNDLE ADJUSTMENT

Figure 5.1: Left: The Jacobian \( \mathbf{J} \), on the horizontal axis is the parameter vector \( \mathbf{y} \), first with 4x6 camera parameters and then the 3x10 3D point parameters. Right: The product \( \mathbf{J}^T \mathbf{J} \). Non-zero elements are marked as dark squares.

The solver. The toy example resulting in the Jacobian in figure 5.1 would not pose a problem. However with more realistic sizes the system quickly gets very large and it is not possible to handle it efficiently, i.e., a system of 100 cameras and 20K 3D points would result in a \( \mathbf{J}^T \mathbf{J} \) matrix with the size 60600x60600 which would consume over 30 TB of memory if double precision were used. How this sparsity is efficiently handled is discussed in section 3.3 in paper F.

5.5 Rolling Shutter Camera

As previously mentioned, in section 4.2, the extrinsic parameters (camera pose) for a camera consists of six parameters, three for rotation and three for translation. If each row of an image is exposed during different time instances each row is associated with a 6 parameter pose, resulting in an extrinsic parameter count of 6x number of image rows in order to represent the camera geometry of a single image. Besides the sheer amount of computation that would be required to solve a bundle adjustment system of this kind, there is a problem with the amount of information contained in one image row. Even if the row contains a large set of point correspondences the pose can not be uniquely determined and thus not solvable for points which all lies on a line.

The solution is to have an interpolation scheme, where the bundle adjustment system optimizes over a set of key rotations and key positions, here denoted as key poses. In the system all intermediate poses are interpolated from the nearest neighboring key poses. Where and how frequent one should place key poses in the image is a regularization trade-off. If too many poses are used the problem
5.5. ROLLING SHUTTER CAMERA

Figure 5.2: Rolling Shutter key-pose $C_0$, $C_1$ and $C_2$ placement

becomes to loosely constrained while too few poses can’t model the camera path accurately enough. Figure 5.2 shows one pose per top row.

The choice of interpolation scheme can also be varied, where different interpolation schemes have different complexity and locality. For paper E and F we have chosen to linearly interpolate the position and use spherical linear interpolation (SLERP) for the rotations. The advantages of using linear interpolation are the locality, where only the two closest key poses are used for the interpolation and where differentiation is used, linear interpolation is simpler and faster to calculate. On the negative side is the inability to represent a continuous and more realistic camera path, when compared to a more complex interpolation schemes as the cubic B-splines.

The camera matrix $C_j$ for a rolling shutter camera is like the one in equation 4.4 except that the rotation $R$ and the translation $t$ is now a function of the image row $y$:

$$C_j(y) = R_{j,j+1}(\tau)[I] - t_{j,j+1}(\tau). \quad (5.3)$$

$R_{j,j+1}(\tau)$ is the SLERP interpolated rotation between key rotation $R_j$ and $R_{j+1}$, and the camera translation is linearly interpolated between position $t_j$ and $t_{j+1}$. $\tau$ is the time for which the image has been captured. $\tau$ could also be expressed as a linear function of the image row.

Compared to a global shutter camera the rolling shutter cameras have one additional parameter that needs to be estimated. This is the time difference between the acquisition of the first image row and the last image row, marked as $\tau_r$ in the figure. This parameter is referred to as the read-out time and there exist various methods to estimate it. The method used in paper E and F uses an LED diode which varies its intensity with a known frequency. The diode is positioned close to the lens in order to light as much of the image as possible. Because of the rolling shutter, the varying intensities will be registered at different rows as is illustrated to the right in figure 5.3. Counting the number of rows between each intensity peak gives an estimate of the number of rows per time unit, and with the known frequency, the read-out time for a complete image can be estimated. The read-out time is denoted $\tau_r$ in figure 5.2.
5.6 Rolling Shutter Rotation Rectification

If the camera translation between a set of frames is small in relation to the distance to the depicted scene, the parallax displacements in the image can be assumed to be small. This assumption is exploited in paper E, where we only use a small time window of three frames from a 30 Hz video stream. During this time interval the parallax, for certain camera paths and scenes, is very low and allows us to simplify the camera pose to a rotation only model [8].

Replacing the standard camera model in equation 4.4 with the rolling shutter camera model from equation 5.3 and eliminating the translation component \( t \), results in

\[
\lambda_j \begin{pmatrix} p_j \ 1 \end{pmatrix} = R_{j,j+1}^T(\tau)X
\]

for camera \( j \). Due to the zero translation assumption, no projection displacement is present, hence the distance parameter \( \lambda \) has no effect on the solution and can be set to 1 for all points. Let us consider two image points, one in image 1 (\( p_1 \)) and the other one in image 2 (\( p_1 \)). By using the relation in (5.4) it is possible to eliminate \( X \), which results in

\[
\begin{pmatrix} p_1 \ 1 \end{pmatrix} = R_{1,2}^T(\tau_1)R_{0,1}(\tau_0)\begin{pmatrix} p_0 \ 1 \end{pmatrix}.
\]

(5.5)

Here the time variable \( \tau_0 \) is linearly depended on the image row position of \( p_0 \), and the same is valid for \( \tau_1 \).

A cost function which minimizes the re-projection error for this problem is defined as

\[
\sum_{k=1}^{K} ||p_{1,k} - \text{proj}(R_{1,2}^T(\tau_{1,k})R_{0,1}(\tau_{0,k})p_{0,k}^T1)||^2
\]

(5.6)

where \( k \) is the 3D point index. Once the cost function is defined any choice of a non-linear least squares solver can be used to minimize (5.6). By eliminating all unknown depths the optimization problem is reduced to only include six free parameters, three from key rotation \( R_1 \) and three from key rotation \( R_2 \). (\( R_0 \) can be set to \( I \) or static).

Figure 5.3 shows an illustration where the estimated rotations have being used to rectify rolling shutter images in a video sequences [8]. In paper E we use the rotation estimates to rectify image correspondences. These are then used in a standard bundle adjustment frame work to estimate camera poses and reconstruct 3D geometry.

5.7 Rolling Shutter Bundle Adjustment

Using rolling shutter rectification as a preliminary step, as done in paper E, has its advantages. If the assumption of small translations between frames holds reasonably well the approach allows for classical structure from motion approaches to be applied. Additionally, the overhead in computation for a rectification is negligible if compared to e.g. a bundle adjustment step.
5.7. ROLLING SHUTTER BUNDLE ADJUSTMENT

The estimated rotations are here used to rectify a rolling shutter image. The left image is the original and the right is the rectified. These are images and results from [8].

The drawbacks are that zero translation will not only estimate the wrong translation of the camera, it might also result in a less accurate rotation, if points that have been affected by parallax are included. Because of this, the rotation estimate can only be based on point correspondences from a few cameras (normally $\leq 3$ [29]). Since only few images can be used for the pose estimate, there is a risk of the pose estimate being worse than if point correspondences from all images can be used. If a full 6 degree of freedom pose model is used there is no need to limit the number of images used. Secondly, doing a preliminary rectification causes the errors in the rectification to be “locked-down”, which later could degrade the performance of any structure from motion method that follows.

These drawbacks are the subject of paper F where we present a rolling shutter bundle adjustment system, using the full six degree of freedom pose parameterization. The system takes advantage of the sparse structure of the Jacobian which makes it efficient. As in paper E, we use SLERP interpolation for the rotations and linear interpolation for the camera translation. The locality of the linear interpolation keeps the dependencies as low as possible, which in its turn keeps the Jacobian less populated and hence reduce the calculations.

An example of the sparse Jacobian $J$ and $J^T J$ for rolling shutter bundle adjustment is shown in figure 5.4. Observe the impact of the dependencies reducing the sparseness compared to the matrices in figure 5.1. The 2x12 matrix block size on the left side of the Jacobian $J$ is a result of image points now being dependent on two camera (key-) poses instead of one as in the classical bundle adjustment.

When dealing with video sequences, the camera pose will be similar for the last row of one image and the first row of the next image, because they are captured at approximately the same time ($\tau_d \ll \tau_r$). This observation allows us to model the pose as an interpolation between the current key-pose and the key-pose from the next image. The “re-use” of a key-pose saves us one key-pose per image, similar to what is done when solving for the key-rotations with equation 5.6.
Figure 5.4: This figure is similar to figure 5.1 but for a rolling shutter system with linear interpolation. Left: The Jacobian $J$. Right: The product $J^T J$

Key-poses that are used over two images show up in figure 5.4. Where normally one camera pose and one 3D point would only have dependencies in one image, forming one 2x6 block of non-zero elements in the Jacobian, here they instead have two 2x6 block. The “re-use” of key-poses results in a denser but smaller Jacobian matrix and when modeling the motion as one key-pose per frame, there is only one extra pose compared to the global shutter bundle adjustment formulation. This extra pose is from the last row of the last image.

In paper F we show how to solve the rolling shutter bundle adjustment, and we also demonstrate how to make use of the mentioned sparsity. We further show that this approach is superior to the previously presented method in section 5.6 and the computational overhead compared to a standard bundle adjustment solver is well below a factor of two.
Chapter 6

Concluding Remarks

6.1 Result

A focus of this thesis is on real-time pose estimation, where the first method that is presented uses a set of images with known camera poses associated to them. This allows a system to be trained to estimate a pose from a new view in an efficient way.

An alternative pose estimation approach is presented for cases where no prior information is available. Three of the contributions can be connected to this approach, first we present a fast KLT tracker which runs on the GPU, enabling real-time performance and saves valuable CPU cycles. Next we show a new pose estimation method which uses the minimal case of five point correspondences (e.g. from the KLT-tracker) in two views to solve the pose estimation problem. This approach is based on a non-linear least squares formulation and is shown to be a competitive alternative to a closed form solver. Additionally we show how to improve accuracy or performance when a minimal case solver, such as the five point method, is used within a RANSAC framework.

We believe that CMOS sensors with rolling shutters will remain the dominant sensor technology for the mass market for a long time to come. That is why some effort is made towards solving the structure from motion problem for this kind of sensors. This thesis demonstrates two systems which are able to do structure from motion on rolling shutter cameras. This is to our knowledge the first demonstrations of such systems that are able to handle real data from rolling shutter cameras. Within this part we believe the rolling shutter bundle adjustment method to be the most important contribution, where we for the first time show how to efficiently optimize over a complete system of rolling shutter cameras and points.

6.2 Future Work

We have demonstrated one method for which a non-linear least squares formulation is able to compete with a closed form formulation. There is a large set minimal
problems in computer vision for which some of them might be efficiently solved with similar approaches. An interesting aspect of using a non-linear least squares solver is that it is usually easy to extend them to include more than the minimal set. It is interesting to see if an additional final estimate (e.g. for all inliers) of the pose could help before bundle adjustment is applied.

Structure from motion estimation from a rolling shutter camera is still not even close to being as stable as from a global shutter camera, and there is still many interesting aspects to investigate. We believe more stability will come from the use of a more complex interpolation scheme. Another interesting improvement is to alter the frequency of key-rotations and key-translations. It might be beneficial to allow a higher frequency of key-rotations than key-translations, but it may also happen that the estimate could improve if there where fewer key-positions than frames.
Bibliography


