Large scale semantic 3D modeling of the urban landscape

Esteban Lopez, I.

Citation for published version (APA):
Esteban Lopez, I. (2012). Large scale semantic 3D modeling of the urban landscape

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Modeling and understanding large urban areas is becoming an important topic. In a world where everything is being digitized, a semantic and accurate 3D representation of a city can be used in many applications such as event and security planning and management, assisted navigation, autonomous operations, city surveying or urban planning to name just a few.

Different sources of information can be used for this purpose. Cameras at the aerial or ground level, laser scanners, satellite imagery or geospatial information data are common and widely used. In particular, images based approaches have received a lot of attention given the flexibility and low cost of the cameras used to record the scene.

Given the complexity of the complete process, difficulties arise mainly in three stages: the decomposition of the image based reconstruction process, the accurate estimation of the camera positions and the semantic interpretation of the resulting model.

This thesis deals with the problem of large scale city-scale reconstruction and modelling using a monocular camera. The goal of the research was twofold:

Firstly, to obtain an accurate, fast and inexpensive method to perform 3D reconstruction. Secondly, to obtain a semantic model of the reconstructed environment.

We decompose the reconstruction procedure and design a processing pipeline for 3D reconstruction of urban areas by exploring the range of algorithms and methodology choices. By careful reasoning and comparison of state-of-the-art methods we are able to optimize the results of the algorithms involved. We propose an algorithm for estimating optimally and in closed form, the scaled translation of a camera with as little as one correspondence between the 3D space and the 2D image space. Finally, we approach the problem of semantic modeling of large urban areas by merging information from different sources to reach a detailed building level description.
Large Scale Semantic 3D Modeling of the Urban Landscape

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit van Amsterdam op gezag van de Rector Magnificus prof. dr. D.C. van den Boom ten overstaan van een door het college voor promoties ingestelde commissie, in het openbaar te verdedigen in de Agnietenkapel op dinsdag 4 december 2012, te 14:00 uur door

Isaac Esteban Lopez

geboren te Madrid, Spanje
This work was carried out within a large project for event reconstruction based on many video recordings (FES-camera-3D) that was funded by the Ministry of Economical Affairs in the Netherlands, and coordinated by the Netherlands Forensic Institute. The research presented in this thesis was supported by the Netherlands Organization for Applied Scientific Research (TNO). It was performed at TNO Defence, Security and Safety, The Hague, The Netherlands and at the Intelligent Systems Laboratory Amsterdam (ISLA) of the University of Amsterdam, Amsterdam, The Netherlands.
Abstract

Modeling and understanding large urban areas is becoming an important topic in a world were everything is being digitized. A semantic and accurate 3D representation of a city can be used in many applications such as event and security planning and management, assisted navigation, autonomous operations, city surveying or urban planing to name just a few.

Different sources of information can be used for this purpose. Cameras at the aerial or ground level, laser scanners, satellite imagery or geographical information data are common and widely used. In particular images based approaches have received a lot of attention given the flexibility and low cost of the cameras used to record the scene.

Given the complexity of the complete process, difficulties arise mainly in three stages: the decomposition of the image based reconstruction process, the accurate estimation of the camera positions and the semantic interpretation of the resulting model.

This thesis deals with the problem of large scale city-size reconstruction and modeling using a monocular camera. The goal of the research was twofold. Firstly, to obtain an accurate, fast and inexpensive method to perform 3D reconstruction. Secondly, to obtain a semantic model of the reconstructed environment.

We decompose the reconstruction procedure and design a processing pipeline for 3D reconstruction of urban areas by exploring the range of algorithms and methodology choices. By careful reasoning and comparison of state-of-the-art methods we are able to optimize the results of the algorithms involved. We propose an algorithm for estimating optimally, and in closed form, the scaled translation of a camera with as little as one correspondence between the 3D space and the 2D image space. Finally, we approach the problem of semantic modeling of large urban areas by merging information from different sources to reach a detailed building level description.
To my family and friends, who always supported me in the long way to Amon Amarth.
When I was a kid, I wanted to be a scientist, the crazy type, with white hair, thick black glasses and a white lab coat. So I guess my affinity to science was there from the very beginning. Unfortunately when you grow up you realize science is not like in the movies, where a crazy scientist performs crazy experiments that will change the world. Science is a bit duller than in the movies, though it is also a bit more exciting. You get to learn and study things that many people cannot (and does not want) to understand. Science gives you the feeling that you are making the world better, if only marginally. So there you go, the first acknowledgement for the work I did for my PhD is to my childhood, that was in general happy and relaxed and of which I have great memories. Of course part of that will be to be a scientist was later fostered by my parents, who instead of getting me stupid Christmas presents, got me chemistry, astronomy, geology or engineering games. Thank you Mom and Dad. Naturally those games only take you so far, exactly until the end of the instructions, then they become predictable and boring. That’s when my curiosity kicked in. I’ve built alarms for my bedroom from scrap parts, I’ve taken my scooter apart (to make it better of course) to the dismay of my father and I have built a tattooing machine out of parts of an old phone when I was barely a teenager.

The second acknowledgment will probably be lost in the wind, since it is dedicated to two great teachers I had during my high school years: my language teacher Amelia and my chemistry teacher "El Mol". They will most likely never read this, but they were a strong influence that ultimately brought me to becoming a Doctor.

Also from my high school days I want to acknowledge my old friend Alberto, with whom I don’t talk as much as I should. He was the nerdy type all the
way, with a fantastically twisted sense of humor and a remarkable love for complicated mathematics. Thank you Alberto for the inspiration to be top of the class.

Fast forward and I’m already in The Netherlands. With an enormous culture shock and utterly alone in Amsterdam, I spent half the winter studying trying to cope with all the new lectures, the books, the language and the coolddd.... This acknowledgment is to two great study partners Victor and Nako, with whom I wrote software, I got robbed, I built robots, I got an enormous sofa stuck in a staircase and I had lots of fun.

From my first year in Amsterdam I want to acknowledge my neural nets teacher Daan van den Berg. Thank you Daan, I really enjoyed your non conventional lectures and got to understand a complex problem of network organization by means of abstract thinking.

By then, I was supposed to come back to Spain, since my original plan was to finish my BSc and then head back to the mother land. But those influences made me crave for more. More math, more abstract thinking, more lectures and more learning. And so I enrolled at an MSc degree in Artificial Intelligence at the University of Amsterdam. Those two years went fast, really fast. My first lecture on Machine Learning was a true punch in the gut. I thought I had enrolled in Chinese instead of Artificial Intelligence. Thank god I met Markos, who overtime became a good friend and the best study partner one could ever dream of (as well as a fantastic researcher). Together we fought the assignments, the lectures, the Amsterdam life and the not so great study partners we had to take every now and then. Thank you Markos for our time together, in some ways I would not be a Doctor without you.

The last portion of my MSc degree was exciting since I was not following any lectures, but actually making (sort of) scientific work. This acknowledgment is to my supervisors Olaf, Zoran and Ben. With them I learnt about SLAM, mapping, processing thousands of images and building filters to estimate data. From the three, it was Olaf with whom I had the longest and most interesting discussions. Thank you Olaf for teaching me so much.
And as simple as that, I found myself with an MSc degree, not wanting to go back to the corporate life and still craving for more learning. A couple of interviews with who eventually became my promotor and supervisor and I was enrolled in the treacherous path of a PhD degree. I won’t lie to you, the path to mount Doom is a lonely and dangerous one. However, it is also full of joy. It is a bit like a roller coaster, sometimes you are high up, almost touching the clouds, but then the downhill kicks in and you have a tremendous urge to vomit. The path to becoming a Doctor is a learning path, that is for sure. I never though I could learn so much, in so many aspects that are not necessarily linked to Science. During the 5 long years, I met many people that I need to acknowledge for their support in some way or another. Since the list is kind of big, I will just start following my instinct.

Judith was my daily supervisor for those long years. Thank you Judith for your almost infinite patience, your support and understanding. Thanks to you I’ve visited many countries, I’ve gone to an amazing summer (really, with sun and all) school in beautiful Sicily, I’ve had a chance to meet amazing people and work in the things I found interesting.

Prof. Frans Groen was my Promotor. Even though our interaction was not on a daily basis, now after all the work is done, I can see how his tremendous experience played an important role in me writing this book. Without that long term vision I would not have finished. He always knew where everything was going. Thank you Frans for your support and guidance.

Leo . . . my last torturer . . . I mean supervisor. Despite my occasional mixed feelings I want to thank you deeply for your thoroughness and strict scientific attitude, which ultimately made my work much better. I’m sure all those trees we had to kill to accommodate all your comments will resent you forever, but I will be grateful for ever too. Thank you, really.

Now that all the big three are acknowledged, I want to thank the rest of the people that played an important role. First of, my good friend Carsten(ito). Thanks a lot for the ping pong we played together with ideas (sometimes it was more ping than pong), for the unconditional listening, for the fantastic
motorbike trips, to the motorbike maintenance we did together, for the climbing (which was THE way to vent steem) and for being a great friend. On a related note, I want to acknowledge my other big motorbike partners, The Stig and Jens. Guys, thanks for being so nerdy and making the bike trips to great.

From my time in Switzerland, I want to thank my colleagues there, Christoph, Horesh and Olivier. Thank you for making my stay in EPFL a great experience.

Another remarkable person from TNO was Ronald. Thank you Ronald for having an open mind and for your unconditional support.

I also want to thank my first friend in Amsterdam, Wouter. Thank you Wouter for our coffee times, our discussions to make the world better and for being a great friend.

Of course my sister would kill me if I would not acknowledge her here. So thank you Mongui for (almost) always supporting me and helping me through the difficult bits. Also, thank you Lola for being always the unconditionally happy one. Also part of my family, thank you Cruti for always telling me "you are almost there".

Finally, but most importantly, thank you Surfi, for so many things that I cannot fit them in a full thesis. Love you.
Contents

List of Figures xiii

List of Tables xix

1 Introduction 1
   1.1 From 3D to 2D, the Camera Paradigm . . . . . . . . . . . . . . . . . . 3
   1.2 From 2D to 3D, the Reconstruction Process . . . . . . . . . . . . . . . 4
   1.3 The Challenges Ahead and Our Contribution . . . . . . . . . . . . . . . 6
   1.4 Overview of this Thesis . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2 Essential Concepts in Visual Geometry 9
   2.1 Formalizing the Geometry of the Imaging Process . . . . . . . . . . . .10
      2.1.1 The Pinhole Camera Model . . . . . . . . . . . . . . . . . . . . . 10
      2.1.2 Real Life Imaging Devices and the Lens Problem . . . . . . . . . 13
         2.1.2.1 Barrel and Pincushion Distortion . . . . . . . . . . . . . . 14
         2.1.2.2 Radial distortion . . . . . . . . . . . . . . . . . . . . . . 15
      2.1.3 Geometry of Projection and the Camera Motion . . . . . . . . 15
   2.2 About Referencing Cameras, Images and Objects . . . . . . . . . . . .18
      2.2.1 Camera Reference Frame . . . . . . . . . . . . . . . . . . . . . . 18
      2.2.2 Image Reference Frame . . . . . . . . . . . . . . . . . . . . . . 19
      2.2.3 World Reference Frame . . . . . . . . . . . . . . . . . . . . . . 19
   2.3 Some Computer Vision Concepts . . . . . . . . . . . . . . . . . . . . .20
      2.3.1 Image Features . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
      2.3.2 Image Feature Matching . . . . . . . . . . . . . . . . . . . . . . 21
   2.4 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
CONTENTS

3 History and State-of-the-art in 3D Reconstruction
   3.1 User Guided Reconstruction
   3.2 Automated Reconstruction
   3.3 Large Scale Reconstruction
   3.4 Discussion

4 The 3D Reconstruction Pipeline
   4.1 Six Step Approach
   4.2 Step 1: Calibration of Internal Parameters
   4.3 Step 2: Estimating the Camera Motion (Egomotion)
      4.3.1 Image-to-Image Approaches
         4.3.1.1 The Essential Algorithm
         4.3.1.2 Normalization and Constraints
         4.3.1.3 The 5-point algorithm
         4.3.1.4 Outliers
      4.3.2 Image-to-World Approaches (PnP)
         4.3.2.1 The Essential Algorithm
         4.3.2.2 Discussion
      4.3.3 The Scale Challenge
   4.4 3D Reconstruction of Image Features
      4.4.1 Linear Method
         4.4.1.1 Dense Reconstruction
         4.4.1.2 Discussion
   4.5 Optimization
      4.5.1 Local Optimization
      4.5.2 Global Optimization
      4.5.3 Discussion
   4.6 Modeling
      4.6.1 Surface / Mesh Fitting
      4.6.2 Primitive Fitting
      4.6.3 Discussion
   4.7 Semantic Modeling
   4.8 Conclusions
5 Optimal Local Scale in Camera Motion

5.1 Related Work .................................................. 63
5.2 Definitions and Notation .................................... 64
  5.2.1 Disturbed Image Features ............................... 64
  5.2.2 Disturbed Point in Space ............................ 64
  5.2.3 Disturbed Translation ................................. 65
  5.2.4 Disturbed Rotation ................................. 65
5.3 Robust and Scale Consistent Motion Estimation ............ 66
  5.3.1 Robust Frame-to-Frame Motion ....................... 66
  5.3.2 Computation of the Local Scale - Least Squares .... 67
  5.3.3 Solving for the Scale $s$ ............................ 68
    5.3.3.1 Solving with Method 1 .......................... 69
    5.3.3.2 Experimental Motivation ....................... 69
5.4 Error Propagation in Scale Computation ..................... 72
  5.4.1 Error Propagation in the Computation of $A^+$ ....... 72
  5.4.2 Error Propagation in computation of $A$ ............... 74
  5.4.3 Error Propagation in Computation of $b$ .............. 74
  5.4.4 Summary and Conclusions on Error Propagation ....... 74
5.5 Computing the Local Scale Optimally ....................... 75
  5.5.1 Considerations .......................................... 75
  5.5.2 Maximum Likelihood Estimation ....................... 75
5.6 Experiments and Results .................................. 76
  5.6.1 Comparison with State of the Art Methods ............ 76
    5.6.1.1 EPnP: ........................................ 76
    5.6.1.2 P1.5P: ...................................... 77
  5.6.2 Outdoors Visual Odometry ............................ 79
5.7 Conclusions ................................................ 84

6 Semantic Modeling of Tightly Packed Cities ................. 85

6.1 Related Work ............................................. 86
  6.1.1 Discussion and Approach ............................. 86
6.2 Sources of Information ..................................... 91
  6.2.1 GIS and the Outline Polygons ....................... 91
## CONTENTS

7.2.7.3 Reconstruction and Modeling ........................................ 131
7.3 Conclusion ........................................................................... 132

8 Conclusions ........................................................................... 135
  8.1 Future Work ....................................................................... 140

9 Sources of the Chapters ............................................................ 143

10 Summary .............................................................................. 145

A Sample Code ......................................................................... 149

References .............................................................................. 151
List of Figures

1.1 [LEFT] Monocular vision where backtracking a ray is not enough to identify the position of the object in space. [RIGHT] Stereo vision where backtracking two rays intersect in the object in space. ........................................ 5

2.1 LEFT: Pinhole camera model. Rays of light reflected from the real object pass through a small opening in the camera and intersect the imaging sensor. The image is therefore recorded flipped both vertically and horizontally. RIGHT: Camera as a central projection. Rays of light reflected from the real object are projected through the center of projection onto the imaging sensor. .................................................. 11

2.2 LEFT: One axis of the world coordinate system intersects perpendicularly with the image plane at the Principal Point. In this setup, the origin of the image coordinate system coincides with the intersection point. RIGHT: In this setup, the Principal Point and the origin of the image coordinate system do not coincide. ............................................... 12

2.3 LEFT: Barrel distortion. RIGHT: pin cushion distortion. .................. 14

2.4 World, image and camera reference system of our choosing. ............. 20

4.1 3D reconstruction and modeling pipeline. The methods discussed in each of the steps are indicated. .......................................................... 37

4.2 Original (LEFT) and undistorted (RIGHT) images after performing camera calibration. ................................................................. 39
LIST OF FIGURES

4.3 LEFT: Reconstruction through intersection of rays between camera center and matching image points. RIGHT: Rays spanning between the camera center and matching image points generally do not intersect in space. It is common to assume that the solution will lie on the shortest line connecting the two rays. ........................................ 48

4.4 Matching of polygonal planes to a raw point cloud. .............................. 57

4.5 Textured set of polygonal planar patches. .......................................... 57

5.1 The scale problem in monocular vision. Estimating the camera pose $C_1$ is intrinsically scale free. By sliding the camera over an unknown amount along the translation direction $t_{[0,1]}$, both structures $S$ become indistinguishable reconstructions, only the scale is different. If three cameras are considered, the local scale accounts for the ratio between the scales of the two pairs of cameras (there are three pairs of cameras, though only 2 of the scale ratios are linearly independent). The global scale accounts for the scale of real life objects. .......................................................... 62

5.2 Error on the estimation of the scale using the 4 proposed methods (2000 runs). Method 2 is the worst given that the motion on the $y$ direction does not contribute to the computation of $s$. Methods 1 and 3 offer similar performance, with a slight preference for method 1. Method 4 is slightly better than the rest. .......................................................... 71

5.3 Error on the estimation of rotation and translation of a third camera pose given 2 previous camera poses and the reconstructed structure using linear triangulation (2000 runs). TOP: error in the rotation calculated as absolute angle of $\Delta R$ in 5.7. The 1-point coincides with 1.5 because the rotation is calculated with the 8-point algorithm. MIDDLE: error in the translation vector calculated as the distance of the difference vector between ground truth and estimation. BOTTOM: error in the angle estimation of the translation direction. NOTE that 1-point (Least Squares) and 1-point (Optimal) are the same for translation and rotation accuracy. Moreover 1-point and P1.5P are the same for rotation accuracy. ........ 78

5.4 Sample images of first dataset. ............................................................. 80
5.5 Top: Estimated trajectory (red circles) and recovered point cloud for first set. Bottom: reconstructed 3D model. No global Bundle Adjustment was necessary.  
5.6 Top view of the reconstructed scene (in black) and camera positions (red circles).  
5.7 Sample images of second dataset.  
5.8 Estimated trajectory (red circles) and recovered point cloud for second set. The camera started at the right-bottom corner moving towards the right side and ending in the same position. This was ensured using the same image for first and last position. No global Bundle Adjustment was applied.  
5.9 P11 dataset consisting of 11 images.  
5.10 3D reconstruction of P11 benchmark dataset. No bundle adjustment was used and the scale between consecutive images was computed using our optimal algorithm.  
6.1 Pipelines for Semantic Modeling of tightly packed cities using GIS, point clouds and aerial images.  
6.2 Colored GIS data obtained from OpenStreetMap.org.  
6.3 Palais de Rumine. BLUE: OpenStreetMap data. RED: Manually enhanced polygons.  
6.4 Point cloud obtained from aerial images.  
6.5 Point cloud obtained from ground images.  
6.6 Aerial Image obtained from Google Maps.  
6.7 Average error over 10 runs in the estimation of gravity as a function of the number of planar patches used.  
6.8 LEFT: facade points project near the building outline. RIGHT: roof points project inside the building outline.  
6.9 Representation of the error function for optimization.  
6.10 Ground surface estimation process: TOP-LEFT. Height-map initialization. TOP-RIGHT. Large declination smoothing. BOTTOM-LEFT. Isolated patch removal. BOTTOM-RIGHT. Final smoothed ground height.  
6.11 Validation of the estimated ground surface.
LIST OF FIGURES

6.12 Three stage procedure for obtaining the height of facades. TOP-LEFT: distribution of point-facade distances over height and distance. TOP-RIGHT: exponential function used to grade points based on their distance to the facades. BOTTOM: resulting graded distribution. 105

6.13 Summed up distribution of the height of a single building (BLUE) and derivative (RED). 106

6.14 Estimation of the height of facades for 3 different buildings. 106

6.15 Comparison between estimated facade height and real building. 107

6.16 LEFT: The aerial image and the outline of the polygon of a building is shown. Red crosses represent the Harris corners found in the image. Blue circles are the corners that lie inside the polygon. RIGHT: triangulated vertices. 109

6.17 Top view of the registered sources of information. Green shows the aerial point cloud. Red shows the ground based point cloud. Blue shows the GIS data. 111

6.18 Top view of the textured ground plane and the GIS outlines of buildings. 112

6.19 Google image used for texturing the estimated ground surface. 113

6.20 Views of the model of the city center of Lausanne where the height of the buildings has been estimated. 114

6.21 Area: Model of the city of Lausanne, color is assigned to buildings based on the area they occupy, from smaller (blue) to bigger (red). 115

6.22 Height: Model of the city of Lausanne, color is assigned to buildings based on their height, from smaller (blue) to bigger (red). 115

6.23 Volume: Model of the city of Lausanne, color is assigned to buildings based on the volume they occupy, from smaller (blue) to bigger (red). 116

6.24 Two views of the complete model of the city of Lausanne. Texture has been applied to the rooftops and the ground surface. 117

7.1 3D modeling pipeline. Some well known alternatives for each step are pointed out. 120
7.2 LEFT: distorted image of a street where lines that are straight in real life can be seen. Three of those straight lines consisting of three points are selected and can be seen in red. RIGHT: the selected lines are used to compute the radial distortion parameters which are then used to undistort the original image resulting in the undistorted image of the street.

7.3 TOP: 4 input images of a mountain view in Switzerland. BOTTOM: stitched images after computation of the image-to-image homography. Vertical lines visible on the right side of the image are caused by two factors: slight differences in the automatic camera setup due to changing conditions when measuring the light in every image; the camera translating slightly on every image instead of only rotation.

7.4 TOP: 3 views of a street out of 57 images taken the camera roughly pointing towards the buildings. BOTTOM: Top view of the camera positions (in red) and reconstruction of the SIFT features used to compute the poses.

7.5 Black pyramids: representation of the position of cameras 1, 2 and 3 before scale adjustment. Grey pyramid: camera 3 after scale adjustment. Firstly, the relative displacement between cameras 1 and 2 is computed using, for instance, the 8-point algorithm. This yields a scale free motion of camera 2 with respect to camera 2. Then the displacement of camera 3 with respect to camera 2 is calculated. This yields a scale free displacement represented as the black pyramid in camera 3. Those two camera displacements are on a different scale. We set the reference scale as 1 for the first camera displacement and the scale of the second camera displacement between cameras 2 and 3 is computed, resulting in the scale adjusted position of camera 3 represented as the grey pyramid.

7.6 Reprojection error (as the sum of squared distances between the reprojected points and the original image features) during local refinement. After only two iterations the error is close to a minimum, making the procedure very efficient.
7.7 TOP: Images used to obtain a scaled consistent motion. BOTTOM-LEFT: resulting 3D structure before optimization. BOTTOM-RIGHT: resulting structure after global optimization. The originally estimated position of the cameras (before optimization) was altered to produce a noisy pointcloud.

7.8 TOP-LEFT: Colored point cloud. TOP-RIGHT: Set of planar patches found with the RANSAC approach. BOTTOM-LEFT: Colored set of planar patches. BOTTOM-RIGHT: Dense reconstruction using PMVS2

7.9 A few undistorted views of a street.

7.10 Triangulated SIFT features and scale consistent camera trajectory. The images are shown in the computed position.

7.11 TOP: Colored planar patches. BOTTOM: Textured patches.
List of Tables

4.1 Reconstruction steps and our choices of methods for each step. .................. 59
5.1 Algorithm comparison .................................................. 77
1

Introduction

Creating and understanding digital 3 dimensional models of large urban areas is becoming an important topic both in science and industry. A semantic and accurate representation of a city can be used in many applications such as assisted navigation, autonomous operations, city surveying, forensic investigation or urban planning, to name just a few. There are many possibilities with respect to how to obtain the necessary information to build a model. Naturally, one could approach the problem by using simply user input. This is however a painstaking job that requires advanced knowledge of modeling software along with knowledge of the environment. Alternatively, some source of information could be used to guide the reconstruction. This information could exist as a set of measurements or geometric constraints that describe in some way the layout of the environment. Some sources of information include data recorded by lasers, ground images, aerial or satellite imagery and Geospatial Information Sources (GIS).

Laser data is an accurate source of information, where the device records distance measurements between itself and the objects in the environment. These measurements can then be translated into points in 3 dimensions. If we obtain sufficient number of points, we have a point cloud in 3D space. Laser devices are however expensive machines, difficult to use and slow in recording. Also, the measurements typically consist only of a set of distances to points, though some state of the art laser models also record color data for each point. These sets of points are merely geometric representations and cannot be considered a model. We define a semantic model as a 3D representation that
1. INTRODUCTION

contains information about the environment that is not purely geometrical, namely: the
direction of gravity, the shape of the ground and a geometrical model of buildings.
Images can also be used to obtain information. We, as humans, are used to creating,
understanding and interacting with a world in 3D. Based mostly in our ability to see
our surroundings, we create a mental representation of the scene or object in front of us
where we can establish spatial relationships such as depth, position or shape. Recording
images is also cheaper and easier than operating laser devices. Therefore vision based
methods for reconstructing a scene in 3D have received much attention in the current
research. Single-camera, or monocular, systems are particularly popular given their
flexibility and low cost. Some of the first attempts try to mimic the stereoscopic nature
of the human vision.

GIS data typically consist of 2 dimensional information of urban areas. It contains
information about buildings, streets, public services, etc. Professionally made GIS data
is expensive since it is manually created by experts employing accurate GPS data.
Some freely available and user-contributed initiatives exist such as the Open Street
Map projects where users manually annotate information about the city. GIS data is
typically geolocalized though only 2 dimensional.

This thesis deals with the problem of large scale city-size reconstruction and modeling
from a monocular camera. The goal of the research was twofold. Firstly, we wanted
to obtain an accurate, fast and inexpensive method to perform 3D reconstruction. Sec-
ondly, we wanted to obtain a semantic model of the reconstructed environment.

Given the flexibility, availability and low cost of consumer cameras, we focus on the use
of monocular off-the-shelf cameras to model urban environments. This will render our
methods flexible and inexpensive but also bring certain algorithmic issues that need to
be addressed. The characteristics of the monocular urban setting are:

- An urban environment consist of man-made structures such as roads and build-
ings.

- We are interested in modeling buildings and roads, avoiding movable objects like
cars or pedestrians and ignoring smaller structures such as mailboxes or lamp
posts.

- Sequences of images are recorded while the camera is moving.
1.1 From 3D to 2D, the Camera Paradigm

- In order to capture as much of the buildings and roads as possible, while maintaining overlap between consecutive images, these are recorded pointing diagonally towards the buildings.

- The camera therefore moves along the streets, with a mostly forward motion.

- Facades and ground consist mostly of planar structures.

- Given the large overlap between consecutive images, the reconstruction process is likely to produce a large amount of points.

Given the extent of the field, there are many important questions for which we want to find an answer. What are the best algorithms? How many steps are in the reconstruction procedure? How do the choices that we make in each step affect the overall result? Is it possible to obtain accurately the positions in space of the images used for the reconstruction? Do we need an optimization procedure to refine the estimation of the model? Which representation to use? Are colored points in space a sufficiently rich descriptor for a scene? And finally, how do we bridge the gap between representation and understanding?

In the remaining of this chapter, we first explore the camera paradigm and explain the basic idea of an imaging process. During this procedure, we reduce the world from 3 dimensions to just 2 dimensions. Then we introduce the basic problem of using the recorded data to obtain a 3 dimensional representation, effectively restoring the third dimension. Finally, we state some of the challenges along the way and give an outline of this Thesis.

1.1 From 3D to 2D, the Camera Paradigm

The origin of the word camera is the Latin camera obscura, referring to an optical device to project images on a screen. It was originally used for entertainment or, when projecting onto a piece of paper, as an aid to trace the scene easily. The device, in its outmost simplicity, consist on a dark chamber with a tiny hole in one side. The light rays pass through the hole projecting on the opposite wall. This produces an image upside down and it is the origin of modern cameras and the imaging process.
1. INTRODUCTION

The next step of these simple camera devices consists of placing one or more lenses in front of the hole. The word lens derives from the Latin word for lentil because of its shape. The oldest known artificial lens is dated more than three thousand years old. Lenses change the trajectory of light rays. This can have a magnifying effect on the resulting image, or a focusing effect where the light rays are concentrated in a small point (which can be used, for instance, to start a fire). When placed in front of the camera hole, the effect is translated in the way the image is displayed on the projecting surface. Effects such as zooming or distorting the image can be obtained. Modern cameras use lenses mostly to improve the image quality by allowing more light while keeping the image in focus. They are also used to zoom in/out.

A complete camera device consists of a lens placed in front of the hole where the rays of light pass through, projecting on a surface beyond. This procedure, called projection in mathematical terms, creates a representation of the original object where the number of dimensions is reduced from 3 to 2. The projection procedure is explained in Chapter 2.

1.2 From 2D to 3D, the Reconstruction Process

Understanding the imaging process and the dimensionality reduction from the real scene to the image is only the beginning of the path towards 3D reconstruction. The true and intriguing puzzle starts by wondering if this imaging process can be reversed. By reversing it we expect to recover the local depth of the image so we can obtain a 3 dimensional representation of the scene, or as we described it before, a 3D reconstruction.

We humans create a mental representation of the things that we see. This representation allows us to infer spatial relationships such as depth, shape or geometrical relationships. Since our eyes work in a similar way as a camera, we can almost be certain that the process must be somehow reversible. There is however, a small difference with respect to a camera as we have described it. We have two eyes, or two cameras that record the scene at the same time from two different points of view. This makes us stereoscopic camera systems. So even when we look at a static scene without moving our head, we perceive and understand depth, which is the basis of a 3D reconstruction.

The human ability to understand and process depth information is out of the scope of this thesis, partly because our mighty brain does not only use images but also details
stored in our memory and based on our previous experiences. However, if we explore the very basic principles of human vision geometry, we hope to use this information to infer methods and techniques to retrieve the 3 dimensional shape of imaged objects.

Complete details on camera models are given in Chapter 2. However, simple intuition goes a long way. The object projection onto the camera is simply a ray of light bouncing from an object and then onto the camera recording sensor. So in principle, if we backtrace the trajectory of the ray, we can be certain that we will find the object at some point. The evident problem is that we cannot know the distance to the object, we can only be certain of the direction in which it can be found. If we now consider the fact that we have two cameras mounted in our head, we could backtrace the light rays that bounced off the same object point and impacted on both camera sensors. This principle is shown in Figure 1.1. Then we can be certain that the backtracked rays will intersect in space where the object is. This process is called triangulation and a basic principle in 3D reconstruction.

![Figure 1.1:](image)

We now know that if we have two cameras, for which we know the position and orientation, and if we are able to identify an image point in both cameras that corresponds to a ray of light that bounced off the exact same spot in an object, we can then backtrack those ray trajectories to find the object and therefore estimating its position in a 3 dimensional space. Even though this is only the intuition behind the reconstruction
1. INTRODUCTION

process, it serves well as an introduction to the problem of obtaining a 3 dimensional meaningful representation of an imaged scene.

We will use only one camera; fortunately, under many circumstances the same procedure could be applied in this case. If we set the constraints and assume the scene is static but we are able to move the camera and then retrieve its position, we can still perform the reconstruction by the backtracking method.

This intuition already sets the requirements of a 3D reconstruction using single camera systems:

- A camera that is able to record the scene by projection of rays of light
- The means to move the camera around a static environment
- A method to estimate the position of the camera in space
- The ability to identify corresponding projected rays (or corresponding image points)
- A backtracking method to find the object in space

Having understood the camera recording process and the intuition behind the reconstruction process, in the following section we detail the most important challenges that we will face in 3D reconstruction.

1.3 The Challenges Ahead and Our Contribution

In the previous section we sketched a simplified version of the steps of a reconstruction pipeline where cameras are employed to record a scene and the resulting images are in turn used to obtain the reconstruction. However the exact details on the different steps that need to be performed are yet unspecified. Variations in both the pipeline design and choice of algorithms for each of the steps will have a definitive impact on both the quality of the model and the computational complexity. The reconstruction procedure we envisioned would find an imaged object in space, however no meaningful information is obtained.

The challenges ahead are focused on those two areas: the accurate estimation of the position of cameras in space, and the modeling of reconstructed scenes.
State-of-the-art methods approach the problem of reconstruction and camera position estimation as a single procedure. Computationally expensive numerical optimization methods are typically employed that aim at solving a set of geometrical constraints. Our contribution for the first challenge focuses on the analysis of the different steps of the reconstruction process, and the tailored design of an urban modeling pipeline where accurate camera positions and reconstructed structures can be found.

Obtaining a reconstruction in 3 dimensions is however not the end of the line. If these models are ever to be used in other tasks besides visually appealing multimedia, some information needs to be inferred. The process of semantic modeling clearly represents the door to highly profitable uses of the reconstructed data, specially when dealing with the city-size reconstructions. State-of-the-art methods applied to urban scenes aim at modeling mostly stand alone buildings. The modeling process is usually done by means of a set of predefined shapes that are fitted to the reconstruction. Topological information is rarely and barely modeled. Our contribution in the area of semantic modeling focuses on obtaining certain topological and physical characteristics of the urban scene, and on modeling the city as a collection of individual buildings with unique properties.

1.4 Overview of this Thesis

The remaining of this Thesis is organized in seven chapters.

In Chapter 2 we specify the visual geometry concepts presented in the previous sections and put them into a mathematical context. We formalize the camera model and the details of the use of lenses. We also define our choice of geometrical reference frames and present the mathematical details of the projection process.

In Chapter 3 we present an extensive literature review, exploring the field of 3D reconstruction categorized based on different aspects such as user driven versus automatic procedures. This review allows us to put our research into context and points to the goals of each of the following chapters.

In Chapter 4 we design the 6-step reconstruction pipeline for large scale urban environments. We introduce the mathematical concepts, motivating our decision for each step of the algorithm. We focus on the development of inexpensive and fast methods to obtain a 3D reconstruction from sets of consecutive images.
1. INTRODUCTION

In Chapter 5 we focus entirely on the mathematical details of the estimation of the scale of the camera motion. We describe the least square solution and our optimal method for the computation of the local scale.

In Chapter 6 we focus on large scale semantic modeling of a city. We explore the different sources of information that are used to create the semantic models and devise a method for the correct registration of the data. We propose a method to accurately estimate the direction of gravity (or vertical direction). This is then used used to estimate the isosurface over which the city is settled. Finally a parametric model of the building and the rooftops is obtained.

In Chapter 7 we introduce our Matlab toolbox FIT3D, which implements most of our work in 3D reconstruction. We analyze the motivations and goals of this publicly available toolbox and compare with state of the art alternatives.

Finally, in Chapter 8 we summarize the contents of this Thesis, present the overall conclusions and point to some directions for future work.
The field of 3D reconstruction and modeling spans the complete process of obtaining a geometric or semantic model of an imaged scene or object. This encompasses not only the methods and algorithms for retrieving the depth of the image but also the deep understanding of the imaging process in its mathematical and geometrical terms. We described in Chapter 1 intuitively how a moving camera can be used to retrieve the 3 dimensional structure of a recorded scene. This procedure is also called Structure from Motion (SfM) \[1\].

The problem has been studied in great detail in the literature. In this chapter we follow the work of Hartley and Zisserman to formalize the geometry of the camera and the imaging process. We also describe the relation between the simplistic camera described in the previous chapter and modern digital cameras by means of a calibration matrix. Once we understand how the camera is defined mathematically we then move on to the actual methods to perform the reconstruction. Furthermore, there are a large amount of different conventions with respect to the geometrical frames. This makes it difficult sometimes to integrate different approaches or steps in one single reconstruction procedure. In this chapter we establish our point of view, defining the referential frames to describe geometrically: cameras, images and objects in space. Finally, we formalize the geometrical projection and describe the relation between corresponding sets of points in images by means of the fundamental and essential matrices. These two contain the essence of the camera motion which can then be used for the actual reconstruction of
the scene. The work of Hartley and Zisserman \cite{Hartley2004} is among the most well known on visual geometry and is used as a general guide, though deep details are left out. The standards and definitions presented in this chapter are used in the rest of this Thesis.

2.1 Formalizing the Geometry of the Imaging Process

Understanding the geometry of the camera implies understanding how the images are formed and how we can backtrack the process for 3D reconstruction. In this section we formalize the pinhole camera model by describing the mathematical projection, the parameters that define the camera and the referential frames used to describe it all.

2.1.1 The Pinhole Camera Model

As we described in Chapter 1, the pinhole camera is the most basic imaging device where rays of light pass through a small hole in one side of a dark chamber. These rays impact the image sensor behind the hole where the image is recorded. The hole is called the \textit{focal point}. The smaller the hole, the sharper the image, but also the more dim it becomes (save for degeneracies with very small holes due to diffraction). The size of the hole is commonly referred as the \textit{aperture} of the camera. The rays cross in the camera hole, producing therefore an upside down image (see Figure \ref{fig:pinhole-camera} LEFT). In the Pinhole Camera Model, the aperture is assumed to be an infinitely small hole and all cameras treated from here on are assumed to be compliant with that model.

Cameras can be geometrically described as a central projection where the real life object is projected onto a recording device through a center of projection, which is of course the focal point. In such a representation, it is common to represent the recording device \textit{in front} of the center of projection \cite{Hartley2004} (see Figure \ref{fig:pinhole-camera} RIGHT). The focal point is also called \textit{camera center} or \textit{optical center}.

The distance of the sensor to the focal point is called the \textit{focal distance} and usually referred as \( f \). This parameter describes the geometry of the imaging process which can be encoded in a \textit{camera calibration matrix} called \( K \). The camera matrix represents a mapping between 3D points in the real world to points on the image sensor.

The projection of objects into the imaging sensor is a linear mapping between 3D points and image points in homogeneous coordinates:
2.1 Formalizing the Geometry of the Imaging Process

Figure 2.1: LEFT: Pinhole camera model. Rays of light reflected from the real object pass through a small opening in the camera and intersect the imaging sensor. The image is therefore recorded flipped both vertically and horizontally. RIGHT: Camera as a central projection. Rays of light reflected from the real object are projected through the center of projection onto the imaging sensor.

\[
\begin{pmatrix}
U \\
V \\
W
\end{pmatrix} \rightarrow \begin{pmatrix}
fU \\
fV \\
W
\end{pmatrix} = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix},
\]

where \((U, V, W, 1)\) represents the homogeneous coordinates of a point of the imaged object in 3D space, and \((fU, fV, W)\) represents the projected homogeneous coordinates of a point in the image space.

At this point, we can already see that there are at least 2 different coordinate systems: the coordinate system of the objects in the real world, and the coordinate system of the projected objects in the 2D image plane. We will describe in more detail the coordinate systems in the following section, but let us for now assume that there are two generic reference frames for both the object in 3D space as seen from the camera and the imaged object in image space, and let us call them the *Camera reference frame* \((C_{rf})\) and the *Image reference frame* \((I_{rf})\).

It is commonly assumed that one axis of the \(C_{rf}\) will intersect the image plane perpendicularly. This intersection point is called the *principal point*. This perpendicularity
2. ESSENTIAL CONCEPTS IN VISUAL GEOMETRY

Figure 2.2: LEFT: One axis of the world coordinate system intersects perpendicularly with the image plane at the Principal Point. In this setup, the origin of the image coordinate system coincides with the intersection point. RIGHT: In this setup, the Principal Point and the origin of the image coordinate system do not coincide.

Assumption is related to the position of the image sensor. In most cameras the sensor is placed parallel to the focal plane. This allows the virtual plane in 3D space where imaged objects are in focus on the image to be also parallel to the sensor. Certain cameras or lenses do not follow this assumption, though their use is limited to creative or architectural photography, creating focus planes in the image that introduce visual effects.

If we could choose the location of the $I_{rf}$, we could set it such that it would coincide with the projection of the $W_{rf}$ in the image (see Figure 2.2 LEFT). However, in most cases, we have no choice in the position of the center of the $I_{rf}$ or it is not convenient to transform it. If the principal point and the origin of the $I_{rf}$ do not coincide, we need a mapping between them:

$$(U,V,W)^T \rightarrow (fV/W + p_u, fV/W + p_v)^T,$$  \hspace{1cm} (2.2)$$

where $(p_u, p_v)^T$ are the coordinates of the principal point in the $I_{rf}$.

Combining this with our previous definition of the projection, we obtain a more general description where the difference between the principal point and the center of $I_{rf}$ are accounted for:
2.1 Formalizing the Geometry of the Imaging Process

\[
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix} = H
\begin{pmatrix}
fU + Wp_u \\
fV + Wp_v \\
W \\
1
\end{pmatrix} = H
\begin{pmatrix}
f & p_u & 0 \\
f & p_y & 0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\] (2.3)

A typical sensor in modern digital cameras is a finite sensor consisting of pixels that capture light rays. These pixels might not be arranged in a square array, which introduces a nonhomogeneous scale factor in each image axis. If we consider the number of pixels per unit of distance in image coordinates \((m_u, m_v)\), we can create a mapping that will account for a finite sensor with nonsquare pixel arrays:

\[
\begin{bmatrix}
fm_u & 0 & p_u & 0 \\
0 & fm_v & p_v & 0 \\
0 & 0 & 1 & 0
\end{bmatrix},
\] (2.4)

where \(fm_u\) and \(fm_v\) are usually written as \(\alpha_u\) and \(\alpha_v\).

There is one more parameter for a complete generalization of the camera model. The skew factor \(s\) can be interpreted as a skewing of the pixel elements in the sensor array so that the x-axis and y-axis are nonperpendicular. This could be the result of the image sensor not being parallel to the optical axis. This is encoded in \(K_{(1,2)} = s\). It is commonly assumed to be zero in most commercial and consumer cameras.

Throughout our work, we assume the use of cameras and sensors with NO skew factor, therefore, we will always consider a camera calibration matrix on the form:

\[
K = \begin{bmatrix}
\alpha_u & 0 & p_u \\
0 & \alpha_v & p_v \\
0 & 0 & 1
\end{bmatrix},
\] (2.5)

where \(\alpha_u = fm_u\) and \(\alpha_v = fm_v\).

2.1.2 Real Life Imaging Devices and the Lens Problem

Modern cameras are different from the pinhole camera in that they have a lens placed at the hole. Lenses are used primarily to get sufficient amount of light in (achieve the desired image quality), to change the focal length (zooming in and out), to change the aperture and control sharpness, and to introduce certain visual effects.

The use of lenses can however introduce a distortion factor in the recorded image. This distortion can take a variety of shapes depending on the quality and construction
2. ESSENTIAL CONCEPTS IN VISUAL GEOMETRY

materials of the lens. Changing the focal length by zooming in and out may also change
the distortion factor. Some of these distortions are used for artistic reasons and some
of them produce the effect of an image sensor that is not perpendicular to the camera
coordinate system.

Most everyday consumer cameras only introduce two types of distortion: barrel and
pincushion.

2.1.2.1 Barrel and Pincushion Distortion

Barrel distortion (see Figure 2.3 LEFT) produces and effect where points are imaged
further from the center as the distance from the center increases. This distortion is
typically found in wide angle lenses where the goal is to record a bigger portion of the
scene, even if distorted. This is the most typical type of distortion found in consumer
lenses in modern cameras.

The pincushion distortion (see Figure 2.3 RIGHT) has the opposite effect of the barrel
distortion. As the distance with the center of distortion decreases, the points are imaged
closer to the center.

Figure 2.3: LEFT: Barrel distortion. RIGHT: pincushion distortion.
2.1 Formalizing the Geometry of the Imaging Process

2.1.2.2 Radial distortion

Both distortions can be seen as a radial distortion, where the location of the true pixel is altered based on some function of the distance with respect to the center of the distortion.

It is important to understand that in order to obtain a linear projection (where a ray can be traced from the imaged point to the real object) into the imaging device, these distortions need to be accounted for. Otherwise, the backtracking and the point matching procedures required for the reconstruction step would be incorrect. In computer vision, the distortion in a lens is measured and the corresponding inverse transformation is applied so that the camera becomes effectively linear projection device.

Radial distortion can be described as a function of the distance to the center of distortion:

\[
\begin{pmatrix}
u_d \\
v_d
\end{pmatrix}
= L(\sqrt{u_l^2 + v_l^2})
\begin{pmatrix}
u_l \\
v_l
\end{pmatrix}, \quad (2.6)
\]

where \((u_d, v_d)\) are the distorted image pixels, \((u_l, v_l)\) are the linearized, or corrected, image pixels.

\(L(r)\) is the distortion function dependent on the distance \(r\) to the center of distortion and is usually approximated by a Taylor series or order \(n\).

\[
L(r) \approx 1 + k_1 r + k_2 r^2 + k_3 r^3 + \ldots + k_n r^n \quad (2.7)
\]

Note however that the distortion function \(L(r)\) differs when modeling the transformation from distorted pixels to linearized pixels, or the other way around. The function is not easily inverted, though an approximation can be found when the distortion factor is sufficiently small. It is then common to model the distortion some way or other depending on the reconstruction procedure and the way the images are further processed.

2.1.3 Geometry of Projection and the Camera Motion

We have precisely defined the camera model, which allows us to map points in space to points in an image. However, we have not defined how the position of the camera (sometimes called camera motion) can be described nor have we integrated it in the projection process, that is, we have assumed that the camera center is at the origin of a World Reference Frame \(W_{r,f}\).
The projection from camera coordinates $X_{\text{cam}}$ to image coordinates $x$, defined earlier in Equation 2.3 can also be described as:

$$x = K[I|0]X_{\text{cam}},$$

(2.8)

with $K$ as in 2.5, $I$ a $3 \times 3$ identity matrix and $0$ a $3 \times 1$ vector of zeros.

This projection operation however only works for cameras that are located at the center of the world. If the camera is located elsewhere, we need to incorporate the transformation between the world center and the camera center in the projection process.

Hartley and Zissermann chose to describe that transformation as the rotation and translation $(R', t')$ that is required to bring the center of the world to the camera center. We chose however the inverse transformation, that is the rotation and translation $(R, t)$ that is required to bring the camera to the center of the world. The reason to chose this representation is that it depicts more accurately the fact that we have a center of the world that is universal and nonmovable, while the camera is the one moving around the environment.

Since we will be dealing with sequences of images it is also sensible to reference the camera centers in a relative fashion. We can then distinguish between the absolute and relative camera poses. The absolute camera pose is described as the rotation and translation that will be required to bring the camera to the center of the world. The relative camera pose is described as the rotation and translation that is required to bring each camera to the previous camera pose.

This relative and absolute description of motion is very convenient for us, as we will see in the following sections. Additionally, we can elegantly accumulate the relative camera motion by a simple matrix multiplication.

The projection operation for cameras not located at the center is then described with respect to the absolute rotation and translation required to bring the camera to the origin of the $W_{rf}$:

$$x = K[R|t]X_{\text{cam}} = PX_{\text{cam}},$$

(2.9)

where we define the camera position, or projection matrix, as $P = K[R|t]$. $P$ is also sometimes called camera motion since in monocular systems it represents the different positions of a camera moving through the environment.
Both absolute and relative motions are described with a $3 \times 4$ matrix containing a $3 \times 3$ rotation matrix $R$ and a $3 \times 1$ translation vector $t$. When expressed in homogeneous coordinates using a $4 \times 4$ matrix with last row $[0^T |1]$ the relative motion of cameras can be accumulated by a simple multiplication.

This definition of the camera position based on the projection process is very intuitive since it encompasses the motion as a rotation and translation in 3D space. However, since we are interested in the motion between images, it is important to define the relation between points in two different images. The epipolar geometry "is the intrinsic projective geometry between two views" [1]. It does not depend on the image itself but rather on the internal parameters and the relative position of the cameras. This projective geometry does not say anything about the 3D structure, contrary to equation 2.9. The relation with the previous camera motion definition is straightforward: if a point $X$ in the real world is projected as $x$ in a first image, and then projected as $x'$ in a second image, then those points must satisfy the following relation defined through the fundamental matrix $F$:

$$x'^T F x = 0$$

(2.10)

The proposed relation however does not state anything about the camera calibration. A more specific relation that accounts for $K$ is the essential matrix $E$, which relates the calibrated points in two images.

Say we consider the calibrated imaged points $\hat{x}$, which are the projection $x$ in the image plane of a point $X$ in 3D space, but accounting for the calibration parameters:

$$\hat{x} = K^{-1} x$$

(2.11)

Then introduce those calibrated image points in equation 2.10 obtaining:

$$\hat{x}'^T K^T F K \hat{x} = 0$$

(2.12)

So the relation between the essential matrix and the fundamental matrix is defined as:

$$\hat{x}'^T E \hat{x} = \hat{x}'^T K'^T F K \hat{x} = 0$$

(2.13)
The difference between the fundamental matrix $F$ and the essential matrix $E$ is that the latter already encodes the camera calibration. This implies a total of 5 degrees of freedom: 3 for the translation, 3 from the rotation minus 1 for the overall scale.

Several solutions for the camera matrices can be retrieved up to a scale ambiguity [1] from the essential matrix using the method from Horn [2]. Horn’s method first extracts the rotation $R$ and then the translation $t$ from $E$ by observing that $E = BR$ where $B$ is a skew symmetric matrix that satisfies $Bv = t \times v$ for all vectors $v$. The method provides four possible solutions ($[R_+, t], [R_-, t], [R_+, -t], [R_-, -t]$).

Choosing the correct one involves a voting approach, accepting the solution where most of the structure can be recovered in front of both cameras.

2.2 About Referencing Cameras, Images and Objects

So far we have assumed some coordinate system to be present to reference both the real life object and the image pixels. It is important to define clearly those coordinate systems since all transformations that need to be applied for the reconstruction will depend on them.

Furthermore, we have introduced the need for a coordinate system to reference real life objects, and one to reference pixel locations in the image. Additionally, when we introduce multiple images we also need to introduce a universal reference frame so we can reference those camera positions with respect to the real world.

2.2.1 Camera Reference Frame

The first reference frame we need is the camera coordinate system $C_{rf}$. It is typically placed to have its origin at the focal point (the center of projection) and we must decide the direction of two of the axes since one is commonly perpendicular to the image sensor. Hartley and Zisserman chose a right hand coordinate system where the z-axis points towards the scene, and the y-axis and x-axis point upwards and to the left respectively (when looking from the center towards the z-axis).

We chose however an x-axis that points to the right while retaining the upward y, effectively adopting a left-hand side coordinate system. We make this choice given that image features are typically represented from left to right. Furthermore, we chose the z-axis to be perpendicular to the image sensor. From this frame, we can reference the
2.2 About Referencing Cameras, Images and Objects

points in space that belong to the object, the imaging sensor position at a distance \( f \) from the center and the size of the sensor.

2.2.2 Image Reference Frame

We also need to define a 2 dimensional local frame \( I_{rf} \) to reference the points in the image. These points will be the pixels in the images that are obtained through the projection mechanism.

It is important to decide if we chose to put the sensor behind the center of projection, or in front. This, along with considerations on the software used to read the images, will determine the selection of our reference frame. Since we chose a left-hand camera reference frame, and we want to avoid an inverted image behind the projection center, we chose to place the sensor in front of the projection center and use the same orientation on the x- and y-axis as in the camera reference frame.

Therefore, the x-axis points to the right of the image as seen on a screen or paper, while the y-axis points towards the top of the image. On the other hand, given that Matlab references the images from the top-left corner, we place the center of the image reference frame in that corner.

2.2.3 World Reference Frame

Finally, having defined the image and camera reference systems, we need one more coordinate system \( W_{rf} \) to be able to reference multiple cameras in space. This is the world reference coordinate system and it is used as an absolute coordinate system from which cameras or objects can be referenced in space.

This center is effectively the center of our reconstruction universe. It can be placed arbitrarily anywhere. In a scenario where multiple cameras are used, it seems natural to choose the center of the world to be placed coincident with one of the camera centers.

Since we are concerned with sequences of image, we chose to place the center of the world at the position of the first image of a given set. Therefore, image zero is always placed at the origin of the coordinate system of the reconstruction.
2. ESSENTIAL CONCEPTS IN VISUAL GEOMETRY

2.3 Some Computer Vision Concepts

The extension of the pinhole camera model with the calibration matrix accommodates modern cameras where the imaging sensor captures arrays of pixels. Those pixels are the ones referenced in the previous sections as image points. We need to define a procedure to establish sets of corresponding points or pixels that when backtracked intersect the real object in the same spot. In this section we describe how these set of points can be found in every image and then matched across different images.

2.3.1 Image Features

We consider an image feature as a pixel in the image for which we can find some unique description. The image feature is then composed of the pixel location (in $I_{rf}$) and its descriptor. Traditionally (before 1996), these image features were defined by user input. This meant the user had to click on given remarkable pixels in the image such as corners or salient features in the structure.

As early as 1988, Harris et. al. present a corner and edge detector extending the work from Moravec eight years earlier. The Harris corner detector was used in one
of the first automated reconstruction procedures in 1996 which we shall discuss in the next chapter. Even though this detector was popular, we fast forward to 1999 where Lowe proposed Scale Invariant Features (SIFT). Later in 2004 he refined the method [5]. Since then, SIFT features have become one of the most popular choices for computer vision algorithms. They are robust against rotation and scaling. Lowe provided not only an algorithm to find the features but also a rich descriptor. In 2006 Bay et. al. [6] present a more efficient alternative to SIFT called Speeded Up Robust Features (SURF). Even though they report better performance that SIFT, both have been frequently used in computer vision. In 2010 Tola et. al. [7] present a local image descriptor called DAISY. As opposed to SURF or SIFT, DAISY is very efficiently computed densely throughout the whole image. This allows for a much larger set of image features that can be used for reconstruction. Tola’s method produces dense depth maps when using wide baseline stereo pairs of images. Several implementations for extracting SIFT features are available. Lowe provides both a feature detector and descriptor. Vedaldi [8] provides an open implementation of both detector and descriptor.

2.3.2 Image Feature Matching

In the previous section we gave an overview of the most widely used methods for finding points of interest in the images. Those points need to be matched with points found in different images so that the reconstruction process can be performed. Traditional work using user selected image features also required the user to provide the correspondence with other images. That is a laborious job that can hardly be done for more than a couple of images. In 1996 Deriche et. al. [9] use the Harris corner detector to propose a feature matching method that was later used to compute the epipolar geometry. Their matching method consist of exploring the second image using a window of a certain size centered on the location of the image feature to be matched with. This limits their approach to small camera motion where the features are not expected to move much from their original location. SIFT, SURF and DAISY all provide the means to perform feature matching with no limitation with respect to where the target image feature might lie.
In our work we use the SIFT detector and descriptor given its ability to find large number of features in the structure and texture present in facades. For the matching procedure we use the algorithm proposed by Lowe \[5\] based on finding the nearest neighbor in the descriptor space and using a threshold to determine how close the matches need to be. This matching procedure is used to match image features across consecutive images, though a more advanced approach is used to improve robustness as we will discuss in Chapter 4.

2.4 Summary

In this chapter we have reviewed and discussed the essential concepts from visual geometry that are required to understand the 3D reconstruction process. We have introduced the pinhole camera model that fits most consumer cameras and used this definition to develop the fundamental theoretical groundwork. We have defined the camera calibration $K$ and lens distortion, the projection operation that we introduced in Chapter 1 and the two primordial matrices $E$ and $F$ that define the relation between matching projecting points in two different images. Furthermore we have described how objects, cameras and image features can be referenced and our choices with respect to those coordinate systems. Finally we have introduced the concept of image feature and described some alternatives for finding and matching image points across multiple images. These concepts represent the basics over which in the following chapters we are going to develop our methods and techniques for large scale and semantic 3D reconstruction.
History and State-of-the-art in 3D Reconstruction

In Chapter 2 we have established the basics of visual geometry and the standards we use in the rest of this Thesis; this chapter is focused on the state-of-the-art in 3D reconstruction. Given the extent of the field and the number of subprocesses involved, we focus on urban scene reconstruction and study the details and patterns used in well founded and state-of-the-art methods. We begin our journey in the late 90’s and explore reconstruction methods based on user interaction. That leads towards more automated approaches were information is automatically extracted from images and other sources. Finally we review current methods for city size reconstructions. Details on specific steps of the reconstruction process are discussed when appropriate and otherwise left for Chapter 4 when we discuss in detail each of the reconstruction steps.

3.1 User Guided Reconstruction

In Chapter 2 we have reviewed the mathematical principles of the pinhole camera model and how these are used to describe the imaging process of objects in real life. We have also defined the coordinate systems used to reference cameras, objects and images. These however do not say anything about the reconstruction process. In this section we make a thorough review of the history of 3D reconstruction from the perspective of methods that use mostly user interaction. We begin by exploring the early methods and work our way towards state-of-the-art techniques.
3. HISTORY AND STATE-OF-THE-ART IN 3D RECONSTRUCTION

Traditional fully manual reconstruction is a painstaking job. The work is usually performed by creating basic geometric primitives based on visual inspection of the area to be reconstructed. It requires not only a deep understanding of the reconstruction software but also considerable experience and many human work hours.

Early work can be categorized in either geometry-based approaches or image-based approaches. Geometry driven methods require digitizing the imaged object plans or drawings. These objects typically consist of architectural landmarks. A very user intensive process of placing elements in the scene is then performed to build the complete model to the required level of detail. Image driven approaches on the other hand take advantage of image geometry. They require a lot of images and offer poor reconstruction fidelity and still require user supervision for finding image feature matches across frames (stereo matching).

Debevec et. al. proposed in 1996 a hybrid approach for modeling and rendering architecture from a sparse set of images. Their approach was supervised by a user but combined the advantages of both geometry-based and image-based techniques. In their system the user was responsible for selecting a few images and interactively building a basic geometric model. Based on stereo matching, the user introduced step by step different primitives such as boxes, prisms and surfaces of revolution. During the modeling stage, the user was requested to select edges in the images that match components in the model. Their system relied on edge features rather than point features, given that they are easier to localize and are more robust against occlusion. Novel views of the scene were created using viewpoint dependent texturing and additional details were recovered using automatic stereo correspondence.

Having defined a model and the corresponding edges in the selected images, the system then solved the stereo matching problem by minimizing a function that measured the disparity between the projected edges of the model and edges marked in the images. Texture was finally applied in a view dependent approach where the original image was warped to match the novel view point.

The work of Debevec et. al. was one of the first to offer an easy to use, noninvasive system that required only sparse images of the scene. In summary they created a user guided system that can accurately reconstruct architectural scenes in a fraction of the time of previous attempts by taking advantage of the combination of geometry- and image-based approaches.
Even though their work significantly decreased user interaction with respect to earlier geometry-based approaches, it was still labor intensive and the user interaction stage had more weight than the image-based stage.

In 1999, Cipolla et. al. [11] tipped the balance towards automatic reconstruction. Their approach was similar to Debevec, requiring the user to select a few edges in the images that were either parallel or perpendicular in the world. They took advantage of the strong rigidity constraints of parallelism and orthogonality in both indoor and outdoor architectural scenes.

Having defined edges that fulfill their constraints, the internal calibration parameters were found by exploiting the vanishing points of three mutually orthogonal directions. A projection matrix $P$ was then computed linearly for each viewpoint from three vanishing points and one reference point. These projection matrices were further refined via epipolar constraints and additional image feature correspondences. Finally, the projection matrices were used to obtain more image feature matches and 3D textured triangles to represent the scene.

Their system was an interesting advance from Debevec’s work in the sense that it only required the user to select a few meaningful edges. The remaining computations were done automatically with the mathematical principles we have described in Chapter 2. Their approach worked with as few as two images, and although it is extendable to more images, it becomes more user intensive as more edges need to be selected in all images. Additionally, the system only worked with the strong constraint of having three orthogonal directions visible in the scene, which is not always realistic, especially in urban environments when only one facade and the ground are visible in the image (or buildings are not truly orthogonal).

More recent work by Sinha et. al. [12] in 2008 pushes even further the emphasis towards automated computations. As Cipolla, they present a user guided reconstruction system where the user is required to select 2D outlines of planar sections. One major difference with previous systems is the preprocessing step. They employ a SfM technique similar to Snavely et. al. [13] where image features are matched across a collection of pictures in order to recover camera poses and a sparse 3D point cloud. Additionally, they extract lines that are used to estimate vanishing points. All this information is used during the interactive modeling process where the user sketches polygons in the images that correspond to planar sections in the real world. Using a robust framework, the normal
3. HISTORY AND STATE-OF-THE-ART IN 3D RECONSTRUCTION

Vector and depth are estimated for each plane, building a piecewise-planar model that is finally textured. The final stage allows the user to use brush strokes to remove artifacts caused by structures that were not modeled.

The major advantage of the system compared to previous work is the preprocessing stage in which most of the camera parameters are estimated using image feature correspondences. This allows the user to concentrate on defining the piecewise-planar model while a very large collection of pictures is used for building the model. Additionally, the created model is not based on textured triangles but on large piecewise-planar polygons. The texture of those polygons is computed using an optimization procedure in order to generate a seamless texture map, which is also an extension to the viewpoint dependent texturing approach of Debevec. The authors show some reconstructed models of varying detail, with user times ranging from a few minutes up to two hours where roughly a hundred images are used and a few hundred polygons are produced. Even though this is an impressive modeling system that requires only simple interaction, it would still prove unfeasible for very large image collections or extended areas such as a city.

We thus see an evolution from user guided systems to automated approaches. Information was first used to guide the position of the cameras, then to accurately estimate both the camera calibration and projection matrices and finally to extract large amounts of 3 dimensional points and to compute texture or higher dimensional primitives.

3.2 Automated Reconstruction

One of the first attempts to obtain a fully automated reconstruction was by Hartley [14] in 1992. He proposed a non-iterative algorithm to estimate both the calibration and projection matrices of two cameras given a set of feature correspondences between the two images. His approach represented an important step forward from previous work in that he estimated both the internal parameters (focal length and principal point), previously assumed known, and the relative pose of the two images. His work served as the foundation of modern SfM techniques and 3D reconstruction methods. At this point, all image feature correspondences were matched manually across images.

Pollefeys et. al. [15] proposed in 1996 a system to extract 3D structure from a set of images taken with cameras with different internal parameters. Their approach eliminated the constraint of using a single camera with the same internal calibration for
3.2 Automated Reconstruction

all images. They also automated the image feature matching process using Deriche’s corner matcher [9].

As with the user guided systems, there is also the possibility to use lines as image features rather than only points. Furthermore, given the availability of higher resolution cameras, it is also possible to obtain aerial images of urban areas or architectural scenes that can be used for reconstruction.

Baillard et. al. [16] presented in 1999 an approach based on aerial images where line features were found and matched. These line features were then used to automatically find a piecewise planar reconstruction. Their work innovated by using aerial images and linear features but was still limited to 6 views. They extended further their work in 2000 [17] to incorporate the space-sweep strategy of Collins [18].

In 2000 Hartley and Zisserman [1] summarized all the basic ground work in visual geometry in what was to become one of the most widely used textbooks in computer vision. Their work represents a milestone in 3 dimensional scene reconstruction from images.

Werner et. al. [19] extended in 2002 the piecewise planar reconstruction of Baillard. Ground level images were used instead of aerial ones. This produced a higher level of detail at the facade and ground level. After the basic piecewise planar reconstruction was obtained, the facets were used to guide the search for indentations and protrusions in order to model windows and doors. As with previous work, their approach still required three principal directions to be visible. Also, the images needed to contain sufficient information to obtain the vanishing points of these directions. One of the most interesting contributions was the plane sweeping algorithm used to refine the planar facets. The method consisted of sweeping a plane across the scene in some principal direction and computing the correlation of the projected pixels. More detailed geometrical models for rectangular sections (doors and windows) and wedge blocks (dormer windows) were then fitted to candidate areas that did not coincide with the determined planes. As with its predecessors, the method was limited to three images that needed to depict three principal directions and only two types of detailed models were used. Generalization to a larger more generic scene with more images was difficult.

In 2004, Pollefeys [20] presented a complete fully automated system to reconstruct models from images. His system was no longer bound to a small set of images but rather could deal with uncalibrated images recorded with a hand-held camera. Instead of
computing the camera positions based only on image feature correspondences, Pollefeys built a collection of object points that were then matched with image features to obtain a novel camera pose. He employed an optimization method based on the work from Triggs et. al. \[21\] in order to refine the estimated camera positions and the reconstructed model along with an iterative procedure to optimize the object points in 3D space. He also performed a dense reconstruction from adjacent images and fused the resulting models into a single large reconstruction. The resulting model was approximated by a triangular mesh to which texture was applied. Reconstructions of several tens of images were produced depicting both urban and architectural areas. His work was one of the first fully automated reconstruction systems where multiple images were used to accurately reconstruct the scene. An interesting aspect of Pollefeys’s system was the combination of individually researched steps integrated in a single reconstruction pipeline.

Snavely et. al. \[13\] in 2006 employed Pollefeys’ reconstruction pipeline and built a complete system to explore large collections of images. They made use of robust SfM techniques and created a system where the user could navigate image collections by taking advantage of the reconstructed structure. They showed experiments where hundreds of images were used to reconstruct and navigate architectural scenes. Also, new images could be registered and added to the collection on the fly. They took advantage not only of the information obtained from image feature matches but also from the Exif information contained in the image file, which they used in the initialization step. Their work resulted in the toolbox Bundler, freely available online for performing SfM for unordered image collections. Given that the image collections were sparse, a key element in their work was the use of optimization techniques for obtaining consistent reconstructions. They solved and optimized the camera poses using the sparse bundle adjustment by Lourakis et. al. \[22\], which minimized a cost function based on the reprojection error and the reconstructed structure in order to optimize the camera poses and calibration matrices. Even though they used large collection of images, the system was not applied to large scale scenes. Mostly single buildings, architectural landmarks, were used.

\[1\]Exchangeable Image File format is a standard for the specification of files used in digital cameras. It contains information about time and date, camera configuration and internal parameters, location and copyright.
3.3 Large Scale Reconstruction

More recently, in 2007 Mordohai et. al. extended the reconstruction pipeline proposed by Pollefeys by adding GPS and INS data and collecting the image sequences from video. Furthermore, they built a system consisting of up to 8 cameras that they mounted on a moving car to accurately record a large environment. They followed a linear approach in the complete pipeline: calibration, pose estimation, and reconstruction. Given that the camera pose was initialized with the GPS and INS data, they could process many more images than previous approaches while still employing stereo dense matching. In their experiments they reconstructed a \( 80 \times 40 \text{m} \) building with a total of 3,000 image frames from two of the cameras. They also recorded the area using laser and employed the data for evaluation, obtaining a median error of 2.6 cm. The work was used in the DARPA challenge on urban navigation due to speed and accuracy. One of the essential differences from the previous approaches was the use of additional sensors, but also the fact that they provided a dense reconstruction and not only a sparse point cloud. Their results were of a moderate scale and not applied to full city size environments.

Following the complete pipeline approach, Micusik et. al. in 2009 presented a system for urban 3D reconstruction based on multiple cameras with an omnidirectional setup. Instead of using external sensors for accurate camera pose estimation, they relied completely on image feature matches. They obtained a dense reconstruction though they made strong assumptions about the principal directions, allowing a dense piecewise planar reconstruction not bound to specific models. They showed reconstruction results of urban areas of around 400 meters using 200 images.

3.3 Large Scale Reconstruction

We have explored the origins and state-of-the-art in terms of automated reconstruction. We are however interested in city scale reconstructions, for which the work presented in the previous section falls a bit short in terms of size and scalability.

In 2009 Agarwal et. al. presented a reconstruction system for what they called extremely large collections of images gathered from the internet. They employed a distributed system to maximize parallelism in image feature matching and reconstruction. They claimed to be able to reconstruct entire cities within the order or one hundred thousand images in under a day. Their success however came at the cost of using a
few hundred computer cores at a cost of a few thousand dollars per day. One of the interesting aspects of their approach was the image feature matching process. With over one hundred thousand images that were recorded from different points of view, under different conditions, using different cameras and not in a sequence, the simple task of finding image similarities became a challenging problem. They employed a technique inspired from text retrieval where every image was represented as a bag of words. The words were obtained by quantizing the image features. This procedure produced a tree where similar images were connected components. Having obtained the relations between images they estimated the camera position and calibration. The traditional approach used by Snavely and others was to incrementally perform nonlinear Least Squares optimization [21] to minimize the reprojection error. At every iteration, more images were added until all the images were accounted for. This approach was however impractical given the number of images handled by Agarwal. Instead of the exhaustive approach, they find a minimal set of images that compute the essential connectivity of the complete set. For this they use the skeletal sets algorithm from Snavely et. al. [26], and once the minimal set is found an efficient bundle adjustment method is applied.

Agarwal’s approach provided a sparse reconstruction of the main touristic sites of a city from collections of images gathered from the Internet. They employed however a black box approach where the problem of reconstruction was solved by a minimization procedure distributed over a few hundred computer cores at a great expense.

One year later, in 2010, Frahm et. al. [27] presented a paper that claimed to reconstruct Rome in less than a day using a single desktop computer. They scaled up the challenge and used a few millions of images. Their work differed from Agarwal in many ways, not only in the scale of the reconstruction. They retrieved not only a set of sparse colored points but a colored depth map. They captured the image appearance using the GIST feature [28] in order to find a set of iconic views [29]. This was the equivalent of the bag of words approach where the results were a set of images clustered based on appearance. They evaluated the epipolar geometry of the images within each cluster and retained only those clusters with a high consistency, which was an indication that a stable 3D structure could be retrieved. They also took advantage of the fact that over 50% of the images were geolocated. This, together with a vocabulary tree search, allowed them to geolocalize the clusters and to identify neighboring areas. When estimating the camera calibration they employed details of the Exif files of the images or approximated them.
3.3 Large Scale Reconstruction

using a popular view angle for the camera model. Iteratively, new cameras were added and their positions were computed. In order to avoid drift and the accumulation of error, a bundle adjustment approach was used to minimize the reprojection error. Finally, for all the images in the cluster a depth map was obtained for every pair of matching images and then all maps were fused together into a single consistent colored depth map.

Their method was based on efficient parallelization and GPU (Graphics Processing Unit) programming and achieved dense reconstructions of the major touristic sites in Rome. They employed close to 3 million images and obtained similar results as previous methods at a fraction of the cost.

Also in 2010 Strecha et. al. [30] proposed an approach where a complete city was modeled. Their focus was mainly on the registration of large scale subportions of reconstructed scenes. They realized the problem of previous systems where main attractions were reconstructed but their geometrical relations were hardly established, creating a sparse set of large scale reconstructions. Instead, they proposed a method where they assumed that those large scale reconstructions exist and focused on using additional sources of information such as geotags, GPS and GIS models to bring the independent sparse models into a single city size model. Their framework was designed to tolerate nonaccurate data and allowed for incremental growth as new images became available. The advantage of their method was speed, since only simple rigid body transformations between precomputed clusters needed to be computed. Also the method was successful even under weak visual evidence. In their approach, they employed a large number of constraints such as visual correspondences between clusters, GPS data or 2 dimensional maps. For these measurements the accuracy differed, therefore they rejected outliers that did not fit the model. Then they estimated the transformations that minimized the alignment error, reducing the search space of transformations by employing up-vectors, which represented the vertical direction in each cluster. They assumed that the cluster consisted of a 3 dimensional point cloud that was obtained from images taken mostly at the street level. In these point clouds mostly facades were visible. For each point in 3D space, they computed the normal as the smaller eigenvector of the covariance matrix built from the nearest neighboring points. Then they estimated the vertical direction using the Expectation Maximization (EM) algorithm as the one which was orthogonal to most normals. Once all the constraints were defined, they employed nonlinear optimization to estimate the transformation for each cluster. They presented results of a
3. HISTORY AND STATE-OF-THE-ART IN 3D RECONSTRUCTION

large scale map of the city of Lausanne where multiple large scale portions of the city were bundled together to form a single city size 3 dimensional point cloud.

3.4 Discussion

In this chapter we reviewed the developments of 3D reconstruction from the early beginning to state-of-the-art methods: from simple user guided systems to fully automatic city scale reconstructions where millions of images are used.

One of the fundamental issues that we see in the literature is the lack of a unified reconstruction pipeline. Each author tailors her method to fit her needs and then proposes a set of improvements with respect to previous approaches. Even though this works for evaluating the complete method, it makes almost impossible to evaluate the influence of each of the reconstruction steps on the complete picture. The performance of each individual step in the reconstruction is lost in the complete pipeline. This triggers the need to re-think the reconstruction pipeline. If we establish some common grounds where an evaluation is performed on the algorithms employed at every stage, we can better understand the reconstruction problem. We will treat all the steps in detail in Chapter 4 and present an implementation in Chapter 7.

On the other hand, the state-of-the-art work on large scale reconstruction relies heavily on bundle-adjustment-like approaches (see Chapter 4 for more details) where a set of constraints is defined and a nonlinear optimization procedure is used to estimate the corresponding parameters. This is employed mostly for the estimation of the camera positions and calibration parameters. This black box approach has been shown to work well enough but a deeper understanding of the procedure is required in order to speed up the process. The use of these methods also hides the pure motion estimation where even scale drift is hidden in the numerical optimization procedure. We treat in detail the estimation of the camera poses and the scale in Chapter 4 and Chapter 5.

Reconstructions can be obtained in the form of colored point clouds, textured meshes or colored depth maps. Except for some rare occasions where more meaningful data is used such as the gravity vectors, these models do not contain any semantics. This limits the use of such models to visualization purposes only. The models mainly contain geometrical information such as points, planes or directions. However, there is no meaning obtained by grouping those together, such as: these directions belong to the
3.4 Discussion

ground or a building is composed of these planes. We propose in Chapter 6 a set of prototype methods for modeling large scale city reconstructions.
3. HISTORY AND STATE-OF-THE-ART IN 3D RECONSTRUCTION
The 3D Reconstruction Pipeline

As we discussed in Chapter 3, some of the most relevant work on 3D reconstruction, and especially on large scale city reconstructions, relies heavily on solving the calibration, camera positions and 3D structure using a numerical optimization procedure. Some employ hundreds of computer cores [25], others optimize the search space [27] and yet others approach the problem by optimizing the set of geometrical constraints [30]. However, the original reconstruction procedure, where each step was carefully crafted and evaluated, is lost. By re-evaluating and optimizing each step individually, we can gain speed, accuracy and understanding the complete reconstruction process.

In this chapter we use the historical evolution of 3 dimensional modeling to design a homogeneous reconstruction pipeline. We explore the essential algorithms and drawbacks of each of the steps on the pipeline in order to obtain large scale 3D reconstructions from sequences of images.

Our goal is to obtain a near real time reconstruction pipeline with special emphasis on carefully crafting each of the reconstruction steps in order to reduce the need for large numerical optimization. Such a pipeline will increase the flexibility of the reconstruction process, allowing for a better optimization in different fields of application. We focus on urban scene reconstruction, which involves large areas where many images are required.

4.1 Six Step Approach

We discussed in Chapter 3 how the reconstruction processes have moved from user guided systems to visually aided automatic reconstructions. During this evolution,
4. THE 3D RECONSTRUCTION PIPELINE

the pipeline design lost importance compared to large scale numerical optimization
procedures. Yet are certain tasks in the reconstruction process that can be clearly
identified.

- The camera parameters along with the camera positions need to be estimated
  in order to perform the reconstruction. Sometimes, the calibration process is
  considered as the estimation of both, sometimes it is split in two steps: camera
  calibration and pose or motion estimation (see egomotion below). Sometimes the
  estimation is pure image based and sometimes it is based on or aided by external
data from a GPS or INS.

- Once the camera parameters and positions are estimated, an optimization scheme
  is commonly used to correct the parameters and positions given the data. This
procedure improves the consistency of the camera parameters and positions, thus
improving the quality of the reconstructed scene. In most of the recent work
on large scale modeling, this optimization step is the core of the reconstruction
process.

- Having a consistent set of camera parameters and positions, we can obtain a
  sparse, dense, mesh or piecewise planar reconstruction.

- Finally, some semantics is introduced in the model. This may be in the form of
  pre-calculated models for doors, or in the form of properties of the model such as
  the direction of gravity (or vertical direction).

These are the major tasks of the reconstruction process. We take them all into consid-
eration to obtain a 6-step reconstruction pipeline.

We divide the steps in two major blocks. The first one comprises the methods and
techniques required to obtain an accurate and consistent estimation of the camera pa-
rameters, both the internal parameters and the projection parameters (aka camera
poses). The second block comprises the required techniques for obtaining the final 3D
model.
4.1 Six Step Approach

**Figure 4.1:** 3D reconstruction and modeling pipeline. The methods discussed in each of the steps are indicated.

1. **Calibration:** the process of obtaining the internal camera parameters. These are collected in the matrix $K$ and the lens distortion parameters as described in Chapter 2.

2. **Egomotion:** the process of estimating the motion of the camera between frames. These are expressed as one or more projection matrices $P$, where $P = K[R|t]$ as described in Chapter 2.

3. **Refinement:** is the process of optimizing both the calibration parameters $K$ and projection matrices $P$. This optimization is a numerical procedure that aims at minimizing a carefully designed error function.

4. **Reconstruction:** the process of obtaining the 3D representation of the scene as a set of 3D points, aka the point cloud.

5. **Modeling:** the process of converting a sparse 3D point cloud into a set of higher order primitives such as planes, boxes or surfaces of revolution.

6. **Semantics:** the process of extracting information regarding the reconstructed objects or scenes.
In the following sections we explore each of the steps and identify the key algorithms with their benefits and drawbacks. We contribute by adding our insight on the selection of the right algorithm for each of the steps and try to establish its impact in the final reconstruction procedure.

4.2 Step 1: Calibration of Internal Parameters

As we discussed in Chapter 2, the image distortion parameters in $L$ of equation 2.7 need to be determined in order to obtain a linear projection camera. The internal parameters in $K$ of equation 2.5 also need to be estimated before the position parameters of the camera can be obtained. Both image distortion and internal parameters can be estimated beforehand by calibrating the camera, or during the motion estimation process. The first method yields more accurate results since no other parameters are estimated in the process. Additionally, calibration needs to be done only once for every camera, so the overhead of the task is small. This method however assumes that the focal length is fixed for all images (there is no zoom), when the assumption is not true, the second method is employed. For reconstruction pipelines where multiple independent cameras are used, the second approach is commonly used.

The calibration procedure is usually performed in two steps: estimation of distortion and calibration of internal parameters.

These steps usually involve recording images of a certain pattern with known geometry. A typical pattern is a checkerboard with known dimensions. Images of this pattern allow us to estimate the lens distortion using the knowledge about straight lines or right angles on the imaged pattern. By taking pictures of the pattern at different positions and given the knowledge of its geometry, we can also estimate the relative camera position in space and its internal parameters.

One of the most well know calibration strategies was defined by Zhang [31] in 1999. It involves selecting in the images the corners of the black and white squares of a checkerboard pattern and then using an optimization procedure to compute all the parameters. An excellent implementation of this calibration procedure can be found in the Matlab Calibration Toolbox by Bouguet et. al [32]. The toolbox covers all the required calibration parameters and is widely used. In our work we employ the output
4.3 Step 2: Estimating the Camera Motion (Egomotion)

![Figure 4.2: Original (LEFT) and undistorted (RIGHT) images after performing camera calibration.](image)

of that calibration procedure, which also provides the error of the estimation. This error is employed during the estimation of the scale and is discussed in detail in Chapter 5. Using the Calibration Toolbox we obtain the camera calibration matrix $K$ (see equation 2.5) along with the radial distortion parameters (see equation 2.7). This allows us to obtain new undistorted images quickly (see Figure 4.2). Images can be corrected for radial distortion as if they were recorded by a linear projective camera (straight lines in the real world are straight also in the undistorted image).

4.3 Step 2: Estimating the Camera Motion (Egomotion)

Now that we can obtain a sequence of undistorted images of a given scene along with the camera calibration matrix $K$, we focus on estimating the relative displacement between frames. This process is also commonly termed egomotion estimation. We focus on urban models where we employ a sequence of images in which the camera is moving along the scene, therefore we concentrate on motion estimation techniques between two consecutive frames.

There are essentially two different approaches for estimating the relative motion between two images. The first method exploits Image-to-Image similarities, the second method takes advantage of some knowledge of the environment and employs Image-to-World similarities. Most of the state of the art methods, specially the ones we reviewed in Chapter 3, employ Image-to-Image methods to initialize a set of 3D features. Then
4. THE 3D RECONSTRUCTION PIPELINE

Image-to-World methods are used to compare new images with the existing set of 3D features which are updated iteratively.

4.3.1 Image-to-Image Approaches

Image-to-Image approaches take advantage of similarities between two images (see Section 2.3). These similarities are usually defined as corresponding salient points. SIFT \cite{5} is one of the most well known methods for both finding and describing features. However, any alternative feature method can be used, since the methods discussed in this section are based on the fact that feature matches exist and can be defined with image coordinates, regardless of how they are found and matched. Therefore we assume that a set of corresponding points can somehow be estimated between two images.

We discussed in Section 2.1.3 the epipolar geometry and the mathematical relationship between points in two images. We begin with the formal definition to derive an algorithm for estimating the motion between two images.

4.3.1.1 The Essential Algorithm

We know from equation 2.10 that if we have a sufficient number of point correspondences we can compute linearly the fundamental matrix $F$ by setting up a set of linear equations. If we denote the entries of $F$ as $f_{\text{row, column}}$, for every corresponding pair of points $(x, y)$ and $(x', y')$, we can write one equation:

$$x'xf_{11} + x'yf_{12} + x'xf_{13} + y'xf_{21} + y'yf_{22} + y'xf_{23} + xf_{31} + yf_{32} + f_{33} = 0 \quad (4.1)$$

We can also express this as a matrix product:

$$(x',x',x',y',y',x',y',x',y',1)f = 0, \quad (4.2)$$

where $f$ is the vector made up by stacking the entries of the fundamental matrix. For $n$ point matches between two images we can stack one equation per match to form the matrix expression:

$$Af = 0, \quad (4.3)$$

where $A$ is a matrix composed of 8 row vectors as described in (4.2). If 8 point matches are exactly known and if the rank of $A$ is eight, a solution can be found linearly up
to scale. If the matching points are not exact, a Least Squares solution can be found. The Least Squares solution is the singular vector corresponding to the smallest singular value of $A$. If the rank of $A$ is seven (for instance because only 7 point matches are known), a solution can still be found using the constraint $\det(F) = 0$ and solving some polynomial equations (see [1] for full details). Given that it is common to find many point matches, we focus on case of 8 points given that is simpler and less susceptible to errors.

The aforementioned procedure using 8 point matches is the essence of the well known 8-point algorithm, originally due to Longuet-Higgins [33]. There is however something that needs to be considered: the fundamental matrix needs to be singular and of rank 2. In general, this will not hold when solving using the method above.

Enforcing $F$ to have rank two is commonly done through replacing $F$ by a rank 2 matrix $F'$ that minimizes the Frobenius norm. For all the details we refer the reader to the work of Hartley and Zisserman [1].

4.3.1.2 Normalization and Constraints

There is an important aspect that we have not treated so far: normalization. If the algorithm described above is applied over feature points in image coordinates, the solutions are commonly erratic and not robust. Normalization of the input data motivates the most common name for the algorithm, the normalized 8-point algorithm. Normalization is done by a simple translation and scaling of the image feature points before the formulation of the matrix $A$. Hartley et. al. suggest to normalize the points by translating them in such a way that the centroid is at the origin of coordinates and the RMS distance of the points from the origin is $\sqrt{2}$. This transformation is commonly denoted $T$ and it transforms the points $x_i$ in the image as:

$$\hat{x}_i = Tx_i,$$  \hspace{1cm} (4.4)

with $T'$ being the inverse transformation.

The resulting fundamental matrix needs to be de-normalized before motion matrices can be extracted, simply by:

$$F = T'^T \hat{F}'T$$  \hspace{1cm} (4.5)
4. THE 3D RECONSTRUCTION PIPELINE

Extracting the camera positions from $F$ can be done using the method by Horn [2] as discussed in section 2.1.3. Such method produces four candidates for camera matrices. In order to chose the correct camera setup, a voting mechanism is applied where each point votes for the cameras for which the reconstructed 3D point is in front. This linear method however is known to be affected by certain degeneracies, for instance when there is no translation present or when all the image features belong to a planar object. We will consider these in section 4.5 when optimization procedures are discussed.

4.3.1.3 The 5-point algorithm

The 8-point (or 7-point) algorithm is not the only method available to estimate the relative pose between two images. An alternative method was proposed by Nister in 2003 [34] that requires only 5 matching features. The proposed 5-point algorithm is significantly more complex than the classical normalized 8-point or 7-point methods though it was reported to outperform them (see [34]).

When 5 point correspondences are available, five constraints exist in the form of (2.13). Those 5 constraints can be stacked together to form a $5 \times 9$ matrix. Four vectors $\check{x}, \check{y}, \check{y}, \check{w}$ that span the right null space of that matrix are computed either by SVD or QR factorization. The later is recommended by Nister for efficiency reasons. Those vectors corresponds to $3 \times 3$ matrices $x, y, z, w$ and the essential matrix $E$ is known to be:

$$E = E_x x + E_y y + E_z z + E_w w,$$

for scalars $E_x, E_y, E_z, E_w$. These scalars are defined up to a common scale factor (known to be nonzero) and therefore $w = 1$ is assumed. As with the 7 or 8 point algorithms, if more than 5 point correspondences are available, then the four vectors correspond to the four smaller singular values.

The essential matrix is also required to fulfill the constraints:

$$\det(E) = 0$$

and

$$EE^T E - \frac{1}{2} \text{trace}(EE^T) E = 0$$

for scalars $E_x, E_y, E_z, E_w$. These scalars are defined up to a common scale factor (known to be nonzero) and therefore $w = 1$ is assumed. As with the 7 or 8 point algorithms, if more than 5 point correspondences are available, then the four vectors correspond to the four smaller singular values.

The essential matrix is also required to fulfill the constraints:
4.3 Step 2: Estimating the Camera Motion (Egomotion)

Equation (4.6) is then introduced into those constraints. Performing Gauss-Jordan elimination with partial pivoting, a system of equations is obtained where polynomials of degree 2 and 3 in the variable \( z \) are present. Three additional equations are defined and arranged in a \( 3 \times 3 \) matrix \( B \) containing polynomials in \( z \). The determinant of \( B \) must be zero, which leads to a tenth degree polynomial. For this polynomial, the real roots are computed. For each root \( z \), the variables \( x \) and \( y \) are found using the matrix \( B \). Finally, the essential matrix is obtained with equation (4.6).

The implementation of this method is more complex than the 8-point algorithm and the computational expense higher.

4.3.1.4 Outliers

The normalized 8-point algorithm works with as few as 8 points and a Least Squares solution is found. However, as with any Least Squares approach, the presence of outliers is devastating. A common approach for filtering out the outliers is the use of RANSAC (RANdom Sample Consensus) proposed by Fischler et. al. \[35\]. The RANSAC method is an iterative procedure to estimate the parameters of a model from observed data when the data contains outliers. It is guaranteed to converge to an optimal model after an infinite number of iterations. This procedure can be similarly applied to any algorithm for estimating the fundamental matrix \( F \) such as the 5-point and 8-point algorithms.

RANSAC works in a few simple steps. First, the minimum set of 8 feature matching points are randomly chosen. Given those 8 points, the fundamental matrix is calculated. Then the obtained model \( F \) is applied to all feature matches using equation (2.10). Points that are sufficiently close to the model (the reprojection error is below certain threshold) are considered inliers, while those that are too far off are considered outliers. This procedure is run \( N \) times and the best model is the one that contains the most inliers. Once the best model is chosen, the fundamental matrix is calculated again using all inliers provided by that model.

There are many alternative flavors of RANSAC \[36\] that increase the probability of finding an optimal model given a limited number of iterations. A common practice consists of selecting the next minimal set of points at every iteration from the previous best model set of inliers. This reduces the convergence time.
4. THE 3D RECONSTRUCTION PIPELINE

4.3.2 Image-to-World Approaches (PnP)

Image-to-world approaches \[37, 38\] take advantage of the similarities between points in the image space and points in the 3D space. These methods work by finding correspondences between points in a new image and a collection of 3D points that is already known. As we discussed earlier in section 4.3, these methods require an initialization step. This initialization step is performed with an Image-to-Image approach. Points in the two initial images are matched, the relative pose of the images is estimated with any of the methods proposed above and finally the position in space of the matching points is computed. These points are then stored in a database that contains the position in space of the points and the feature description used for the matching process. When a new image is introduced, the matching procedure will take place with the 3D points contained in the database, instead of the feature points found in the previous image. In order to increase the collection of points, once the relative pose of the image is known, new feature matches can be found with previous images. This is an essential step as otherwise when a new image is taken far away from the first pair, the matching cannot be done.

4.3.2.1 The Essential Algorithm

Similar to the Image-to-Image approaches, the essential algorithm is a linear method that takes advantage of the projective relation between the matched points in 3D space and the points in the image space. This linear technique requires 6 matching points, hence the name P6P and it is based on the Direct Linear Transform (DLT) algorithm. The P6P DLT \[39\] approach uses the projective relation between 3D points and the 2D image features. Let us denote the homogeneous vectors of points in 3D space by $X_\alpha$ and the points in the image by $x_\alpha$. Given a projection matrix $P_{(i-1,i)}$, they must comply with the relation:

$$x_\alpha = P_{(i-1,i)}X_\alpha$$

where both $x_\alpha$ and $X_\alpha$ are expressed as homogeneous coordinates as:

$$x_\alpha = [s_\alpha u_\alpha, s_\alpha v_\alpha, s_\alpha]^T$$

$$X_\alpha = [U_\alpha, V_\alpha, W_\alpha, 1]^T$$
4.3 Step 2: Estimating the Camera Motion (Egomotion)

We incorporate the scale $s_\alpha$ in the coordinates of the image feature. Solving for $s_\alpha$ and substituting the system expressed by equation (4.9) we obtain two linear equations for each 2D-3D correspondence (see Chapter 2, section 2.1.3). Since $P$ is a homogeneous $3 \times 4$ matrix, it has 11 unknowns, therefore six correspondences are needed to solve the system, which can be expressed as $Aq = 0$ where $q$ is the vector of entries of the projection matrix $P$ that we want to estimate. Applying Singular Value Decomposition (SVD) over the matrix $A$ yields the Least Squares solution to the 2D-3D correspondences. This results in a projection matrix $P_{(i-1,i)}$ that integrates the rotation, translation and calibration. The translation is obtained as the right null space of $P_{(i-1,i)}$ and the rotation and calibration matrix are found using QR decomposition.

There are a number of alternative solutions [40] [37] [41], both linear and nonlinear, that work with as few as 4 feature matches. For a complete survey of linear methods we refer the reader to work of Ameller et. al. [37]. The most renowned method was proposed by Moreno-Noguer et. al. [38] in 2007. They propose a noniterative algorithm that can solve the camera pose problem with 4 point correspondences or more and a complexity of $O(n)$. Their method, named EPnP, outperforms all previous approaches and is commonly used due to its accuracy and low complexity.

4.3.2.2 Discussion

For urban scene reconstruction where a camera moves along a street, the 8-point algorithm is the best choice over the 5-point algorithm or Image-to-World approaches. As compared to the 5-point algorithm, it offers lower implementation complexity. Also, as shown by extensive experiments by Nister [34], the 8-point algorithm outperforms the 5-point algorithm when the camera moves with a mostly forward motion. Nister provides a deep comparison with the linear approaches (6, 7 and 8 points). The 5-point algorithm performs better in the sense that is not affected by the planarity of a scene, while the linear methods then fail to obtain a solution. With respect to accuracy, the 5-point algorithm performs better in sideways motion when comparing the translational error, achieving 33% of the error of the 8-point method. However, with respect to rotational error or translational error in forward motion, the 8-point algorithm performs better. Nister shows that the break-even point is close to 45 degrees in motion direction with respect to the forward direction. This is precisely our urban setting where the camera moves forward along a street pointing towards the buildings in order to record
the facades. Lack of feature images is not an issue in our setting due to the rich urban textures. Scenarios where all points belong to a single planar structure are unlikely and easy to avoid, though a simple optimization procedure can provide the solution even in that case as discussed further on in Section 4.5. Both the 8-point and 5-point algorithms are common choices in state of the art reconstruction methods when initializing the set of 3D features for computing camera poses.

PnP alternatives are commonly used for computing the camera motion once a set of 3D features has been initialized. There are however a few drawbacks to these methods. When using sequences of images it is very likely that consecutive frames share most of the similarities (except when closing a visual loop). This translates in a higher number of feature matches across consecutive frames than between a collection of 3D points and image features. A larger number of feature matches usually translates in higher accuracy of the algorithms, therefore the preference for Image-to-Image methods. On the other hand computing the motion of the camera based on 3D-2D correspondences implies propagating the error through more steps incurring in a higher accumulation rate. Computing the motion using the 8-point algorithm involves propagating the error in the image features to errors in the computation of the fundamental matrix and then the projection matrix. On the other hand, computing the motion using the PnP algorithms involves propagating the error from the image features, to the fundamental matrix, the projection matrix, the reconstructed 3D point, and finally to the new projection matrix. A complete discussion on the propagation of errors is provided in Chapter 5.

Image-to-world methods are commonly used in 3D reconstruction because they enforce a certain consistency in the camera motion and the reconstructed structure, since new camera positions are computed based on an already existing 3D structure. However, the performance is generally poor and therefore requires a cumbersome optimization procedure. On the other hand, Image-to-Image approaches take advantage of a higher number of feature matches which also provide a higher accuracy on the estimation of the motion of consecutive frames. They however do not provide structure consistency which makes the process less robust.

However, choosing Image-to-Image approaches allows us to take advantage of the higher number of feature matches between consecutive frames and to optimize each of the steps
4.4 3D Reconstruction of Image Features

individually. Also there are some essential benefits with respect to the propagation of
the error in the estimation of the motion.

Table 4.3.2.2 summarizes the pros and cons of Image-to-Image and Image-to-World
approaches.

<table>
<thead>
<tr>
<th></th>
<th>Image-to-Image</th>
<th>Image-to-World</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency in camera and structure</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Scale issue</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Requires optimization</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>More feature matches</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>High accuracy in consecutive frames</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Less error propagation</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

4.3.3 The Scale Challenge

Estimating the relative pose between two images using Image-to-Image methods is not
enough. When a new image is introduced, and its relative pose estimated with respect
to a previous image, the new estimation might be in a completely different scale. For
Image-to-World methods, the scale is implicitly estimated.

As we discussed in the previous sections, when estimating the relative rotation and
translation between two frames, the translation vector is scale free. This means that
the translation can be scaled arbitrarily with no difference to the epipolar geometry.
Even though PnP approaches implicitly solve this problem, we argue that explicitly
estimating the scale improves the accuracy of the estimation. Detailed arguments and
methods are provided in Chapter 5.

4.4 3D Reconstruction of Image Features

We have described methods to estimate the motion of a camera using the 8-point al-
gorithm. This pose estimation process yields the position of the different cameras in
space. We can also obtain matching feature points across frames and then estimate the
3D position of those points.

We need at least two camera poses in order to estimate the spatial position of matching
feature points. The estimation can be done with more cameras but we focus on the case
of two cameras only since the number of constraints is smaller and it better resembles
the scenario of a camera moving through an urban environment. For more cameras the
solution provided below can be extended to obtain a Least Squares estimation.
4. THE 3D RECONSTRUCTION PIPELINE

4.4.1 Linear Method

Following the intuition discussed in Chapter 1, the idea behind reconstruction is to extend a ray between the camera center and a given point in the image for which a match can be found in a second image. This creates two rays per matching point, one per camera. If the camera poses were perfectly calculated, these two rays would intersect in space in the exact location of the real 3D position of the imaged matching point (see Figure 4.3).

However, this perfect estimation will never occur because the image points contain errors that are propagated to the camera pose estimation. In reality, the rays will not intersect in space and we face the challenge of estimating the 3D position of the matching image points. A common approach is to assume that the point should lie on the shortest connecting line between the two rays (green line in Figure 4.3).

The process of estimating the 3D position of the image matching feature is called triangulation. A linear solution can be derived in the same way as used in the 8-point algorithm.

Given two cameras $P$ and $P'$, with the corresponding matching image points $x$ and $x'$, the projective relation shown in equation (4.9) can be transformed in the form $Ap = 0$ in the same fashion as the P6P algorithm. The projective relation produces
two linearly independent equations per image point, which totals to four equations that compose the $A\mathbf{p} = 0$ problem. This is a redundant set of equations since there are only 3 homogeneous unknowns. The solution can only be determined up to scale by finding the unit singular vector that corresponds with the smallest singular value of the matrix $A$. This linear method is equivalent to setting the location of the 3D point in the shortest line segment that connects the two rays, at a location that when projected into the image, is at the center of the projected segment.

There are other more accurate methods for estimating the 3D position of the matching points. Some methods aim at minimizing some error function, for instance the reprojection or geometric error, defined as the sum of squared distances between the reprojected 3D point and the original image features. If this error is minimized an optimal linear solution is provided by Hartley and Zisserman [1]. This procedure however requires an iterative minimization scheme.

4.4.1.1 Dense Reconstruction

The method described above provides a solution to triangulate all found matching points between images. This typically produces a sparse set of points in space. For urban scene reconstructions this translates in a few hundreds points for every pair of images. There are a few common methods to increase the density of the point cloud, these methods are usually referred to as multiview dense stereo. Strecha et. al. [42] presented in 2003 a Partial Differential Equation (PDE) based method for dense depth estimation from multiple images. Some more recent methods by Micusik et. al. [24] and [43] were specifically tailored to urban areas and produced high density reconstructions when surfaces were mostly planar. Strecha et. al. discussed in 2008 [44] a detailed review of state of the art methods and provided a set of standard datasets for evaluation. More recent approaches from Furukawa et. al. in 2010 [45] proposed dense stereo reconstructions based on single pixel matching using color descriptors. Furukawa et. al. also provided an open source tool that could be easily integrated in a reconstruction pipeline.

4.4.1.2 Discussion

Our goal is to obtain a meaningful model of an urban scene, not simply a visually appealing one. For this reason we do not focus on dense reconstruction schemes and
4. THE 3D RECONSTRUCTION PIPELINE

leave those as an alternative when and if more visually dense point clouds are required for visualization. In the rest of our work we focus on reconstructing sparse sets of points in 3D space that are later processed to obtain a semantic reconstruction. We assume those points to be sufficient to obtain the semantic model we wish to obtain. This sparse set of points corresponds with the frame-to-frame image feature correspondences that can be obtained with SIFT or one of its sparse homologues.

Regarding the triangulation method, we choose to use the linear triangulation method. It is faster and requires no iterative refinement. Since an optimization procedure is yet to be applied to refine the camera poses that minimize the geometric error, it would be resources wasted to apply this minimization now for only the triangulated point. In the following section we discuss these refinement methods and how they are applied to optimize the camera motion, the camera calibration and implicitly the geometric error.

4.5 Optimization

Up to this point we have described methods and techniques to go from a set of ordered images to a set of scale consistent pose positions for those images. However, depending on the image quality, scene composition and camera motion, the estimation will vary in quality. For this reason, it is common to apply an optimization step once an initial estimation has been made for the camera positions and calibration. As we discussed in the previous section, the goal of this optimization procedure is to obtain a set of more accurate camera poses that can be used to obtain a more consistent reconstruction. This process is usually known as bundle adjustment and a review can be found in [21] written by Triggs et. al. in 2000.

The optimization can be performed at two different scales. During local optimization a limited number of camera poses are refined. The goal is to refine the cameras and their calibration parameters to reduce drift over time. During global optimization the complete set of cameras is refined. The goal is then to obtain a set of cameras that produce a consistent reconstruction. This step is commonly performed to reduce or eliminate drift over large areas.

There are a large number of flavors in each of the optimization procedures so we will focus on the most relevant and widely used that focus on minimizing the reprojection (aka geometric error). Alternative methods also employ an error function where the
4.5 Optimization

The consistency of the reconstruction is minimized or even the position of the image features are optimized. Typically, the more parameters to be optimized, the higher the computational cost.

4.5.1 Local Optimization

The computational cost of local optimization is constrained by the limited number of camera poses that need to be refined. We start by observing that the position of the first camera of the set cannot be optimized since it is not estimated. It represents the center of our reconstruction universe and there is no point in moving it as long as the rest of the cameras are adjusted accordingly. For every camera in the sequence, we optimize its position and calibration with respect to the previous camera. Each camera pose and calibration are carefully refined, then we adjust every new camera to fit the previous camera that has been already refined. This reduces the drift over time with a low computational cost.

A common approach to refine the estimate of the camera motion and calibration is the minimization of the reprojection or geometric error [1]. This error is defined as the sum of squared distances of the $N$ reprojected 3D points and the original image point. The local optimization is performed as a nonlinear refinement in which the estimation of the projection matrix $P$ is refined so that the reprojection error is minimized. Once the 3D points are obtained by linear triangulation, the image projection is found with equation 2.9.

A common approach is to use the Levenberg-Marquardt algorithm [22] for the iterative optimization. In order to refine the parameters of the rotation we transform $R$ to its quaternion representation, resulting in 3 parameters. The translation accounts for another 3 parameters while the calibration matrix $K$ (of equation 2.5) accounts for another 4 parameters. In total 10 parameters are optimized for every new image. The optimization is performed assuming that the previous image has already been optimized. Using this approach, once one camera is optimized with respect to its predecessors, its parameters are fixed and considered correct.

4.5.2 Global Optimization

As opposed to the constrained local optimization where only a handful of parameters are refined, global optimization aims at refining all motion parameters at once. This
4. THE 3D RECONSTRUCTION PIPELINE

The procedure is more computationally involved given the larger number of parameters and constraints.

There are many alternatives when designing a cost function to minimize in the optimization procedure, for a survey we refer the reader to Pollefeys et al. [21]. There are two methods that are commonly used. On the one hand, motion parameters can be refined. This implies optimizing camera and calibration parameters for every camera in the set, resulting in $11 \times n$ for $n$ cameras, similarly to what we propose for local optimization but for a larger set of cameras. On the other hand, the structure can also be refined given the feature matches across multiple frames. Some approaches even refine the location of the image feature.

A well known approach and accompanying tool to perform global optimization is the Sparse Bundle Adjustment by Lourakis et al. [22]. It employs the nonlinear Levenberg-Marquardt algorithm, as we do during local optimization, in order to refine the complete set of motion and structure parameters.

The work of Lourakis et al. takes advantage of the fact that in general 3D reconstruction problems, the camera is moving in the environment picturing the scene and objects from different points of view. These camera motions typically reduce the overlap between images in a way that far away camera images, in either time or space, have very little in common with respect to the depicted scene. This lack of overlapping results in a lack of interaction among certain groups of parameters with in turn produces a sparse Jacobian. This sparse characteristic is then employed to obtain a highly efficient implementation of the optimization procedure which can be applied to large sets of images and parameters. This procedure is the basis of Bundler, Photosynth and most of the state of the art methods discussed in Chapter 2.

In order for a Bundle Adjustment optimization scheme to work over a large set of images, there need to be constraints between cameras in the form of image feature matches. The larger the number of constraints, the more accurate the result of the optimization can be. In an urban environment, in order to reconstruct as much as possible of the scene, the camera will move in the direction of the street. This means that there will exist fewer constraints across images taken at positions that are far apart. The lower number of constraints results in a decrease in the effectiveness of a global optimization scheme where no information is available regarding the relation between far away camera poses.
We propose an alternative approach that aims at improving the consistency of the reconstruction with a lower computational burden. We employ a technique inspired on the Iterative Closest Point (ICP) algorithm presented by Besl et. al. in 1992 [46]. We have discussed how we employ local refinement to minimize the geometric error over a sequence of frames. Minimizing this error on a global scale will improve the consistency but it will in part destroy our effort to maintain local consistency. Instead, we propose to optimize the camera poses and calibrations by minimizing the sum of squared distances between corresponding 3D points.

For every pair of images we obtain feature matches and a sparse 3D reconstruction of those points. Since we are using sequences of images, it is also expected to be able to match features across 3 frames, though the number of common features rapidly decreases as the images are taken further apart. Matching features across three frames provides image features that are visible in all three images but also provides the possibility to reconstruct the same points using two different pairs of images. If the cameras were estimated perfectly, those 3D points will exactly coincide in space. The accumulated error from the features, the pose estimation and the triangulation guarantees that those corresponding points will not coincide in space. We then design a cost function based on the distance between those corresponding points in space for successive cameras.

$$C(X_{(a,b)}, X'_{(b,c)}) = \sum_{i=1}^{N} d(X_{(i,i+1)}, X'_{(i+1,i+2)})^2,$$

(4.12)

where $X_{(a,b)}$ is the reconstructed 3D point from the pair of images $a$ and $b$ and $X'_{(b,c)}$ is the reconstructed 3D point from the pair of images $b$ and $c$.

This procedure can also be employed locally using 3 cameras at a time, fixing the first two and taking advantage of the chain effect discussed in local optimization.

### 4.5.3 Discussion

In this section we have discussed two areas were numerical optimization is usually employed.

At the local scale we employed a chain approach that minimized the reprojection error. This procedure assumed that the initial camera is located at the center of the reconstruction and therefore cannot be optimized. The consecutive images were refined always under the assumption that the previous image had been already optimized. This
chain effect produces a fast optimization that reduces drift over long camera trajectories. However, minimizing the reprojection error is sometimes not sufficient. If most of the points lie in planar surfaces, the effect of moving those planar surfaces further away or closer to the image has little or no effect on the geometric error, rendering the optimization procedure pointless.

Regarding the large scale, we discussed the most wellknown approach of bundle adjustment. We proposed an approach based on the minimization of corresponding points in 3D space. We defined a cost function based on the distance between corresponding points in space. By minimizing this distance and employing feature correspondences across multiple frames, we improved the consistency of the reconstruction and reduced the number of iterations required.

Furthermore, we solved the problem described above when points lie in planar structures. This procedure can also be applied to the local scale. Employing the chain effect, we used three images at a time, fixing the pose and calibration of the first two and optimizing only the last one. Minimizing the distance in 3D space ensured the consistency of the reconstruction. Naturally, a combination of the two methods can be employed.

4.6 Modeling

Many 3D reconstruction approaches stop precisely here. Given a set of images, the calibration, motion estimation, optimization and reconstruction is performed. This yields a set of points in space, to which the color of the corresponding image feature can be applied. This is called the point cloud. Depending on the structure of the images, the scene, texture, techniques applied, this point cloud would range from a sparse set of points to a more dense cloud. If more points are required dense reconstruction of depth maps can be obtained.

This representation, if sufficiently dense, is usually enough for humans to understand. In particular, when explored in a dynamic 3D visualization tool where the structure can be moved around by the user, one can quickly understand the actual structure. Having the original images at hand helps in this visualization process. However, if the set of points is sparse or there are certain areas with no points, due to for instance lack of texture, visualizing and understanding the 3D structure becomes a challenge. In order
4.6 Modeling

to improve the visualization and understanding of the reconstructed scene, we propose a set of modeling techniques that can be applied to the raw point cloud.

4.6.1 Surface / Mesh Fitting

A classical approach to improve the visualization of a point cloud is fitting a mesh over the reconstructed points or depth maps. The process produces a discrete surface consisting of triangles whose vertices are positioned based on the points in the cloud. When this triangular mesh is textured, a richer model can be visualized. Mordohay et. al [23] employ a technique rooted in the work of Pajarola [47] based on a multi resolution quad-tree algorithm that minimizes the number of triangles while maintaining accuracy. The idea is to find triangles that correspond to planar parts of the structure while these triangles should not bridge big depth discontinuities. It is a top-down approach where triangles are divided according to the criteria explained above. While this technique is usually interesting for visualization purposes, it still produces surfaces that are highly irregular. In an urban environment that consists mostly of human made structures such as buildings or roads, the irregularity of the surface is not desired.

4.6.2 Primitive Fitting

An alternative method to fitting a triangular surface consists of fitting more basic and independent primitives to the set of points. It is similar to the former method in that small primitives are used, though these primitives are not connected. We reviewed in section 3.1 a few approaches for fitting planar primitives to a reconstructed model. Some methods even used boxes and surfaces of revolution. These methods are however not suitable for large urban areas due to the large number of points and the computational cost. In medium size environments we can be dealing with a number of 3D points in the order of hundreds of thousands.

We assume that urban environments mostly consist of human made structures that can be generally approximated by planar structures. We are interested in modeling the buildings and roads and not the protrusions such as street lamps, cars or pedestrians. For this we propose a straightforward and efficient algorithm to fit medium size polygonal shapes to the reconstructed point cloud. First, a point in the 3D set if selected at random. Given this point, the $n$ closest points within a maximum distance $d$ are
selected. Using RANSAC a plane is fitted to these points discarding outliers. If there are enough inliers, then a planar structure has been found. Finally, the outline of the polygonal shape is found as the convex hull. The points used for finding the plane are removed from the remaining set to avoid fitting planes in the same area. This results in a set of planar patches. Figure 4.4 shows the patches randomly colored for an urban scene.

Once the polygonal plane has been obtained, the texture is computed as the most frontal view of the plane. This reduces the parallax effect and produces a natural view of the building facades. We sample the polygon with a number of points defined by a density $d$. We color every point with the pixel corresponding to the projection of the point in the image. We employ the image in which the point is in view and where the normal of the planar polygon is closest to the direction spanned between the projected point and the camera center, or what we describe as the most frontal view. Figure 4.5 shows the previously computed planar patches with the most frontal view used for texturing.

4.6.3 Discussion

The most basic form of reconstruction consists only of colored points in space. When reconstructing only the salient points found in the images using a SIFT or a similar method a sparse set of points is produced. In this section we have discussed some of the methods employed to improve the visualization experience on 3D reconstructions. We have proposed a technique that provides the means to fill the spaces where no points are reconstructed. In our work we do not explicitly use meshes, instead we focus on simple primitives to model the flat structure present in human made urban environments. This produces a more economical model where facades are defined by planar patches. This is a more meaningful reconstruction than that of a surface fitted to the model. The modeling procedure is performed using a fast RANSAC based algorithm that can cope with very large models. We propose a texturing algorithm that employs the most frontal view. This method reduces the parallax effect and the bevel effect in recessed windows or doors.
4.7 Semantic Modeling

We have described methods that can successfully reconstruct and model an urban scene given an ordered set of images. Moreover, we have described methods to improve the visualization of the resulting 3D points through the use of higher dimensional textured primitives. However, most of the information contained in the structure and scene is not considered. Height of the buildings, elevation model of the ground, type of roofs or number of levels in the building are some examples of such semantic information. This information is essential if machine understanding is to be obtained and particularly useful when modeling large scale areas, for instance at a city level. In Chapter 6 we attempt to bridge the gap between purely geometric modeling and semantic information contained in the scene.
4. THE 3D RECONSTRUCTION PIPELINE

4.8 Conclusions

In this chapter we have focused in the design of a reconstruction pipeline for urban scenes. We have described the methods used in each of the 6 different steps. Our final choices are indicated in table ?? Our goal was to design a fast reconstruction procedure that does not rely heavily on numerical optimization. We chose to use 8-point Image-to-Image method for estimating the camera positions given its simplicity and performance when the camera motion is mostly forward motion. This estimation is locally refined with a fast optimization procedure. We also propose a global optimization procedure that intrinsically improves the reconstruction consistency, though as we will show in the next chapter, it is hardly needed. Finally, and to improve the visualization of the model, we proposed a fast RANSAC method to fit planes to the point cloud. The resulting patches are then textured using a most frontal view to obtain a natural view of the model.
4.8 Conclusions

Calibration
We employ monocular cameras to record urban scenes. Images are recorded with a fixed set of parameters (no zoom). Calibration is performed beforehand with the Matlab Calibration Toolbox by Bouguet [32].

Egomotion
We employ the 8-point algorithm due to simplicity and performance for urban scenes. RANSAC is used to reject outliers in the set of feature matches. The local scale is treated in detail in Chapter 5.

Refinement
We optimize each camera position and parameters by minimizing the reprojection error. The previous camera is assumed to be already optimized. Globally, we optimize the complete set of cameras by minimizing the distance in 3D space between reconstructed corresponding points.

Reconstruction
We employ a linear triangulation method to obtain a point cloud.

Modeling
We employ a RANSAC based algorithm to fit planar patches to the point cloud. Texture is applied using the most frontal view of the available images.

Semantics
We focus in modeling large urban areas, in particular topological information, the direction of gravity and the modeling of buildings. We discuss semantic modeling in Chapter 6.

| **Calibration** | We employ monocular cameras to record urban scenes. Images are recorded with a fixed set of parameters (no zoom). Calibration is performed beforehand with the Matlab Calibration Toolbox by Bouguet [32]. |
| **Egomotion** | We employ the 8-point algorithm due to simplicity and performance for urban scenes. RANSAC is used to reject outliers in the set of feature matches. The local scale is treated in detail in Chapter 5. |
| **Refinement** | We optimize each camera position and parameters by minimizing the reprojection error. The previous camera is assumed to be already optimized. Globally, we optimize the complete set of cameras by minimizing the distance in 3D space between reconstructed corresponding points. |
| **Reconstruction** | We employ a linear triangulation method to obtain a point cloud. |
| **Modeling** | We employ a RANSAC based algorithm to fit planar patches to the point cloud. Texture is applied using the most frontal view of the available images. |
| **Semantics** | We focus in modeling large urban areas, in particular topological information, the direction of gravity and the modeling of buildings. We discuss semantic modeling in Chapter 6. |

Table 4.1: Reconstruction steps and our choices of methods for each step.
4. THE 3D RECONSTRUCTION PIPELINE
5

Optimal Local Scale in Camera Motion

We discussed in Chapter 4 the problem of estimating the motion of a camera and reconstructing the 3D structure. In this setup a single camera records the environment. This technique is common given the flexibility and wide availability of consumer cameras. Both Image-to-Image and Image-to-World approaches can be used to obtain an estimation of the camera motion. Furthermore, optimization of the estimation is a common practice, though local optimization (where only a handful of parameters are refined) is often desired due to the smaller computational burden.

We also discussed in Chapters 2 and 4 the nature of monocular systems and the scale ambiguity in the estimation of the motion and the reconstructed 3D scene (see Figure 5.1). The global scale, which describes the real size of things, cannot be recovered unless information about the world is introduced. This is achieved, for instance, by gathering information from a GPS about the distance travelled by the camera.

Additionally, if the motion is estimated on a frame-to-frame basis, considering only the image features and therefore using Image-to-Image approaches, there is a scale ambiguity between the estimated translation vectors (see Figure 5.1). This means that the estimation of the position of two cameras do not share a common scale. We call this the local scale problem.

The local scale accounts for the ratio between the scale of different camera translation vectors. When the local scale is estimated for every camera, all translation vectors can be placed in the same scale frame where distances can be measured in generic units.
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

Figure 5.1: The scale problem in monocular vision. Estimating the camera pose $C_1$ is intrinsically scale free. By sliding the camera over an unknown amount along the translation direction $t_{[0,1]}$, both structures $S$ become indistinguishable reconstructions, only the scale is different. If three cameras are considered, the local scale accounts for the ratio between the scales of the two pairs of cameras (there are three pairs of cameras, though only 2 of the scale ratios are linearly independent). The global scale accounts for the scale of real life objects.

When the global scale is introduced, those units can be translated into real life units such as meters, obtaining the real size of the reconstructed scene.

In this chapter we focus on the local scale problem for which we derive an explicit optimal closed form solution. We first develop the theoretical 1-point method hinted by [48]. Additionally we provide a Least Squares estimate of the scale when more correspondences are available. Then we derive the first order error propagation on the computation of the scale and finally we present an optimal closed form solution. Our proposed method has the same computational complexity as the state-of-the-art methods (see Section 5.2), but with considerable improvement in accuracy.

This chapter is organized as follows. In section 5.2 we present the notation and definitions. Section 5.3 presents a Least Squares closed form solution for the computation of the local scale. In Section 5.4 we provide a first order analysis on the propagation of
the error, from the image noise and camera poses to the computation of the scale. In Section 5.5.2 an optimal closed form method is obtained. Experiments and results are presented in Section 5.6. Finally we draw some conclusions and point out future work in Section 5.7.

5.1 Related Work

There are a number of solutions in the literature to resolve the local or global scale implicitly. Scarammuzza et al. [49] use information about the height of the camera to the ground and position with respect to the axis of the vehicle to obtain the global scale. Also, in [50] a direct/average observation of the speed of the vehicle (reading only the speedometer without any additional aid such as GPS, INS, etc) is used to account for the distance travelled by the camera. These techniques, used extensively in the literature, suffer from a number of disadvantages. When introducing the speed of the camera, or the distance travelled, we are forced to use external devices such as a GPS or a speedometer. If the speed is averaged, then the motion of the camera is limited to a steady rate.

An alternative technique, used in [51] [52] [20], is to solve for the camera pose using 2D-3D correspondences instead of image-only 2D-2D correspondences, therefore employing Image-to-World approaches. This is the so called PnP (Pose from n Points) problem, for which a number of solutions exist [37] [53] [54] [40]. A common linear solution, called P6P, was discussed in Chapter 4. P6P is based on the Direct Linear Transform algorithm and works very similarly to the 8-point algorithm. There are however certain disadvantages when using this range of solutions.

When using PnP methods to estimate the camera pose, the scale is implicitly estimated. These methods suffer from error propagation. Due to the fact that the full camera motion is estimated using only 2D-3D correspondences, the error is propagated as follows: first, from the image features in cameras $C_i$ and $C_{i+1}$ to the estimation of camera pose at $C_{i+1}$. Second, the error is also propagated to the triangulation of the 3D points. Finally, the error is propagated to the estimation of the camera pose at $C_{i+2}$. These effects propagate to subsequent camera poses and accumulate over time.

One of the most advanced algorithms for solving the Pose from n Points problem is the EPnP algorithm by Moreno-Noguer et. al. [38]. They use more than 4 2D-3D
correspondences and solve the camera pose non-iteratively. The authors claim that their technique is at least as good as any state-of-the-art method and is faster and more robust. Tardif et. al. [48] propose a 1.5-point algorithm for solving the scale consistent translation direction while fixing the rotation. The authors claim that this is more robust than using a theoretical 1-point algorithm, though no comparison is provided.

5.2 Definitions and Notation

In the development of our work we assume the pinhole camera model, though the techniques we present can be also be adapted to single view omnidirectional cameras. We begin with the definition of the estimated image features, 3D points and camera poses as a disturbance of the true values. These definitions are useful to understand the derivation of the optimal solution for the local scale and the analysis on the error propagation.

5.2.1 Disturbed Image Features

We define the disturbed images features \( x_\alpha = \bar{x}_\alpha + \Delta x_\alpha \) in homogeneous coordinates (denoted by \( H \)) as:

\[
x_{\alpha H} + \Delta x_{\alpha H} = \begin{bmatrix} x_\alpha \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta x_\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} u_\alpha + \Delta u_\alpha \\ v_\alpha + \Delta v_\alpha \\ 1 \end{bmatrix}, \tag{5.1}
\]

where \( x_\alpha \) is the estimated location of the true image feature \( \bar{x}_\alpha \) and \( \Delta x_\alpha \) is the disturbance.

5.2.2 Disturbed Point in Space

Our method for the computation of the scale requires at least 1 2D-3D correspondence. We define a disturbed point in 3D space \( X_\alpha = \bar{X}_\alpha + \Delta X_\alpha \), or in homogeneous coordinates:

\[
X_{\alpha H} + \Delta X_{\alpha H} = \begin{bmatrix} X_\alpha \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta X_\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} U_\alpha + \Delta U_\alpha \\ V_\alpha + \Delta V_\alpha \\ W_\alpha + \Delta W_\alpha \\ 1 \end{bmatrix}, \tag{5.2}
\]
Note that the disturbance represented here will depend on the error in the motion estimation, the feature location and the method for triangulation.

5.2.3 Disturbed Translation

We represent the translation direction by a unit vector $\mathbf{t}$ with the constraint $||\mathbf{t}|| = 1$, or equivalently $\mathbf{t}^T \mathbf{t} = 1$. We impose this constraint on the disturbed translation $\mathbf{t} = \bar{\mathbf{t}} + \Delta \mathbf{t}$:

$$1 = (\bar{\mathbf{t}} + \Delta \mathbf{t})^T (\bar{\mathbf{t}} + \Delta \mathbf{t})$$

Neglecting the second order terms ($\Delta \mathbf{t}^T \Delta \mathbf{t} \approx 0$) we obtain that $\Delta \mathbf{t}^T \bar{\mathbf{t}} = 0$. This indicates that the disturbance $\Delta \mathbf{t}$ is perpendicular to the unit translation direction $\bar{\mathbf{t}}$.

Expressed in coordinates:

$$\bar{\mathbf{t}} = \begin{bmatrix} t_U + \Delta t_U \\ t_V + \Delta t_V \\ t_W + \Delta t_W \end{bmatrix}, \quad (5.4)$$

with $\Delta t_U t_U + \Delta t_V t_V + \Delta t_W t_W = 0$.

5.2.4 Disturbed Rotation

Most algorithms use a rotation representation based on a $3 \times 3$ matrix $\hat{R}$, therefore we represent the error as a relative disturbance over the true rotation $\hat{R}$ [55]:

$$R = \hat{R}(I + \Delta R) \quad (5.5)$$

The true rotation matrix $\hat{R}$ is an orthogonal matrix ($\hat{R}^{-1} = \hat{R}^T$, or equivalently $I = \hat{R}^T \hat{R}$). The disturbed rotation must be also an orthogonal matrix, therefore we obtain, to first order:

$$I = (\hat{R}(I + \Delta R))^T (\hat{R}(I + \Delta R))$$

$$I = \hat{R}^T \hat{R} + \hat{R}^T \Delta R + \Delta R^T \hat{R} + \Delta R^T \hat{R}^T \Delta R$$

$$I \approx I + \Delta R + \Delta R^T$$

$$\Delta R^T = -\Delta R \quad (5.6)$$
Therefore $\Delta R$ is a skew-symmetric matrix of the form:

$$
\Delta R = \begin{bmatrix}
0 & \omega_3 & -\omega_2 \\
-\omega_3 & 0 & \omega_1 \\
\omega_2 & -\omega_1 & 0
\end{bmatrix},
$$

(5.7)

where $\omega$ is an axis-angle vector representation for a small rotation (the disturbance of the rotation). For brevity we define the rotation matrix with row vectors:

$$
R = \begin{bmatrix}
r_1^T \\
r_2^T \\
r_3^T
\end{bmatrix} = \begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
R_{21} & R_{22} & R_{23} \\
R_{31} & R_{32} & R_{33}
\end{bmatrix}
$$

(5.8)

The absolute disturbance $R\Delta R$ is also represented by its row vectors $\Delta r_1^T$, $\Delta r_2^T$ and $\Delta r_3^T$. Given that the true rotation is unknown, the absolute disturbance can be approximated to first order by multiplying the estimated rotation $R$ with the relative disturbance $\Delta R$.

### 5.3 Robust and Scale Consistent Motion Estimation

We propose an algorithm for robust estimation of the motion of a camera through space. We present the algorithm in two parts. The first part deals with robust frame-to-frame motion estimation. This estimation is scale free. The second part deals with the estimation of the local scale, which we solve in closed form.

#### 5.3.1 Robust Frame-to-Frame Motion

The first part of the algorithm deals with robustly estimating the frame-to-frame motion between two cameras. This is a well known problem and there are a large number of solutions. Among the most used is the 5-point algorithm [34], discussed in Section 4.3.1.3, and normalized 8-point algorithm [1], discussed in Section 4.3.1.1. In our work we choose to use the latter given its simplicity, though our contribution on the computation of the scale can be combined with any frame-to-frame motion estimation algorithm. Additionally, we employ a local iterative refinement step in order to improve the estimate without a large computational burden. Image feature points are calculated.
5.3 Robust and Scale Consistent Motion Estimation

and used as input to obtain a projection matrix $P$ consisting of a rotation $R$ and scale free translation vector $t$.

We summarize the steps of the algorithm:


- Match features using nearest neighbors in the SIFT descriptor space (see Chapter 2).

- Reject outliers using RANSAC [35] (see Chapter 3).

- Estimate the essential matrix defining the frame-to-frame motion using the normalized 8-point algorithm [1] (see Chapter 4).

- Obtain a rotation $R$ and unitary translation $t$ using the method by Horn [2] (see Chapter 2).

- Refine the estimate by minimizing the reprojection error using Levenberg-Marquardt. In this setup the internal calibration parameters (focal length, pixel ratio and focal point) are also optimized.

In general no more than 4 iterations of the refinement step are needed for convergence. The optimization of the rotation is performed in the normalized quaternion space. In Chapter 4 we discussed how this optimization step can cope with some of the shortcomings of the 8-point algorithm such as degeneracies. Given a good set of inliers and a slightly larger number of iterations, the procedure can converge to a good solution even when all points lie in a plane. This motion estimation procedure yields a rotation matrix $R$ and a unitary translation direction $t$.

5.3.2 Computation of the Local Scale - Least Squares

Using the above algorithm we can robustly estimate the motion of the camera through space. However, the translation ratios between different image pairs are still unknown. In this section we derive a closed form solution for the computation of the local scale. As we discussed in Chapter 2, the projective relation between the image features and the 3D points in homogeneous coordinates (denoted by $H$) is defined by:
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

\[ m \mathbf{x}_{\alpha H} = P \mathbf{x}_{\alpha H}, \]

(5.9)

where \( P \) is the scale free projection matrix \( (P = [R|t] \) with \( ||t|| = 1 \)) and \( m \) is the scaling necessary to obtain image points. Given that the camera motion, obtained as described in section 5.3.1 is only calculated up to scale, we introduce the scaling factor \( s \):

\[ m \mathbf{x}_{\alpha H} = [R|st]\mathbf{x}_{\alpha H}, \]

(5.10)

or equivalently:

\[
\begin{bmatrix}
mu_{\alpha} \\
mv_{\alpha} \\
m
\end{bmatrix} =
\begin{bmatrix}
R_{11} & R_{12} & R_{13} & stU \\
R_{21} & R_{22} & R_{23} & stV \\
R_{31} & R_{32} & R_{33} & stW
\end{bmatrix}
\begin{bmatrix}
U_{\alpha} \\
V_{\alpha} \\
W_{\alpha} \\
1
\end{bmatrix}
\]

(5.11)

For every 2D-3D feature match \( \alpha \), we obtain three equations:

\[ mu_{\alpha} = r_1^T \mathbf{x}_{\alpha} + stU \]

(5.12)

\[ mv_{\alpha} = r_2^T \mathbf{x}_{\alpha} + stV \]

(5.13)

\[ m = r_3^T \mathbf{x}_{\alpha} + stW. \]

(5.14)

5.3.3 Solving for the Scale \( s \)

Given the set of 3 equations in 5.12, 5.13 and 5.14, there are at least 4 different ways to solve the system for \( s \).

- **Method 1**: solve using equations 5.12 and 5.14. This implies using \( u_{\alpha} \) and \( t_U \).
- **Method 2**: solve using equations 5.13 and 5.14. This implies using \( v_{\alpha} \) and \( t_V \).
- **Method 3**: substitute equation 5.14 into equations 5.12 and 5.13
- **Method 4**: The last possibility is using equations 5.12 and 5.13 combining \( u_{\alpha} \), \( v_{\alpha} \), \( t_U \) and \( t_V \) in a single solution.

It is at first sight difficult to choose one particular method. In an urban environment where the camera moves along a street, most of the motion occurs in the \( x \) direction, while the motion on the \( y \) axis is mostly negligible. This is particularly evident for
vehicle mounted cameras where the camera will tilt with the inclination of the ground and the image motion will always be the same. In this case, we can motivate our choice of Method 1 by suggesting that the motion in the direction $y$ will not contribute to the solution $s$. Rather, it will disturb it, since the motion will be the result of noise or small irregularities on the road. Naturally, if the camera moves mainly in the $y$ direction, the alternative method 2 should be employed. Method 3 is the equivalent of computing methods 1 and 2 and stacking the results to obtain a LS solution. This implies solving $s$ employing DLT, by solving a system of equations of the form $As = b$ using SVD. Method 4 is similar to methods 1 and 2 but using the ratio $\frac{u_\alpha}{v_\alpha}$ (or the inverse).

5.3.3.1 Solving with Method 1

In order to solve for $s$, we substitute equation 5.14 in equation 5.12 to remove the image scale factor $m$. We then obtain a system of equations in the form $As = b$ such that:

$$A = [t_W u_\alpha - t_U] \quad (5.15)$$
$$b = [(r_1^T - r_3^T u_\alpha)X_\alpha], \quad (5.16)$$

We can construct the vectors $A$ and $b$ using one row for every 2D-3D correspondence $\alpha$, and solve for $s$ using SVD ( Singular Value Decomposition), obtaining the solution in the Least Squares sense:

$$s = A^+ b, \quad (5.17)$$

where:

$$A^+ = (A^T A)^{-1} A^T. \quad (5.18)$$

5.3.3.2 Experimental Motivation

Clearly our derivation and subsequent analysis hold by simply substituting $t_U$ by $t_V$ and $r_1^T$ by $r_2^T$ (along with the corresponding errors). Furthermore, our analysis also holds (or can be easily adjusted) for methods 3 and 4. For method 3, the resulting error will be the result of stacking errors computed for method 1 and 2. For method 4, the error on the ratio $\frac{u_\alpha}{v_\alpha}$ needs to be used instead of $\Delta u_\alpha$. 

69
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

We implemented all 4 methods and compared the accuracy in the estimation of the scale. Figure 5.2 shows the results. We created 3 simulated cameras in space, moving in an urban environment along the $V$ direction. Artificial image features were introduced with an increasing error in the position of the pixels. Results show the average error over 2000 runs. Method 2 is the worst given that the motion on the $y$ direction does not contribute to the computation of $s$. Methods 1 and 3 offer similar performance, with a slight preference for method 1. Method 4 is slightly better than the rest. Our decision to use method 1 instead of 4 is motivated by the fact that method 4 can suffer from division by zero and its behavior is more erratic with a lower number of runs.
5.3 Robust and Scale Consistent Motion Estimation

Figure 5.2: Error on the estimation of the scale using the 4 proposed methods (2000 runs). Method 2 is the worst given that the motion on the $y$ direction does not contribute to the computation of $s$. Methods 1 and 3 offer similar performance, with a slight preference for method 1. Method 4 is slightly better than the rest.
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

5.4 Error Propagation in Scale Computation

Now that we have a closed form solution to compute the local scale, we wish to analyze how the error is propagated. We begin with the first order error propagation in the computation of $s$. To obtain a full picture of the propagation, the subsequent error analysis is performed on the computation of $A^+, A$ and $b$ independently. To first order, the error in equation 5.17 is:

$$\Delta s = A^+ \Delta b + \Delta A^+ b.$$  \hfill (5.19)

This is however expressed in terms of $\Delta A^+$ and $\Delta b$. We need to obtain those expressions as a function of the original disturbances in image features, points in space and camera motion.

5.4.1 Error Propagation in the Computation of $A^+$

We obtain an expression for the computation of $A^+$ in equation 5.18. We now analyze how the error in $A^+$ of equation 5.18 is propagated in its computation. We begin by introducing the disturbances:

$$A^+ + \Delta A^+ = (A + \Delta A)^T (A + \Delta A)^{-1} (A + \Delta A) \hfill (5.20)$$

By first order Taylor series, we may derive a first order approximation for the disturbance in the computation of $A^+$:

$$\Delta A^+ = -(A^T A)^{-1} ((A^T \Delta A + \Delta A^T A) (A^T A)^{-1} A^T + \Delta A^T) \hfill (5.21)$$

Proof:

Neglecting some second order terms in equation 5.21 we obtain:

$$A^+ + \Delta A^+ \approx (A^T A + A^T \Delta A + \Delta A^T A)^{-1} (A + \Delta A)^T \hfill (5.22)$$

We define:
5.4 Error Propagation in Scale Computation

\[ Z = (A^T A + A^T \Delta A + \Delta A^T A)^{-1} \]  

(5.23)

Now, we need to transform \( Z \) in the form \( Z = (A^T A)^{-1} + C \) so we can obtain a description of the disturbance \( \Delta A^+ \). To accomplish this we employ a first order approximation using Taylor series. We first define:

\[
\begin{align*}
  a &= A^T A \\
  n &= A^T \Delta A + \Delta A^T A.
\end{align*}
\]

(5.24)  
(5.25)

We use these in equation (5.23) and transform it to obtain:

\[
Z = (a + n)^{-1} = (a(1 + a^{-1} n))^{-1} = (1 + a^{-1} n)\ a^{-1}. 
\]

(5.26)

The term \( (1 + a^{-1} n)^{-1} \) is approximated using a first order Taylor series:

\[
(1 + a^{-1} n)^{-1} a^{-1} \approx (1 - a^{-1} n)a^{-1} = a^{-1} - a^{-1} na^{-1} = a^{-1} + (-a^{-1} na^{-1}).
\]

(5.27)

This provides a first order approximation of the error propagation in the computation of \( A^+ \):

\[
A^+ + \Delta A^+ = ((A^T A)^{-1} - (A^T A)^{-1} n(A^T A)^{-1})(A + \Delta A)^T
\]

\[ \approx (A^T A)^{-1} A^T - (A^T A)^{-1} (n(A^T A)^{-1} A^T + \Delta A^T) \]

\[ = A^+ - (A^T A)^{-1} ((A^T \Delta A + \Delta A^T A)(A^T A)^{-1} A^T + \Delta A^T) \]

(5.28)

Therefore, the disturbance to first order that is propagated through the computation of \( A^+ \) is:

\[
\Delta A^+ = -(A^T A)^{-1} ((A^T \Delta A + \Delta A^T A)(A^T A)^{-1} A^T + \Delta A^T)
\]

(5.29)
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

5.4.2 Error Propagation in computation of $A$

In the previous section we derive an expression for $\Delta A^+$ in terms of $A$ and $\Delta A$, now we obtain a first order expression for $\Delta A$:

$$A + \Delta A = [(t_W + \Delta t_W)(u_\alpha + \Delta u_\alpha) - (t_U + \Delta t_U)]$$ (5.30)

$$\Delta A \approx [t_W \Delta u_\alpha + \Delta t_W u_\alpha - \Delta t_U]$$ (5.31)

With the expression 5.16 for $A$, $\Delta A^+$ of 5.29 is now computable.

5.4.3 Error Propagation in Computation of $b$

Finally, we need to obtain an expression for the error propagation in the computation of $b$. We follow the same procedure as before, introducing the disturbances in equation 5.16 and propagating to first order:

$$b + \Delta b = [(r_1^T + \Delta r_1^T - (r_3^T + \Delta r_3^T)(u_\alpha + \Delta u_\alpha)) (X_\alpha + \Delta X_\alpha)]$$ (5.32)

$$\approx [(r_1^T + \Delta r_1^T - r_3^T u_\alpha - r_3^T \Delta u_\alpha - \Delta r_3^T u_\alpha)(X_\alpha + \Delta X_\alpha)]$$

$$\approx [(r_1^T - r_3^T u_\alpha)X_\alpha + (r_1^T - r_3^T u_\alpha)\Delta X_\alpha + (\Delta r_1^T - r_3^T \Delta u_\alpha - \Delta r_3^T u_\alpha)X_\alpha]$$

$$\Delta b \approx [(r_1^T - r_3^T u_\alpha)\Delta X_\alpha + (\Delta r_1^T - r_3^T \Delta u_\alpha - \Delta r_3^T u_\alpha)X_\alpha]$$ (5.33)

5.4.4 Summary and Conclusions on Error Propagation

Putting all four equations 5.16, 5.18, 5.33 and 5.29 together yields a first order closed form representation of the propagation of the error in the computation of the scale using any number of 2D-3D correspondences. This result can be used, for instance, to estimate the confidence in the computation of the scale. This, in turn, can be introduced in a SLAM or any probabilistic mapping approach where the covariance of the estimate can now be computed without the need of computationally expensive Monte Carlo simulations. Given that we provide an expression for error in the scale, the error in the 3D triangulation and the frame-to-frame motion needs to be provided. This however can be estimated using either Monte Carlo simulations or, given that the motion is scale independent, can be learned based on noise in the image features. Alternatively, the error propagation on the 8-point algorithm [56] can be employed together with our derivation.
5.5 Computing the Local Scale Optimally

So far we obtained a closed form solution for the computation of the local scale with 1 2D-3D correspondence. We also provided a Least Squares solution when more correspondences are available along with a first order error analysis. Given this, we can also obtain an optimal estimation of the scale in closed form by solving a minimization problem.

5.5.1 Considerations

As show in Section 5.3.2 we can obtain for every 2D-3D correspondence an estimation of the local scale. This estimation is affected by noise, for which we also find an expression in equation 5.19 such that:

$$s_\alpha = \bar{s}_\alpha + \Delta s_\alpha.$$  \hspace{1cm} (5.34)

Each $s_\alpha$ has been obtained using a unique 2D-3D correspondence. $\Delta s$ is described in equation 5.19 as a statically independent random variable with mean 0 and covariance $V[s_\alpha] = \Delta s_\alpha \Delta s_\alpha^T$. We now want to find an optimal average of $s$ that satisfies the constraints:

$$\bar{s}_\alpha = s.$$ \hspace{1cm} (5.35)

for $\alpha = 1, \ldots, N$. This means finding the optimal average for the scale.

5.5.2 Maximum Likelihood Estimation

Following the work in [55] we find the optimal parameter $s$.

We begin by assuming one particular value for $s$. Then each $s_\alpha$ is corrected by minimizing the Mahalanobis distance. This means solving the minimization problem:

$$J[s] = \sum_{\alpha=1}^{N} \left( (s_\alpha - s) \cdot V[s_\alpha]^{-1} (s_\alpha - s) \right) \rightarrow \min,$$  \hspace{1cm} (5.36)

subject to the constraint $\hat{s}_\alpha = s$ for $\alpha = 1, \ldots, N$, where $\hat{s}_\alpha$ is the maximum likelihood estimator and $V[s_\alpha]$ is the covariance of the measured scale $s_\alpha$.

The solution to this minimization is given by:
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

\[ \hat{s} = \left( \sum_{\alpha=1}^{N} V[s_{\alpha}]^{-1} \right)^{-1} \sum_{\alpha=1}^{N} V[s_{\alpha}]^{-1} s_{\alpha} + k \]  

(5.37)

And its covariance matrix:

\[ V[\hat{s}] = \left( \sum_{\alpha=1}^{N} V[s_{\alpha}]^{-1} \right)^{-1} \]

(5.38)

5.6 Experiments and Results

5.6.1 Comparison with State of the Art Methods

In this section we compare our approach for estimating the scale with 2 state-of-the-art methods. For this we built a fully simulated scene of which 3 synthetic images are recorded. Thirty 3D points were projected onto the 3 cameras located at different positions. The internal parameters of the cameras were considered perfect. Gaussian noise was introduced only in the image feature location. The motion of the first pair of images was calculated using the 8-point algorithm. The required 3D points for estimating the scale and motion of the third frame were obtained through triangulation of the noisy image features.

We performed 2000 runs in which a different 3D scene was created and the cameras were randomly positioned. For each run, we estimated the motion of camera 3 using EPnP [38], P1.5P [48] and our novel approaches: the Least Squares method P1Pls and the optimal estimation P1Popt. We incremented the noise in the image feature locations from 0 to 0.4 pixels \[1\] of standard deviation. Then we compared the estimated camera poses with the ground truth camera poses. We performed additional experiments with alternative algorithms such as P6P and P4Pf [40] but the resulting accuracy was worse than any of the presented methods and therefore not shown in the plots.

5.6.1.1 EPnP:

Moreno-Noguer et. al. [38] present a non-iterative O(n) algorithm (EPnP) for estimating the scale consistent camera pose. EPnP requires at least 4 3D-2D point correspondences and is reported to be faster and more accurate than any other closed form

\[1\] 0.4 pixels was the standard deviation obtained when calibrating the cameras and performing corner detection over a checkerboard.
5.6 Experiments and Results

Table 5.1: Algorithm comparison

<table>
<thead>
<tr>
<th></th>
<th>P1Pls/P1Popt</th>
<th>P1.5P</th>
<th>EPnP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. 3D-2D correspondences</td>
<td>1</td>
<td>1.5</td>
<td>4</td>
</tr>
<tr>
<td>Max. 3D-2D correspondences</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Estimation</td>
<td>s</td>
<td>s and t</td>
<td>R, s and t</td>
</tr>
<tr>
<td>Estimation of R</td>
<td>8-point</td>
<td>8-point</td>
<td>–</td>
</tr>
<tr>
<td>Estimation of t</td>
<td>8-point</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Complexity</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

approach [54]. Additionally, the authors compare with one of the top iterative methods and even though the results are marginally worse, EPnP is more stable and faster. The authors provide a Matlab implementation of the algorithm that we used for our experiments.

5.6.1.2 P1.5P:

Tardif et. al. [48] propose a linear 1.5-point algorithm to solve for the scale adjusted translation while fixing the rotation. The algorithm is basically a Direct Linear Transform [1] algorithm where the 3 parameters of the translation are estimated in a Least Squares approach. In this respect the algorithm is similar to our Least Squares approach, only we do not use SVD for solving the system and rather derive a closed form solution. The authors however do not consider the error propagation nor any form of optimal computation. In this case the rotation is also computed with the 8-point algorithm.

Figure 5.3 shows the results of accuracy in rotation, scale and translation direction. Our method only deals with the scale, however we consider relevant the comparison of accuracy between methods that use 2D-3D correspondences (PnP) and methods that use only 2D-2D (image only) correspondences. For the latter there is usually a larger number of correspondences given that only 2 images are required.

Regarding the rotation, the plot shows that accuracy of the EPnP is only marginally worse than that of the 8-point algorithm.
Figure 5.3: Error on the estimation of rotation and translation of a third camera pose given 2 previous camera poses and the reconstructed structure using linear triangulation (2000 runs). TOP: error in the rotation calculated as absolute angle of $\Delta R$ in 5.7. The 1-point coincides with 1.5 because the rotation is calculated with the 8-point algorithm. MIDDLE: error in the translation vector calculated as the distance of the difference vector between ground truth and estimation. BOTTOM: error in the angle estimation of the translation direction. NOTE that 1-point (Least Squares) and 1-point (Optimal) are the same for translation and rotation accuracy. Moreover 1-point and P1.5P are the same for rotation accuracy.
With respect to the estimation of the translation, the 8-point algorithm offers a significant advantage over both the EPnP and the P1.5P algorithms. This can be partially explained by the fact that any method that uses 2D-3D correspondences deals with reconstructed 3D features that contain an accumulated error from both the image space and the estimated camera motion. This error is then passed on to the computation of the rotation and translation. For the case of the methods that use only 2D-2D correspondences, the error propagated to the translation originates only in the image space. In this case the advantage of using a 2D-2D method is clear. There are however scenarios in which such a method will fail. In particular the 8-point algorithm is very sensitive to points lying approximately in a plane. In this case another algorithm can be employed such as the 5-point algorithm, although a simple optimization scheme could deal with some of this shortcomings.

The last figure (see figure 5.3 middle) shows the accuracy of all 4 methods in the computation of the local scale. In this case the P1.5P algorithm performs the worst since the scale is computed together with the translation direction as a Least Squares solution and the error is accumulated as described above. The second method in terms of accuracy is the EPnP. It is significantly better than P1.5P, with a 40% increase in accuracy. The third method is our 1 point Least Squares solution (P1Pls). This method performs very similarly to EPnP offering only an improvement of 10%. Our method for optimal estimation of the scale (P1Popt) performs much better than the other methods offering an improvement of 50% over P1Pls.

### 5.6.2 Outdoors Visual Odometry

We also performed experiments on real outdoors data. In these experiments, images were recorded with a handheld digital reflex camera (DSLR). We used a wide angle lens at a resolution of 1728x1152 pixels. Additionally we tested our method with the well known benchmark dataset P11 [44].

**Outdoors Scene 1 - 60 images:** For the first set, 60 images were recorded covering a total distance of 100 meters of an urban area (see Figure 5.4), following a straight trajectory along one street (see Figure 5.6).

**Outdoors Scene 2 - 174 images:** For the second set, a total 174 images were recorded covering a distance of 200+ meters (see Figure 5.7). In this case the camera moved around a block of houses (see Figure 5.8). The same image was used for first and last
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

position to ensure that the true last camera pose was exactly the same as where the first image was recorded.

The trajectories and triangulated 3D points are shown in Figures 5.6 and 5.8. For the first set, the 3D structure was recovered consistently and the local scale of the map was correctly estimated. Both trajectories were recovered accurately without the aid of any global iterative refinement scheme such as Bundle Adjustment. Given this accuracy, a 3D model was also obtained for the first set (see Figure 5.5) using the methods described in Chapter 4. In terms of accuracy, in the second dataset, the distance between the estimated position of the last image and the first image was below 1 meter. This represents an error of less than 0.5%.

Figure 5.4: Sample images of first dataset.

Figure 5.5: Top: Estimated trajectory (red circles) and recovered point cloud for first set. Bottom: reconstructed 3D model. No global Bundle Adjustment was necessary.
5.6 Experiments and Results

**Figure 5.6:** Top view of the reconstructed scene (in black) and camera positions (red circles)

**Figure 5.7:** Sample images of second dataset.
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

Figure 5.8: Estimated trajectory (red circles) and recovered point cloud for second set. The camera started at the right-bottom corner moving towards the right side and ending in the same position. This was ensured using the same image for first and last position. No global Bundle Adjustment was applied.
**5.6 Experiments and Results**

**Benchmark Scene P11 - 11 images**: We also tested our approach on the well known dataset P-11 provided by [44]. In this set, 11 pictures (see Figure 5.9) were taken sequentially around a fountain. The dataset is provided along with ground truth camera positions that were obtained using a laser range finder. The camera positions were estimated accurately and the image key features were reconstructed successfully. The structure of the fountain was recovered as shown in Figure 5.9. The mean error on the computation of the scale for the 9 cameras (the first one is assumed to be at the center of coordinates and the next one set to a distance of one unit) was 0.14% with a standard deviation of $1.55 \times 10^{-06}$.

![Figure 5.9: P11 dataset consisting of 11 images.](image)

![Figure 5.10: 3D reconstruction of P11 benchmark dataset. No bundle adjustment was used and the scale between consecutive images was computed using our optimal algorithm.](image)
5. OPTIMAL LOCAL SCALE IN CAMERA MOTION

5.7 Conclusions

In this chapter we have presented a closed form solution for the computation of the scale ratio between the translation directions of two consecutive image pairs. Our method requires only one 2D-3D correspondence, though a Least Squares solution is presented when more matches are available. Additionally we provided a first order analysis of the propagation of the error, from the noise in the image features to the computation of the scale. Compared to alternative methods our approach takes advantage of the larger number of image feature matches across consecutive frames. The 3-frame error propagation to the computation of the rotation and the translation direction is then avoided, effectively reducing the accumulation of error over time. Furthermore we took advantage of the estimation of the propagated error and computed the scale optimally also in closed form. This yields results that outperform any benchmark algorithm we compared it to.

Combined with the 8-point algorithm and a local iterative refinement step we achieved accurate results with little drift over large trajectories. Our optimal closed-form solution, integrated in the reconstruction pipeline as we described in Chapter 3, allows for accurate reconstruction. We demonstrated this by reconstructing a street in an urban scene without the need for any global optimization.
Semantic Modeling of Tightly Packed Cities

We have discussed in the previous chapters how 3D modeling has advanced to the point of obtaining 3 dimensional point clouds. We can obtain more visually appealing models by fitting textured primitives or by obtaining dense reconstructions. This information, however accurate, still lacks an essential element: semantics.

Semantics allows you to embed meaningful knowledge into a 3D model. In particular, we are interested in modeling urban areas where we want to represent buildings. For this type of representation, we want to obtain four semantic elements: the direction of gravity, the topographical map of the city (the surface over which the city is built), the shape and height of the buildings and the geometry of the rooftops. These four elements will allow us to obtain an economical and accurate representation of the city with a particular emphasis on the buildings.

Semantics is an interesting tool that can be employed for navigation, city surveying or statistical analysis. A large scale semantic 3D model allows a robot to navigate to a specific semantic point, for instance in front of this building, parallel to that street or facing in the uphill direction, as opposed to the more traditional navigate here command. Regarding city surveying, semantics allows for planning the constructions of new buildings based on the topology underlying the city, optimal trajectories for pipes or wires based on the size, height or roof topology of buildings, simulation of disaster scenarios like flooding, analysis of sun shadows for solar energy machinery or wind flow simulations. Additionally, obtaining semantic details opens a door to obtaining statistical data on
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

the number of stories, roof shapes, size of facades, volume of buildings, surfaces, etc.
In this chapter we present novel methods to extract semantics from a city size point cloud: direction of gravity, city surface, building shape and height and roof geometry.
We employ several data sources to obtain geographical and physical information, and to obtain an accurate watertight \[1\] model of the buildings. This information is then used to compute statistics.
The semantic modeling process is approached in four steps where the modeled city is gradually endowed with a higher degree of meaning. Firstly, the direction of gravity, also called up vector or vertical direction, is estimated robustly from the point cloud. Secondly, the topology of the city is estimated. This technique yields a contour map or isosurface of the ground surface of the city. Thirdly, we estimate the height of buildings’ facades. Finally, the rooftops are geometrically modeled. We integrate these four novel methods into the reconstruction pipeline described in Chapter 4.
The research presented in this Chapter was performed when the author was a visiting student at EPFL under the supervision of Christoph Strecha and Pascal Fua. We express our outmost gratitude for their support, their hospitality and fantastic views of the lake and the mountains from Lausanne.

6.1 Related Work

Most of the early work on semantic modeling focuses on extracting a set of high dimensional primitives to model buildings. We consider it semantic modeling because it does not simply fit geometric shapes to a set of points, but rather aims at obtaining meaningful connections between parts of the buildings.
City centers in Europe represent a modeling challenge relative to suburban areas, specially American, where buildings are usually standalone entities. European cities typically consist of buildings that were constructed at different times and of different styles, packed together to form large clusters. These clusters make single building identification and modeling very complex. Buildings are no longer single structures. Differences in height, elevation or roof topology are also present. We called these type of cities tightly packed cities. We aim at modeling buildings in such scenarios. In particular we want to

\[1\] A watertight model consist of a set of polygons that intersect in such a way that they separate completely the inside and the outside of the model. They are essentially closed polyhedra.
model the basic structure of individual buildings from ground to roof. User interaction will be permitted to narrow down the search space in data registration.

Brenner et. al. [57] proposed in 2001 a mixed method that employed information from ground plans and images to obtain rough building descriptions. They approached the problem in two stages. First they divided building plans which consisted of polygonal shapes into smaller primitives. Each primitive was then fitted according to an aerial image or laser data using segmentation over depth maps. Then all primitives were assembled together to form a single building shape. Their approach was straightforward and provided rough building size reconstructions that could be manually textured. Their method produced a 2.5 dimensional reconstruction since all buildings were assumed to be resting on a single plane (they assumed a flat world).

One year later, in 2002 the work of Scholze et. al. [58] focused on modeling polyhedral rooftops. They inferred the roof topology based on 3 dimensional segments obtained from images. Then they obtained a set of planes that supported those segments. Their method produced a set of consistent planes. A complete topology of the roof of a building was then obtained using five semantic labels that defined how the planes were linked together. Their method was limited by the number of labels and could not cope with arbitrarily shaped rooftops.

In 2006 Verma et. al. [59] proposed a method to obtain watertight models for complete buildings from aerial laser data. They employed a decomposition technique where they fitted a set of simpler primitives to construct a complete building that fitted the point cloud as closely as possible. They then connected those simpler primitives using graphs of roof topologies. Additionally, they provided the means to obtain an estimation of the ground surface underlying the buildings. They characterized points that had a neighborhood with small change in elevation as surface points. Having identified and smoothed those points a triangular mesh was fitted to them. Their method coped well with clearly identified buildings where separation into primitives is simple, though it was not clear how the topology could be identified for more complex buildings with no rectangular shapes.

Engels et. al. [60] proposed in 2007 a method for obtaining roof topologies based on laser data. They employed RANSAC to fit planes to the point cloud. Once planes were fitted to the points, plane regions were obtained using morphological filtering. They assumed that the outline of the building was given as a closed polygon that represented the
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

facades of the building. In order to refine the roof topology they employed a refinement
procedure similar to bundle adjustment. Their method relied heavily in manually set
thresholds which made it unreliable for arbitrary datasets.

In 2008 Lafarge et. al. [61] presented an approach to construct a city model using
building blocks. They extracted 2D-supports from the satellite digital elevation models
either manually or interactively. This produced a set of 2 dimensional polygons that
were then used to fit a 3 dimensional building block. The building blocks were stored in
a library that aimed at covering the full set of possibilities. These blocks were combined
to complete the 2D-support using Bayesian decision. They showed results on city scale
reconstructions using different resolution elevation models. For lower quality models,
the 2 dimensional supports were obtained manually, which defined the structure of the
building in terms of block division. Roof structure was not specifically modeled and left
to the selection of blocks from the library.

Elberink et. al. [62] presented in 2009 a graph theory based approach to reconstruct-
ing the geometry of buildings based on incomplete laser data. They also employed 2
dimensional polygons that denoted the outer limits of the buildings. They assumed, as
previous authors did, that roof topologies could be described as a set of faces and the re-
lations between neighboring faces. They defined some typical roof structures and stored
them in a database, then a matching algorithm was applied over the data. They showed
results on both complete and incomplete point clouds obtained with laser data. The
buildings they reconstruct were mostly stand alone buildings with simple roof geometry
and a simple polygonal outline.

More recently, Chen et. al. [63] proposed a method for modeling accurately the point
cloud of a building with a set of primitives: planes, spheres and cylinders. The shapes
were represented in an algebraic form and then a Least Squares method was used to
fit them to the set of points. Their method was noniterative as opposed to previous
work. Geometric constraints about the relation of the shapes were also introduced in
the Least Squares solution to improve the reconstruction accuracy. First they obtained
a normal for each point in the point cloud. Then the set of points was clustered based
on their normal. Each cluster was approximated by its primitive shape selected by the
distribution of normals. Then a Least Squares method was applied to fit the shapes to-
gether. They presented results using a LiDAR generated point cloud and they assumed
a filtering method could be applied to extract the set of points that belonged to one
single building. As with previous methods, the shape based fitting could only cope with simple roof and building structures.

6.1.1 Discussion and Approach

Most of the methods discussed here can only cope with single building reconstructions. They employ data obtained with highly accurate laser scanners and assume that a polygonal outline is available to isolate the set of points that belong to a building. The density and resolution of the data used varies depending on the size of the environment. Some of the methods rely on libraries that contain pre-calculated models while others employ a limited set of shapes. All those primitives are then glued together to model the complete building. Most of the methods can only cope with simple roof topologies and structures. All these methods perform well over the datasets they employ. However, they do not succeed in modeling tightly packed cities such as the ones described in our urban scenarios. In such cities, the separation between buildings is far from clear and the roof topologies cannot be easily classified using the relation between faces. The methods that automatically extract the polygons that define the building outline will also fail in such scenarios due to the complexity of their structure, the differences in height, elevation and roof topology. Furthermore, save for one approach, the direction of gravity is assumed to be known and the ground surface of the city is roughly estimated or considered flat.

Most of the city centers in Europe do not fit the assumptions made in the reviewed work but rather belong to the tightly packed city style, where the modeling is more challenging. We approach the problem by employing three sources of information. We use point clouds obtained using ground and aerial images. These point clouds are less dense than the laser data and less accurate. We also employ 2 dimensional GIS data obtained by direct user input. Finally, we employ satellite images from Google. Employing these data sources we obtain a semantic model of tightly packed city centers. The steps for semantic modeling are shown in figure 6.1. The sources of information used at every step are also indicated.

First, we present a novel method to accurately estimate the direction of gravity, which allows us to obtain a vertical projection of the point clouds. Inspired by Chen et al [63] and Engels et. al [60], we employ a RANSAC based plane fitting method in a few selected points in facades to estimate the direction of gravity. With this, the
three sources of information can be registered. The direction of gravity and the point clouds are then used to obtain an accurate topological map of the underlying ground surface of the city, where in a similar way as Verma et. al [59] we smooth the total surface by assuming a locally flat surface. Having obtained those, the height of building facades is estimated with a robust method based on the deviation of points from the facades provided by the GIS data. The remaining points belonging to the roofs are modeled through a watertight model composed of planar primitives. Instead of the plane fitting algorithm proposed by Engels et. al. [60], we employ image analysis to find the connecting points of the roof planes and a 2D mesh to obtain the connected planes similar to Scholze et. al [58] only without limitations in the connecting possibilities. These four novelties allows us to construct semantic models of tightly packed urban areas without making any assumptions about the direction of gravity, the underlying surface or the primitives that compose each building. Finally the model is textured using the aerial image.

Figure 6.1: Pipelines for Semantic Modeling of tightly packed cities using GIS, point clouds and aerial images.
6.2 Sources of Information

For the task of semantic modeling we employ three different data sources: GIS data, ground based and aerial point clouds, and satellite images. In this section we discuss the extent, size and details of these sources of information.

6.2.1 GIS and the Outline Polygons

One of the fundamental steps in semantic modeling is separating the set of points that belongs to buildings from the rest of the data in the point cloud. This, for certain types of buildings, can be done without assistance from further data by simply establishing certain constraints on the variation of height in the neighborhood of each point as suggested by Verma et. al. [59]. While this approach works well for standalone buildings, it is unrealistic for tightly packed city centers. Some authors [61] approach the problem by analyzing aerial depth maps and establishing boundaries with the surrounding environment with the possibility to refine them manually. This approach is equally unlikely to succeed in tightly packed urban centers.

A common approach is to assume that the outline of the building is provided. This data can be manually introduced by selecting the corners of each building in an aerial image. For large scale areas this is a daunting job. In order to isolate the set of points that most likely belong to a single building we employ GIS information, freely provided by OpenStreetMap. This data contains a set of closed polygons that represent buildings in the area (see Figure 6.2). The corners of the polygons are represented in a 2 dimensional coordinate system.

OpenStreetMap is created by users through a process of selecting corner points in satellite images. This user centered manual procedure sometimes yields outline polygons that are not accurate enough. Especially for single buildings that are composed of multiple structures[1] OpenStreetMap data tends to be a bit simplistic. For these rare occasions, we extend the GIS data by manually introducing additional points (see Figure 6.3). The GIS data provided by OpenStreetMap also contains labels that describe the use of each building. We apply user interaction only on cathedrals and palaces.

[1]For instance, a cathedral is a single building that is composed of several structures of very different heights and with very different roof topologies.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

6.2.2 Large Scale Point Clouds from Ground and Air Data

Using sets of points clouds obtained with lasers or LIDAR is very common. These devices provide large and dense sets of points (density depends on the distance to the objects, though typically a few hundred points per $m^2$ can be obtained), however they are expensive and difficult to operate.

Dense reconstruction approaches such as the ones discussed in Chapter 3 provide an alternative method for obtaining large sets of 3 dimensional points. Strecha. et. al. \[30\] propose a method to register large point clouds reconstructed using images at the ground and air level. For the data recorded at ground level, a camera is mounted in a car equipped with a GPS unit. For the aerial images a small unmanned airplane is used equipped with GPS and INS sensors. The point clouds are produced with a
6.2 Sources of Information

Figure 6.3: Palais de Rumine. BLUE: OpenStreetMap data. RED: Manually enhanced polygons.

reconstruction pipeline as described by [30] [42] and they present results that span the city center of Lausanne. We employ this sets of aerial and ground based points.

The point cloud obtained from aerial images contains 6.4 million points and covers an area of 0.4 km$^2$ (an average density of 40 points per m$^2$). The point cloud obtained from ground images contains 1 million points and covers a slightly smaller area. The points for the ground based point cloud are mostly located in the facades of the buildings where the density varies a lot throughout the cloud.

Figure 6.4: Point cloud obtained from aerial images.
Figure 6.5: Point cloud obtained from ground images.
6.2 Sources of Information

6.2.3 Satellite Images

We employed satellite images obtained from Google Maps for ground and roof texturing. For the area of Lausanne we composed manually an aerial image (see Figure 6.6) of 16 megapixels and a resolution of 25 pixels per $m^2$. This composition was created through a process of tiling high-resolution images of the center of Lausanne. The image shows the complexity of the layout and construction of the city and the intricacy of the relationships between individual buildings.

Figure 6.6: Aerial Image obtained from Google Maps.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

6.3 Estimation of the Direction of Gravity

Estimating the direction of gravity is an essential step in creating a semantic model from a 3D reconstruction. The gravity or up vector provides information regarding the direction in which buildings are constructed and serves as an orthogonal reference for facades. For a city size scale consisting of a few hundred buildings and covering an area of $1 \text{km}^2$, the theoretical direction of gravity varies $0.0025\%$ or $0.009$ degrees, within that area (assuming a perfectly round earth with all the mass located at the center). Estimating the up vector also serves to provide projections onto the ground plane. Because of this small variation, we will consider a single ground plane for the whole city. This projection of the 3D model or point cloud is used to register different datasets such as aerial, ground based or satellite images.

In order to find the direction of gravity, we assume that on average, facades are built orthogonal to the direction of gravity. Using the technique described in Section 4.6.2, we employ RANSAC to find planar patches in the point cloud obtained from ground images. Given that the point of view is at the ground level (the bottom of the facades), it is reasonable to assume that most of the reconstructed points belong to facades (see Figure 6.5). This procedure yields a set of planar polygons and their normals. We then find gravity as the vector $z^*$ that minimizes the sum of the dot products with each facade normal $n_i$:

$$z^* = \arg \max_z \sum_{i=0}^{N} |z \cdot n_i|,$$

with $||z|| = 1$ and $||n_i|| = 1$.

Both $z$ and the normals are normalized to length 1. Possible outliers are removed using RANSAC. In other words we find the direction that is most orthogonal to most facade normals. The Levenberg-Marquadt algorithm is used finally for refinement.

Alternative methods compute the normal of each point in the cloud based on the surrounding points and then use those normals to compute the up vector. Our method employs fewer points and outliers are removed. This provides a lower computational burden.
6.3.1 Results

For the point cloud of Lausanne containing 1 million points, 2000 planar patches are found that contain at least 200 points. Outliers are rejected using RANSAC with a fitting threshold of 0.1 meters. The nonlinear optimization converges after 3 iterations. For evaluation purposes we compare the obtained gravity vector with the up vector provided by the INS sensor of the unmanned airplane used to gather the data, obtaining an error of 0.0031 degrees, only 3 times bigger than the theoretical error based on the size of the area. Figure 6.7 shows the average error over 10 runs in the estimation of gravity as a function of the number of planar patches.

![Graph showing average error over 10 runs in the estimation of gravity as a function of the number of planar patches used.]

6.4 User Driven Dataset Registration

All three sources of information are in different scales and coordinate systems. Before we can model the buildings, we need to register them. We employ the direction of gravity to project the point clouds vertically obtaining 2 dimensional sets of points.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

GIS data and satellite images are 2 dimensional, while the point cloud is 3 dimensional. We tackle the registration problem in two stages. Firstly we use simple user input consisting of corner matches between the 2D GIS data/aerial image and a vertical projection onto a plane of the 3 dimensional point cloud. All 3D points are projected onto a plane orthogonal to the direction of gravity, which we estimate using the method described in Section 6.3 and we call the ground plane. A simple user interface displays both the GIS data with the building outlines, the aerial image and the 2D projection of the point cloud. The user then selects at least 2 pairs of corresponding building corners. Those points are used to compute the rigid 2D transformation and scaling to bring the GIS data/aerial image into the same coordinate system as the point cloud such that:

$$\mathbf{x} = R(s\hat{x} + t),$$

(6.2)

where $R$ is the 2 dimensional rotation, $s$ is the scaling factor, $t$ the translation and $\mathbf{x}$ and $\hat{x}$ the selected matching points in projection and GIS data/aerial image respectively. Rotation, translation and scaling account for a total of 4 parameters.

Only two points are required to obtain the transformation in closed form. For robustness we allow the user to select more points. For every pair of points we obtain two linear equations. Compiling those equations together we obtain the transformation as a Least Squares solution using the DLT algorithm.

6.4.1 Registration Optimization

The second stage of the registration of GIS data aims at refining the Least Squares estimate. We are employing the estimation of the direction of gravity (see Section 6.3), which we assume is as accurate as possible. Therefore we fix the rotation angle around the direction of gravity. This leaves three parameters to optimize: a rotation around the direction of gravity and a 2 dimensional translation in the ground plane. Optimization in the vertical direction is not required since the GIS data is essentially 2 dimensional and we will later estimate the ground surface over which the buildings are constructed. We have two fundamentally different point clouds. The set of points generated from aerial images contains mostly points of the roofs, while the set of points generated from ground images contains mostly points of facades. We know that points that belong to a facade should project vertically near the outline of the building as described in...
the GIS polygons. Points that belong to the roof should project vertically inside a slightly stretched outline polygon to accommodate for roof protrusions. Based on this, we construct an optimization procedure. The goal is to minimize:

- the accumulated distance between the projected facade points \( x_f \) and the closest outline segment \( l_c \)

- the accumulated distance between the vertices \( O \) of the convex hull of the roof points and the closest outline segment.

Therefore, we construct a cost function \( f(x_f, x_r, L) \) such that:

\[
f(x_f, x_r, L) = w \sum_{i=0}^{N} d(x_{fi}, l_c) + (1 - w) \sum_{j=0}^{N} d(O_{x_rj}, l_c)
\]  

(6.3)

where \( x_f \) is the vertical projection of the 3D points from the ground based point cloud, \( x_r \) is the vertical projection of the points from the air based point cloud, \( L \) is the polygon for a given building consisting of segments \( l \), with \( l_c \) being the closest to a given projection and \( O \) is the outline of the vertical projection of the rooftop points, computed as the convex hull. The weight factor \( w \) is chosen as the ratio between the number of vertices of the convex hull and the number of facade points plus vertices.

This method aims at fitting the GIS data to the vertical projection of the city such that the outlines of the buildings fit best with the actual point cloud. It is based on the assumption that the number of points in the cloud that belong to the ground is low with respect to the number of points that belong to facades or rooftops.

The distance between one projected point \( x_{fi} \) and the closest segment \( l_c \) is defined as the shortest distance between a point and line segment. The distance between two polygons is defined as the sum of distances between each of the \( M \) segments of \( O_{x_r} \) and the closest segment in \( L \), defined as \( l_{cO} \). The closest segment is defined as the one with minimal distance between the centroids of the two segments. This measure is motivated by the fact that the buildings are roughly aligned by user input and the points that belong to the roof are expected to fall within the area enclosed by the GIS polygon, therefore most of the segments from the convex hull will be very close to the segments of the GIS polygon (see figure 6.8).
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

Figure 6.8: LEFT: facade points project near the building outline. RIGHT: roof points project inside the building outline.

The error function is represented in Figure 6.9. Whiter areas represent a lower error value. The value for each point (either point of the facade or a vertex of the convex hull) is obtained by projecting the point in the map. The further away from the GIS polygon, the larger the error. The optimization is performed for all buildings at once.

Figure 6.9: Representation of the error function for optimization.
6.5 Ground Surface Reconstruction

Once the direction of gravity is obtained, we estimate the topological map of the city. For this we employ a surface fitting approach that works from the bottom up. All points from the point clouds must lie on top or above the surface of the city. We start the procedure with the ground plane. This plane is sampled using a square grid with a grid constant of 5 meters and located at the height of the lowest point in the cloud. For reference, we call the sampled plane or grid the heightmap since it maps the height of the ground surface of the city model. This map can be displayed as a 2D image similar to a contour map.

Every node of the grid is set to a height equivalent to the lowest 3D point (from the point cloud) contained in a cell of 5 meters along each side centered at the node. This produces an irregular fitting where nodes are initialized only where evidential data is found. Additionally, we build a grid of the same size as the heightmap that we use to store weights that support the evidence for the estimation of height. We call this the weightmap and it is used later on the height propagation procedure. The weight is initialized as the ratio between the number of points contained in the 3D cell and the total number of points in the cloud.

After the height-map has been initialized, we smooth out the sudden changes in height. These sudden changes in height are caused by lack of evidence such as empty areas in the point cloud. We define the maximum declination as the absolute difference between one cell with evidence and the minimum height of the eight surrounding cells. If the maximum declination is above a certain threshold, then the height of the cell is set of the minimum of the neighboring cells. This threshold represents the maximum ground elevation change in the area. Additionally, small isolated structures are removed (see Figure 6.10 - BOTTOM-LEFT) since they most likely represent small reconstructions that do not account for a full building or even a large part of one, for instance when barely any points are reconstructed or when points belong to a tree or a small statue. This is achieved by finding small patches in the weight-map and setting them to zero. The final step consists of smoothing the heightmap based on the weightmap. We apply a convolution filter constructed with the weights of the surrounding cells.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

6.5.1 Results

We apply our proposed method to the point cloud (both air and ground) of the city of Lausanne. The complete set of 7.4 million points is reduced using density driven sampling to obtain 0.5 million points. Figure 6.10 shows the four stages of the estimation process where buildings are gradually smoothed out of the resulting topological map. Figure 6.11 compares the obtained topological map with an isoline map provided by the council of Lausanne. Red lines represent the same height. The hill where the cathedral is located is accurately recovered along with the boundaries with Rue Saint-Martin.
Figure 6.11: Validation of the estimated ground surface.
6.6 Estimation of the Height of Buildings

Once the surface of the city is estimated, we obtain the height of the facade by observing the change in the average distance between the sparse points and the facades. Since the distance from point to facade is the orthogonal segment that spans from the point to the facade and we have estimated the direction of gravity, this is equivalent to computing the distance in the projected space. One building commonly consists of a multiple set of facades so we only consider the distance of each point to the closest facade.

We propose a method that works in three consecutive stages. First we construct a 2D vector \( V = [z_i, d_i] \) containing the height \( z_i \) of each sparse point \( i \) and the distance \( d_i \) to the closest facade normalized by the distance to the centroid of the 2D outline polygon. Over this vector a 2 dimensional histogram is constructed (see Figure 6.12 TOP-LEFT). Then we multiply this distribution by an exponential function (see Figure 6.12 TOP-RIGHT), obtaining a distribution where relevance of points is graded as they are further away from the facades (see Figure 6.12 BOTTOM).

We then sum up over the distance and obtain a 1 dimensional distribution (see Figure 6.13). Finally, we compute the derivative and obtain the roof line as the minimum height at which the derivative is higher than 20% of the maximum value. This value represents the highest point in the facade and the point where the roof begins. We employ the 20% threshold to represent the noise present in the point cloud. The idea is to identify the first instance where points start to deviate from the facade due to architecture rather than noise.

6.6.1 Results

The facade estimation algorithm is applied over all buildings present in the GIS data for which sufficient evidence exists to support the separation between roof and facade. The GIS data of Lausanne contains 937 buildings. For 742 of them sufficient evidence is found to obtain an estimate for the facade height in the point cloud. Three different building models are shown in figure 6.14 along with the corresponding roof points. In all three the height of the facade is accurately estimated allowing for a clear cut separation between roof points and facade points. Since the building facade model is entirely based on the accuracy of the GIS data, for buildings where the point cloud does not correspond with the GIS building outline, the facade cannot be correctly estimated.
Figure 6.12: Three stage procedure for obtaining the height of facades. TOP-LEFT: distribution of point-facade distances over height and distance. TOP-RIGHT: exponential function used to grade points based on their distance to the facades. BOTTOM: resulting graded distribution.

Figure 6.15 shows the estimated height of a building together with an image of the real building.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

Figure 6.13: Summed up distribution of the height of a single building (BLUE) and derivative (RED).

Figure 6.14: Estimation of the height of facades for 3 different buildings.
Figure 6.15: Comparison between estimated facade height and real building.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

6.7 Rooftop Modeling

Having obtained an estimation of the height of the facades, we propose in this section an automatic method for estimating a geometric model of the rooftop. We consider the roof as the outline polygon that coincides with the outline of the building, and a set of height control points that define the 3-dimensional shape. Those control points must lie inside the outline polygon.

Some authors consider the rooftop as a collection of planes and a set of predefined configurations they can be found in. This technique, together with the pre-calculated models approach, cannot be applied to highly irregular building outlines such as the ones found in city centers.

We approach the geometric modeling problem in four stages. First, the control points of the roof are introduced. Second, the height of those points is estimated. Third, the set of planes that form the geometric roof is determined. Finally, a margin approach is employed to reduce the number of faces of the geometric model.

6.7.1 Finding Control Points

The first step consists of introducing control points to be used in the geometric structure. These points relate to corners, ridges and similar key points in the structure of the roof. In order to automatically estimate those points, we employ the 17Mpx aerial image. We compute the corners using the Harris method for the part of the image that lays inside the building outline polygon. The motivation to use a corner detector is the fact that key points in the roof structure are very likely to represent places where two or more planes intersect. This confluence of planes implies a significant difference in the direction of the normal, which in turn will produce a difference in lighting. This difference will be further enhanced by changes in shadows and materials. This method is very likely to recover most of the relevant points in the roof structure, though it will also recover nonrelevant ones such as points where the intensity in aerial image varies due to material changes.

6.7.2 Estimating the Height of Control Points

Once the control points are determined, their height is estimated using the points of the aerial point cloud that lie inside a polygon that slightly bigger than the outline of
the building. The polygon is bigger to accommodate for small errors in the registration. We employ a polygon 30% bigger, the more accurate the registration, the lower the number. The height of each control point is estimated using a similar technique as the one employed in the estimation of the ground surface. For every control point, a height is set equal to the lowest height of the 3D points contained just above the point in the direction of gravity. If no points are present, the height is set to the height of the building, placing the control point in the roof plane.

6.7.3 Finding Planes

We want to obtain a geometrical and watertight 3D model of the rooftop, therefore we now need to calculate the planes that intersect in the estimated control points.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

For this we employ the Delaunay triangulation method on the vertical projection of the points. This yields a set of triangles that cover the complete rooftop making a watertight geometric model. The vertices of the triangles are the estimated control points and the vertices of the outline polygon of the building.

6.8 Modeling the city of Lausanne, Results and Discussion

In this Chapter we have proposed methods for estimating certain semantic aspects of the reconstruction of a city. For our experiments we obtained two 3D point clouds of the city of Lausanne from Strecha et. al. [30]. The aerial point cloud consisted of 6.3 million points reconstructed from aerial images recorded with an UAV (Unmanned Aerial Vehicle). GPS and INS sensors were also used to obtain the reconstructed points. In this case the recorded INS information provided an estimation for the direction of gravity. The ground based point cloud consisted of 1 million points reconstructed using images recorded with a ground vehicle.

GIS data was also obtained from the Open Street Map project. The outline polygons for 937 buildings were obtained. Using Google Maps, a 17 Mpx image was composed of the area covered by the point clouds.

Figure 6.17 shows a top view of the three sources of information registered. Figure 6.18 shows the aerial image (see Figure 6.6) used for texturing the ground plane. The registered buildings obtained from GIS are also displayed with a fixed height. The same procedure can be applied over the computed ground surface and it is shown in Figure 6.19.

Once the point cloud has been registered and the ground surface has been estimated, the height of each building can also be estimated. Two views of the entire city center of Lausanne are shown in Figure 6.20.

6.8.1 Statistical Modeling of Building Properties

Obtaining a 3 dimensional model of buildings by estimating the height of their facades is useful for visualization. Within those models an additional layer of information is inferred, namely the height of the buildings, the area they occupy and their volume. We employ these properties to obtain 3D visualizations of the buildings. Details of the
6.9 Conclusions

Figures 6.17, 6.21, 6.22, and 6.23 show the three building properties mentioned above. In these models, identifying key buildings is easy and has many potential uses in city surveying or planning.

6.9 Conclusions

In this Chapter we have developed the methods and techniques to obtain a semantic model of a city based on three sources of information: point clouds, GIS data and aerial images. The goal was to obtain a model with valuable information for city surveying tasks.

In a stepwise approach, we have increased the level of complexity of the simple point clouds. First, we showed a robust and accurate method to estimate the direction of gravity. This is a key element in order to put the model in a realistic reference system. Based on this direction, an estimation of the topological map of the city was obtained. For this we developed a multistep grid based approach that created a smooth surface over which buildings could be placed. Using freely available GIS data, we developed
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES

Figure 6.18: Top view of the textured ground plane and the GIS outlines of buildings.

a method to estimate the height of the facades of buildings. This, together with the estimated city surface, allowed us to place the GIS data in a truly 3 dimensional context, placing watertight buildings in a realistic environment. This produced a set of flat roof models for the buildings. We improved those by creating a geometric model of the rooftops. For this, a set of control points was calculated based on the aerial image of the building. Then the height of those control points was estimated according to the point cloud, obtaining a watertight geometric model of the rooftops.

Finally, a texture can be applied to both the surface and the roof models, obtaining therefore a rich and visually appealing representation of the city that can be seen in figure 6.24.

The information obtained during the modeling process can also be used for semantic visualization. We have shown visualizations of the height, area and volume of the buildings in a complete city.

There are two challenging aspects of the modeling process. the geometric characterization of the rooftops and the evaluations of the methods. The geometric characterization is difficult due to the limited accuracy of the aerial reconstruction and the complexity of the roof structure in dense city centers such as Lausanne. More accurate aerial images and alternative control point detectors might be used to obtain better geometric descriptions. The second challenging aspect is the evaluation of the methods. Obtaining
Figure 6.19: Google image used for texturing the estimated ground surface.

ground truth for the semantic elements we obtain is difficult and costly.
Figure 6.20: Views of the model of the city center of Lausanne where the height of the buildings has been estimated.
6.9 Conclusions

**Figure 6.21:** Area: Model of the city of Lausanne, color is assigned to buildings based on the area they occupy, from smaller (blue) to bigger (red).

**Figure 6.22:** Height: Model of the city of Lausanne, color is assigned to buildings based on their height, from smaller (blue) to bigger (red).
Figure 6.23: Volume: Model of the city of Lausanne, color is assigned to buildings based on the volume they occupy, from smaller (blue) to bigger (red).
Figure 6.24: Two views of the complete model of the city of Lausanne. Texture has been applied to the rooftops and the ground surface.
6. SEMANTIC MODELING OF TIGHTLY PACKED CITIES
During the previous chapters we have explored the world of urban 3D reconstruction and modeling. Along this path we had the opportunity to review the most relevant articles in the field and tasted some of the discrepancies in the choice of methods and tools among the different authors. The lack of a global framework hampers researchers to develop and test their specific techniques in a global context. This missing component motivated the creation of FIT3D, a Matlab implementation of the first 5 steps of the reconstruction pipeline (see Figure 7.1) as it was discussed in Chapter 4. It aims at bridging the gap between fields, providing a unifying framework. FIT3D is a highly flexible Toolbox built for Matlab. The basic functionality provided is based on well founded research, which has been presented in chapters 3, 4 and 5.

We built FIT3D with education and research in mind, not providing an ultra efficient implementation for 3D reconstruction but rather a comprehensible guiding tool for students and scientists. The toolbox consists of a large number of independent Matlab functions written in a clear fashion, containing a large amount of inline comments. References to the original methods are provided within the code, which is equation friendly (operations and variable names follow the same equation structure as the original papers). In our spirit of integration and distribution, we provide example scripts and data sets for every of the five steps as well as the integration with related projects. Of course sometimes implementing readable code that is also efficient and generic is a difficult task.

FIT3D is a selfcontained Toolbox. We have not implemented all the required algorithms, in particular for image feature extraction and matching, but provide the necessary ones
7. FIT3D TOOLBOX

as external packages. All external packages provided are distributed under some form of Open Source license.

In this Chapter we put FIT3D into a global perspective, comparing with some of the alternatives and discussing how FIT3D completes their shortcomings. We explore some of the tasks that can be accomplished with little effort and finally conclude with some pointers to future versions of the toolbox.

![3D modeling pipeline. Some well known alternatives for each step are pointed out.](image)

**Figure 7.1:** 3D modeling pipeline. Some well known alternatives for each step are pointed out.

7.1 Related work

In recent years the literature on 3D reconstruction has grown considerably, as was presented in Chapter 3. Along with these papers, a number of tools or software packages have been made available to the research community, and in some cases, to the general public. However, these packages are either very specific to the task at hand or are enclosed in a format that makes integration with novel techniques virtually impossible.
7.1 Related work

7.1.1 Photo Tourism and Bundler

Photo Tourism [64] (see Chapter 3) is one of the most well known programs available. It provides a sparse 3D reconstruction from a set of unordered images of the same scene. The techniques used for estimating the relative pose between the images are well known and are also available in a more scientific package called Bundler [64]. Bundler is built in C++ for efficiency and it takes a collection of images, image features and image matches as input and produces a colored sparse 3D point cloud. The initial camera parameters are taken from the Exif (Exchangeable Image File format) data, which must be available, and the procedure relies mostly on solving a large optimization problem. This optimization step is performed with a modified version of the Sparse Bundle Adjustment of Lourakis and Argyros [22]. Bundler is implemented using optimized C and C++ coding for improved performance, which makes the integration with alternative algorithms somewhat complex and tedious. FIT3D presents a number of advantages using novel pose estimation techniques and refinement procedures. Rather than focusing on solving a problem in a black box approach we implement an efficient linear pipeline for robust pose estimation. Additionally, FIT3D provides a set of basic tools to primitive fitting and texture generation as discussed in Chapter 4.

7.1.2 Hartley and Zisserman

The book by Hartley and Zisserman [1] is one of the most relevant references in the field of multiple view geometry and Structure From Motion (SFM). With the second edition they provide some Matlab code to perform a number of operations. However the number of functions is very limited, with a scarce amount of inline comments and only a handful of example scripts. No full 3D models can be obtained from the raw material they provide. FIT3D is similar to this in that the functions perform very specific tasks and are coded in Matlab. Additionally, it provides a larger number of functions with more comprehensive inline comments and references. Data sets and example scripts are also provided. We showed in Chapter 2 that we follow Hartley and Zisserman’s general conventions, therefore integrating the functions provided by them is possible without much effort.
7. FIT3D TOOLBOX

7.1.3 Peter Kovesi

Peter Kovesi [65] also provides a number of Matlab functions for both image processing, multiple view geometry and model fitting. The inline code provided by Kovesi is very generous in the input/output/example structure but somewhat short in some of the inner workings of the functions. No examples scripts are provided. FIT3D overlaps with Kovesi’s work in the area of model fitting and projective geometry. FIT3D extends Kovesi’s work in that it is focused on the full process of 3D reconstruction and provides full examples on the use of the Toolbox.

With respect to the integration to other packages, FIT3D is fully compatible with the Camera Calibration Toolbox of Bouguet [32]. We also provide integration with the SIFT implementation provided by Lowe [5], the open source VLFeat [8], the 5-point algorithm from Nister [34] and PMVS2 [66] through a set of interface functions.

7.2 Some Functionalities of FIT3D

In this section we briefly explore some of the tasks in visual geometry that can be easily accomplished using FIT3D. Except semantic modeling of Chapter 6, all steps required for obtaining a 3D model as we presented them in chapters 4 and 5 can be performed using the Toolbox. For each of the examples provided below, a short introduction is provided along with the results and the few lines of Matlab function calls required to achieve them.

7.2.1 Radial Distortion from Straight Lines

As we discussed in Chapter 4, in our work we employ the camera calibration toolbox from Bouguet [32] using Zhang method [31]. This calibration also provides the radial distortion parameters. However, since the inverse radial distortion model is also useful, FIT3D provides the means to calculate the radial distortion from the distorted image to the undistorted image by means of selecting a set of straight lines in an image. After these lines are selected, the model up to fifth order is calculated and the image is corrected. This is useful when it is not possible to calibrate the camera based on known geometry patterns, for instance in traffic cameras where it would be difficult to visit all locations to record the pattern.
We begin with a distorted image of a street where 3 supposedly straight lines are selected by indicating 3 points on each (see figure 7.2). From this information the distortion parameters are computed using a linear optimization procedure to maximize the straightness of the lines after compensation. Finally the image is corrected. For increased accuracy more lines can be selected and then nonlinear optimization is employed. Results are shown in figure 7.2.

![Figure 7.2: LEFT: distorted image of a street where lines that are straight in real life can be seen. Three of those straight lines consisting of three points are selected and can be seen in red. RIGHT: the selected lines are used to compute the radial distortion parameters which are then used to undistorted the original image resulting in the undistorted image of the street.](image)

> [distParams,undistImg] = getRadialDistortion('street.jpg',3,3);

### 7.2.2 Simple Visual Geometry Through Homography

An excellent way to start understanding visual geometry and camera pose estimation is by understanding the homographic transformation. A homographic transformation, in terms of camera pose, is that in which the camera does not translate but only rotate
around a vertical axis through the focal point. This is a simplification over the more general case of motion where translation occurs. Estimating the homography between two images allows us to stitch them together to form a panoramic image. Solving the homography involves establishing a set of corresponding points in the images. This is achieved using VLFeat feature detector and matching algorithm.

The `stitchPano` function will stitch two images by selecting corresponding features and calculating the homography. This procedure can be chained to sequentially stitch more images. Results of stitching 4 images are shown in figure 7.3.

```matlab
» [panoramic,H] = stitchImages('ind1.jpg','ind2.jpg',0.4);
» imshow(panoramic);
```

**Figure 7.3:** TOP: 4 input images of a mountain view in Switzerland. BOTTOM: stitched images after computation of the image-to-image homography. Vertical lines visible on the right side of the image are caused by two factors: slight differences in the automatic camera setup due to changing conditions when measuring the light in every image; the camera translating slightly on every image instead of only rotation.
7.2 Some Functionalities of FIT3D

7.2.3 General Camera Motion and Outlier Removal

A more general camera motion where translation also occurs can be estimated through the fundamental matrix \( F \). Finding a set of corresponding points and using the 8-point algorithm, the trajectory of a camera can be estimated. This follows precisely the choices we made regarding the reconstruction pipeline as discussed in Chapter 4. RANSAC is used to remove outliers in the set of feature matches and Horn’s method to obtain the camera position \( P \) from the fundamental matrix. Results are shown in figure 7.4, where the motion of a camera is estimated based on 57 images that record an urban scene.

\[
[F, \text{inliers}] = \text{ransacF}(X_1, X_2, K);
\]
\[
P_{\text{all}} = \text{getCameraMatrix}(F);
\]
\[
X_{1i} = X_1(\text{inliers, :});
\]
\[
X_{2i} = X_2(\text{inliers, :});
\]
\[
P = \text{getCorrectCameraMatrix}(X_{1i}, X_{2i});
\]

**Figure 7.4:** TOP: 3 views of a street out of 57 images taken the camera roughly pointing towards the buildings. BOTTOM: Top view of the camera positions (in red) and reconstruction of the SIFT features used to compute the poses.
7. FIT3D TOOLBOX

7.2.4 Linear Scale Estimation

In Chapter 5 we discussed how the estimation of motion based only on image features introduces the local scale problem. We provided one linear solution that requires only one 2D-3D correspondence and an optimal solution for when more evidence is present. The accurate estimation of the local scale is essential in order to reduce the drift in camera motion in large reconstructions.

For this example, we have 3 estimated camera poses $P_1$, $P_2$ and $P_3$ along with 3-frame image feature correspondences $X_1$, $X_2$ and $X_3$. The first camera is at the origin ($P_1 = [I|0]$) and the other to cameras positions are calculated on a frame-to-frame basis using Image-to-Image feature matches. We triangulate the image features between cameras $P_1$ and $P_2$, resulting in the set of space points $X_{3D}$. The scale of the third camera is computed linearly as described in Chapter 5. The small difference in camera position shown in Figure 7.5 serves to emphasize how important accurate motion estimation is at a frame-to-frame scale in order to avoid camera drift over long trajectories.

```matlab
>> [x3d] = findTriangulationLM(X1, X2, P1, P2, K1, K2);
>> [scale, P3scaled] = findScaleLinear(P3, x3d, X2, X3);
```

7.2.5 Camera Pose and Structure Optimization

The process of optimizing the camera pose is commonly known as Bundle Adjustment \[21\] and it involves the refinement of the motion estimate using a numerical iterative approach. This procedure is typically performed through the minimization of a cost function. As we discussed in Chapter 3, many of the state-of-the-art approaches rely heavily on solving large numerical optimization schemes to obtain accurate camera poses. We however approach the problem in two stages. First we estimate the camera-to-camera displacement as accurately as possible, including nonlinear optimization. Then we apply optimization to the complete ensemble of camera positions (though as we showed in Chapter 5, this is not always necessary).
7.2 Some Functionalities of FIT3D

Figure 7.5: Black pyramids: representation of the position of cameras 1, 2 and 3 before scale adjustment. Grey pyramid: camera 3 after scale adjustment. Firstly, the relative displacement between cameras 1 and 2 is computed using, for instance, the 8-point algorithm. This yields a scale free motion of camera 2 with respect to camera 2. Then the displacement of camera 3 with respect to camera 2 is calculated. This yields a scale free displacement represented as the black pyramid in camera 3. Those two camera displacements are on a different scale. We set the reference scale as 1 for the first camera displacement and the scale of the second camera displacement between cameras 2 and 3 is computed, resulting in the scale adjusted position of camera 3 represented as the grey pyramid.

Local Optimization

FIT3D provides the functionality to refine the motion estimate of a camera and its camera calibration matrix given a set of matching features and the previous camera motion estimate. This procedure is described in detail in Chapter 4. In this example we iteratively refine the second camera pose and internal parameters to minimize the sum of reprojection error in the first and second camera. P1 is assumed to be at [I|0]. Since the 8-point algorithm already delivers a good estimation, the optimization typically converges in 1 or 2 iterations. Results are shown in figure 7.6.

```matlab
[P2refined,K2refined] = bundleAdjust(P2,X1,X2,K2);
```
Global Optimization

As discussed in Chapter 4, there is a multitude of variants on designing the function to minimize in a global optimization scheme. We provide an alternative method to minimize the sum of square distances of corresponding reconstructed 3D points. For this example we employ a particularly noisy setup where camera motions and scale are poorly estimated. Given the errors in the reconstructed structure, we apply our global refinement scheme to obtain an improved reconstruction and more accurate camera positions. Results are shown in figure 7.7 where 4 images record a small alley.

\[ (P_{\text{refined}}) = \text{bundleAdjustNFrames3D}( \text{threeFrameMatches}, K_s, P_s, \text{iterations}, \text{maxDistance}); \]

7.2.6 Primitive Fitting and Texturing

Throughout our work (see Chapters 4 and 6) we employ a RANSAC based approach to fit planar polygons to a set of 3D points. We employ this technique to find planes iteratively that are used for modeling or gravity estimation. This procedure is flexible and robust and can easily cope with very large sets of points. In this example we use the data set of a small alley to find planar patches in the 3D point cloud (see Figure 7.8). Each planar patch is assigned a different color. Then the most frontal view is used to texture the resulting set of planes. A complete description of the method employed is presented in Chapter 4. PMVS2 was employed to obtain a dense reconstruction for comparison.
7.2 Some Functionalities of FIT3D

Figure 7.7: TOP: Images used to obtain a scaled consistent motion. BOTTOM-LEFT: resulting 3D structure before optimization. BOTTOM-RIGHT: resulting structure after global optimization. The originally estimated position of the cameras (before optimization) was altered to produce a noisy pointcloud.

» [planes] = fitPlanesRANSAC( X3D, minPoints, maxPlanes, ransacThreshold, ransacIterations);

7.2.7 Full Reconstruction Pipeline

FIT3D implements all the small tasks that are required during the 3D reconstruction process. In the previous sections we have seen how some of these tasks are solved with a couple of simple commands. All these can be combined into a single pipeline where the reconstruction of an urban area can be obtained. In this procedure, we follow the reconstruction pipeline as discussed in Chapter 4. For simplicity and because it is out of the scope of the toolbox, we assume that a calibration matrix \( K \) has been obtained with the Matlab Calibration Toolbox of Bouguet [32].

7.2.7.1 Camera Pose Estimation

Having obtained the calibration parameters and undistorted images, we proceed to compute the scale free poses of the camera. Figure 7.9 shows some undistorted views of an urban area. Two scripts are required for obtaining the relative camera poses. Firstly, the SIFT features are calculated for all cameras. Secondly, these features are used to compute the relative camera poses using the 8-point algorithm and a nonlinear
7. FIT3D TOOLBOX

Figure 7.8: TOP-LEFT: Colored point cloud. TOP-RIGHT: Set of planar patches found with the RANSAC approach. BOTTOM-LEFT: Colored set of planar patches. BOTTOM-RIGHT: Dense reconstruction using PMVS2.

local refinement step. The resulting camera trajectory (see Figure 7.10) is scale free and therefore a reconstruction cannot yet be obtained. All the details of the function calls can be found in the help and the example scripts.

Figure 7.9: A few undistorted views of a street.

```matlab
[F,Files] = getSIFT(strcat(install_path,'MotionEstimationexamplesdataFloriandeSet1img'),8,1,'VL_FEAT');

[Fplus, PcamPlus, PcamX, FplusExtra,KBA] = computeCameraMotion('VL_FEAT',F, 0.4, 1, 2500, K, 8, true, '8pts', 0.001, 500);
```
7.2.7.2 Scale Estimation

After the relative camera poses have been estimated using the 8-point algorithm, we obtain the relative scale. This yields a scale consistent set of camera poses that can be used for reconstruction.

\[ \text{[PcamScaled, pts, scales]} = \text{adjustScaleWith3Frames(Fplus, PcamX, K, 10, 'linear', 1);} \]

7.2.7.3 Reconstruction and Modeling

The first step in the modeling process is obtaining the 3D reconstruction of the SIFT features obtained in all the images. This is achieved by linear triangulation of all the features contained in consecutive images (see Figure

\[ \text{MAP = build3dcMap(Fplus, PcamScaled, K, 1, 30, false, true, false, 300, 0.1, 100, 500, 50, 50, 5, 1);} \]

**Figure 7.10:** Triangulated SIFT features and scale consistent camera trajectory. The images are shown in the computed position.

However, this set of points in space is sparse and difficult to interpret. It also contains empty spaces in areas where no features were found, for instance in the windows and doors. We can apply the modeling methods described in Chapter 4. We employ a RANSAC approach to find planar patches in the point cloud (see Figure 7.11 - TOP). After a number of planes have been found, the most frontal view is used to texture
7. FIT3D TOOLBOX

the patches, obtaining a more visually appealing and rich model (see Figure 7.11 - BOTTOM).

The modeling procedure requires setting a number of parameters. For a complete explanation of the parameters, please consult the documentation provided with the toolbox.

```matlab
» MAP = build3dMap(FplusExtra,PcamScaled,K,Files,distanceFromCamera,plotImagesInSpace,
plotPointCloud,fitPlanes,nofPlanes,planeThreshold,density,closestNPoints,minPoints,ransacIterations,
maxDistance,startFrame);
```

Figure 7.11: TOP: Colored planar patches.BOTTOM: Textured patches.

7.3 Conclusion

In this Chapter we have presented some of the functionality of our FIT3D Toolbox for 3D reconstruction from a set of ordered images. We believe that FIT3D fills a gap in the
set of research tools that will help other scientist and students develop and apply their methods and techniques in a more global context. FIT3D allows them to obtain a full 3D model with little effort during integration. As compared to other relevant tools, FIT3D presents a set of novel approaches for both robust monocular visual pose estimation and refinement by point cloud distance minimization. Even though FIT3D provides functionality for global refinement, the robust pose estimation techniques employed are in most cases sufficient and do not require the global refinement step. This is clearly demonstrated in Chapter 5 where after computing the scale consistent poses of a camera moving around a building block, the error at the end of the trajectory is below 0.5%. Only for those more difficult data sets does it need to be applied. Also, FIT3D is provided in an educative format where special care has been taken in building Matlab functions that are easy to use and understand. Along with the toolbox a set of example scripts and data sets are provided.

The first release of FIT3D implements the first 5 steps of the reconstruction pipeline as described in Chapter 4. A future version of FIT3D will also implement the novel methods for estimating the scale optimally as discussed in Chapter 5 and the semantic modeling techniques described in Chapter 6. While maintaining the educational point of view and the flexibility of the toolbox, the goal is to construct a increasingly versatile set of visual geometry tools to aid the computer vision scientists and students.
7. FIT3D TOOLBOX
Conclusions

In this thesis we have treated the problem of large scale city-size reconstruction and modeling using a monocular camera. The proposed methods were designed to work in urban environments in order to obtain a 3D city model including the ground, the buildings and the rooftops. In order to record as much as possible of the buildings, the camera was moving at the street level, pointing towards the facades and moving in the direction of the streets. The resulting sequence of images pictured man-made objects, mostly facades and ground, which are typically flat. The images were recorded by either hand-holding the camera while walking or employing a vehicle mounted camera and driving around the city. In both cases, the camera would travel parallel to the ground with a mostly forward motion.

Most state-of-the-art reconstruction approaches rely heavily on numerical optimization in order to solve a set of geometrical constraints and to obtain a 3D reconstruction. Even though these methods have been shown to work well, they are computationally expensive and the details of the reconstruction process stay hidden behind the optimization procedure. These methods typically produce a set of points in space, the point cloud. Some techniques exist to model urban areas with 3D shapes, moving up from simple points. These methods analyze point clouds, aerial images or GIS data in order to obtain models of the buildings. These techniques have been shown to work on suburban areas where buildings are separated from each other, though they are not suited for tightly packed cities where the boundaries of buildings are sometimes indistinguishable. A large scale semantic model of a city has potential applications in the fields of forensic investigation, assisted or autonomous navigation, city surveying or urban planning. The
goal of our research was twofold. Firstly, we wanted to obtain an accurate, fast and inexpensive method to perform 3D reconstruction. Secondly, we wanted to obtain a semantic 3D model of urban areas.

These two goals attempt to answer the questions we posed in Chapter 1, namely: What are the best algorithms? How many steps are in the reconstruction procedure? How do the choices that we make in each step affect the overall result? Is it possible to obtain accurately the positions in space of the images used for the reconstruction? Do we need an optimization procedure to refine the estimation of the model? Which representation to use? Are colored points in space a sufficiently rich descriptor for a scene? And finally, how do we bridge the gap between representation and understanding?

We approached the first of our goals by taking a detailed look into each of the required steps for 3D reconstruction. In order to do so, we designed a 6-step reconstruction pipeline (section 4.1). We analyzed the available algorithms and made the choices that were best suited for urban reconstruction (sections 4.2 - 4.7). We motivated the use of the 8-point algorithm given its better performance for typical motions of the camera in the urban environment (section 4.3.2.2). We proposed an inexpensive local optimization for every pair of cameras (section 4.5.1). By assuming the previous camera position and parameters had already been optimized, we proposed a chain optimization in which each camera was optimized sequentially. This procedure improved the estimated camera position and parameters at a low computational cost. Since cameras were referenced locally with respect to the previous camera, the procedure could be easily parallelized to obtain a fast and accurate motion estimation.

The motion estimation procedure revealed the scale problem (section 4.3.3): when estimating the translation direction of a camera using Image-to-Image approaches, the scale of the translation could not be recovered. Image-to-World methods implicitly solve the scale, though we showed the lower overall performance when estimating the rotation and translation. Instead of employing Image-to-World approaches to estimate the camera position, we employed the 8-point algorithm to estimate the rotation and a scale free translation. Then we proposed a method for estimating the scale with only one 2D-3D correspondence (section 5.3.2). Our method provided a closed form Least Squares solution when more than one correspondence was available. Furthermore, we provided a first order error analysis (section 5.4) and derived an optimal Maximum Likelihood method to estimate the scale very accurately (section 5.5.2). Our proposed method
performed significantly better (section 5.6) than alternative approaches in estimating both the camera positions and the scale of the translation.

The 8-point algorithm is known to suffer from degeneracies, in particular when all points lie in a plane on the scene. For these special cases, we proposed an optimization procedure that works with at least three images (section 4.5.2). We reconstructed corresponding matching points from two pairs of images. This produced two sets of corresponding 3D points. The optimization of the camera positions and parameters worked by minimizing the distance of the corresponding points in 3D space. We showed how this procedure could overcome the points in a plane problem and how it could be extended to optimize globally with a lower computational expense than standard Bundle Adjustment methods.

In our setting, the camera was moving mostly forward, parallel to the ground. If the camera would not move mostly forward, the 5-point algorithm should be employed to estimate the camera rotation and translation, instead of the 8-point algorithm. If the camera would not move parallel to the ground, methods 3 or 4 from Chapter 5 could be employed to obtain an optimal solution for the local scale. Local and global refinement techniques do not depend on the urban assumptions and only on the visibility of image features across different images. If the overlap between three consecutive images would be greatly reduced, the global optimization would suffer since it relies on matching features across 3 frames. In the same way the performance of local optimization would be reduced if the overlap between consecutive images were small.

We provided a theoretical basis for our proposed methods and showed experimental results to support them. We recorded an urban scene with a handheld camera while taking a stroll (section 5.6.2). For that 200+ meters trajectory the error was 0.14%, and more importantly, no global optimization was required. Similar results were shown with the wellknown P11 dataset. Our proposed six step reconstruction pipeline provided an alternative method to standard numerical optimization procedures in order to obtain fast 3D reconstructions of urban areas from sequences of images. Additionally, we implemented all methods in Matlab and created our freely available Toolbox called FIT3D (Chapter 7).

The second goal of our research was to model the 3D reconstructions by means of semantics. In particular, we were interested in estimating the direction of gravity (or the vertical direction), the topological map of the city, and a watertight model for each
building. These semantics provide fundamental information for machine understanding of the city landscape. We proposed a basic modeling method to fit and texture planar patches to the point clouds (section 4.6.2). These patches, while providing a visually appealing model, did not introduce any semantics. The fast and robust plane fitting method however was employed to find the facades in the point cloud. By assuming that, on average, facades are built perpendicular to gravity, we were able to find very accurately the direction of gravity (section 6.3). This direction is potentially useful for city scale surveying: forensic investigation, trajectory analysis, flooding scenarios, rainfall analysis, etc. Then we proposed a set of prototype methods to estimate the remaining semantic elements. We employed the direction of gravity to model the topological map of the city (section 6.5). By establishing a grid under the point cloud, we grew the grid in the direction of gravity so that it would fit the point cloud. Smoothing was applied to obtain a surface that represented the topological map of the city. The final semantic element we wanted to introduce was a watertight model of the buildings in the city. For this, we employed GIS data to obtain roughly the outline of each building (section 6.2.1). Then, we analyzed the distribution of the distances between the points in the point cloud and the outline of the building to find the rooflines (section 6.6). This allowed us to estimate the height of the facades to create a building model with a flat roof. Then, employing aerial images of the buildings, we extracted a set of control points for which the height was estimated to obtain a geometrical model of the roof, which was connected to the facades (section 6.7).

Our proposed semantic model represents a significant step towards automatic modeling of city. Our method is more robust than alternative techniques and less computationally expensive. It relies on the assumption that on average, the facades are built perpendicular to gravity. The estimation of the topological map is a true novelty. Related methods either assume the surface to be flat, or estimate ground points by analyzing the height change in the neighboring area, without estimating a complete surface. We showed experiments for the city of Lausanne and compared the resulting topological map with an isoline map. Most relevant ground features appeared to be correctly estimated. Our method, thanks to the smoothing stage, is able to roughly estimate the areas where no points were available. A topological map, together with the direction of gravity, is a powerful tool for large scale analysis and simulation on a city area. Final
building models were estimated. The height of the facades and the shape of the rooftop compared for two buildings.

The semantic modeling methods rely on point clouds of urban areas obtained from both the ground and the air. We assumed the ground based point clouds contain mostly points of the facades of buildings, which is essential for the estimation of the direction of gravity. These facades are assumed to be on average perpendicular to the direction of gravity, which is assumed to be the same for the complete model. If the ground based point cloud contained more points of the ground than facades or the facades were not perpendicular to one vertical direction, the gravity could not be estimated reliably (our proposed method for the estimation of gravity would not work on the Death Star in outer space). The preliminary methods presented to obtain the watertight models of buildings rely on the density, distribution and accuracy of the points. For the estimation of the height of facades, the outline of the building is assumed to roughly match the points in the area. If the outline in the GIS were oversimplified and would not fit the points, the height could not be estimated. The accuracy of the rooftop geometry depends on the estimated facade height and the accuracy of points that belong to the roof.

In this thesis we have provided the theoretical and experimental grounds to support our contributions in 3D reconstruction and modeling. We designed a reconstruction pipeline that can accurately recover the structure of an urban scene from sequences of images without the need for expensive numerical optimization. We provided a valuable insight into the steps required for reconstruction that would allow a real time implementation. This allowed us to identify the local scale problem. We developed the theoretical analysis of the error propagation and proposed an optimal method to estimate the local scale that outperforms state-of-the-art approaches. After obtaining a fast reconstruction procedure, we focussed on estimating semantics. We proposed preliminary methods to obtain valuable information at a large scale, obtaining a semantic 3D model of a city. These methods provide the starting steps towards a meaningful representation of a city. Our proposed novel methods for the estimation of the topological map and the height of facades significantly contribute to the tasks of automated city modeling.
8. CONCLUSIONS

8.1 Future Work

One of the most pressing challenges in the methods presented for semantic modeling is evaluation. In order to be able to use these methods across the application fields, it is necessary to understand how accurate they are and under which conditions they will break. We have shown some preliminary evaluation for the estimation of gravity, which we evaluate with the measurements obtained with an INS sensor. A deeper error analysis would be desired involving the sensor accuracy, number of readings, etc.

For the evaluation of the ground surface, we showed a topological map of the city of Lausanne. A numerical evaluation would also be desired. A more detailed topological map could be obtained along with the height of isolines. Then the registration could be obtained based on GPS information in order to obtain a numerical evaluation. The evaluation of the methods described to estimate the height of facades and the geometry of rooftops is even more challenging due to the lack of availability of ground truth information. If accurate images are available at the ground level with known camera parameters (position and calibration), the height of buildings could be obtained by manually indicating the ground and roofline points, and then establishing a real life measure based on some known measurements (for instance the width of a window). This however would be a painstaking job and could only be done for a handful of buildings. More accurate laser readings could be employed, though at this time it is difficult to envision an accurate and reliable large scale ground truth measuring process.

The same issue affects roof topology. A hand annotated method could be employed for the 2D geometrical structure of roofs. Applying this at a large scale and obtaining the real 3D structure seems however difficult.

Images are commonly used in the literature for estimating features and computing camera poses. We used then in the methods presented as a source of features to obtain points in space and also features to obtain the geometry of roofs. However, images contain a lot of semantic information: doors, windows, cars, facade outlines, etc. Many of these elements could be extracted employing image analysis and machine learning techniques. Once these elements are found in the images, they could be mapped to the 3D model of the city.

We have employed in our methods for modeling large scale point clouds. However, the original images that were used to obtain the point clouds were not available. An...
alternative source of information would be Google Street View images. These images are geolocalized, therefore the registration with our model would be straightforward. The challenge would be the lack of camera parameters and lens information, without which the mapping from image to model would be very difficult.

For many years, state-of-the-art methods have been focused on obtaining accurate camera position and calibration parameters. The semantic information in the images and the models had been somehow forgotten. We are now at a point where camera pose and calibration can be estimated very accurately and very fast. We believe that the emphasis of city modeling is going to be semantics and the ability to use those for large scale applications.
8. CONCLUSIONS
Sources of the Chapters

Some of the contents and results of the chapters of this thesis are partially based or inspired on the following publications:


9. SOURCES OF THE CHAPTERS


Summary

The objective of this thesis was to research and develop the methods and techniques for obtaining large scale semantic models of urban areas. In particular we were interested in fast and accurate techniques that do not rely heavily on numerical optimization. To achieve this, we focused on employing monocular cameras to record the environment due to their simplicity, flexibility and low cost.

In Chapter 1 we presented the field of 3D reconstruction and discussed the camera paradigm. We also introduced by means of intuition the reconstruction procedure that aims at recovering the 3D structure recorded in images. The challenges ahead were also introduced, namely how to design a reconstruction pipeline, how to recover the scale between different images and how to model an urban scene semantically.

In Chapter 2 we introduced some visual geometry concepts. We described the camera imaging process by means of the pinhole camera model. We also described a more generic camera where lenses are used to improve the quality of the image. We chose a coordinate system where objects, images and cameras can be referenced. These concepts, along with basic concepts in computer vision serve as the groundwork of the methods we have developed.

In Chapter 3 we discussed the history and state-of-the-art in the reconstruction literature, concluding that the most advanced methods solve the reconstruction problem with a numerical optimization procedure. This black box approach attempts to establish constraints between camera positions and image features in order to minimize a cost function, solving then the camera positions and obtaining a 3D reconstruction. While
these methods work well, we believe that more attention should be devoted at each of
the individual steps of the reconstruction procedure.
Thus in Chapter 4 we proposed a 6-step reconstruction pipeline. We discussed each of
the steps for the task of 3D urban reconstruction with the goal of obtaining a fast an
accurate reconstruction pipeline. We treated in detail the first five steps, reasoning and
motivating our choices of algorithms. Semantic Modeling was left a later chapter. We
provided an overview of available methods for each of the steps. With respect to the
estimation of the camera motion, we described the scale challenge, which is treated in a
separate chapter. In the end, we provided a complete reconstruction pipeline for urban
scenes where no large numerical optimization is required.
In Chapter 5 we described in detail the local scale problem. We reviewed the current
solutions, which typically involve using Image-to-World methods or real life observations
regarding the speed of the camera or its position with respect to the ground. We
begun with definitions regarding the error present in image features, points in space
and camera rotation and translation. We also detailed our camera position estimation
algorithm based on our discussion and algorithmic choices from Chapter 4. Then we
derived a Least Squares closed form solution for estimating the scale where only one
Image-to-World match is required. Furthermore, there are different ways to solve the
system of equations. We proposed to solve the scale \( s \) using only the motion of features
in the direction of \( x \) (Method 1), which is the direction in which the camera moves
the most in an urban environment. Then we performed a first order error propagation,
which allowed us to compute the error of the estimation of the local scale \( s \). Finally, we
proposed a Maximum Likelihood estimation to obtain an optimal scale. We provided
a comparison with state-of-the-art methods, where we prove the superiority of our
method.
In Chapter 6 we focused on extracting semantics for an urban 3D model. In partic-
ular, we wanted to obtain the direction of gravity, the surface over which the city is
constructed, a watertight model of the buildings from ground to roof line (where the
facade meets the roof), and finally a geometric model of the roof tops. We proposed
to use three sources of information: a ground-and-air based point cloud reconstruction,
an aerial image obtained from Google Maps and a GIS map obtained from Open Street
Map. Firstly, we proposed a fast RANSAC-based method for estimating the direction
of gravity (or the vertical direction). We assumed that, on average, the facades in the
city are built perpendicular to this direction. We compared the resulting direction with the direction provided by an INS sensor, resulting in an error of 0.0031 degrees. Secondly, we proposed a four-stages method to estimate the topological map of the city. Our method produced an accurate elevation map of the ground over which the city is built, which we compared with an isoline map available online. Thirdly, we proposed a method to estimate the height of the roof line of each building. The method works by means of observing how the distribution of facade-pointcloud point distance changes with height. The resulting heights were applied over the complete set of buildings of a city. Finally, we modeled the rooftops geometrically by analyzing the aerial images of the buildings, obtaining a complete watertight model for each building.

In Chapter 7 we presented our toolbox FIT3D, which is a Matlab implementation of the work we have developed for urban 3D reconstruction. We compared the toolbox with other well known available packages and showed some examples of the functionalities contained within the toolbox.
10. SUMMARY
Appendix A

Sample Code

Below is the code for the 8-point algorithm as described in Chapter 4.

```matlab
function F = eightpoint(X1, X2)
    % Normalize the points following the centroid normalization recommended
    % by Zisserman (p109, p282)
    [X1t,T1] = normalize2Dpoints(X1);
    [X2t,T2] = normalize2Dpoints(X2);
    % Arrange the matrix A. Only the first two equations are needed as the
    % last one is redundant
    A = zeros(size(X1t,2),9);
    for i=1:size(X1t,2)
        A(i,:) = [X2t(1,i)*X1t(1,i), X2t(1,i)*X1t(2,i), X2t(1,i),
        X2t(2,i)*X1t(1,i), X2t(2,i)*X1t(2,i), X2t(2,i), X1t(1,i),
        X1t(2,i), 1];
    end;
    % Find the fundamental matrix F' in 2 steps
    if(sum(sum(isnan(A)))==0)
        % Compute Λ. Λ is closest to A in frobenius form
        [U,D,V] = svd(A);
        %D(end,end) = 0;
        %Af = U*D*V';
        Af = A;
        % Linear solution: Determine F from the singular vector corresponding
        % to the smaller singular value of Λ
        [Ua, Da, Va] = svd(Af);
        % The matrix F' is composed of the elements of the last vector of V
        f = Va(:,end);
        % Reorganice h to obtain H
    end;
end;
```
A. SAMPLE CODE

Fp = reshape(f,3,3)';
% Constraint enforcement: Replace F by F' such that det(FÂť)=0 using
% SVD
[Uf,Df,Vf] = svd(Fp);
Df(end,end) = 0;
Ff = Uf*Df*Vf';
% Denormalization
F = T2'*Ff*T1;
else
  'ERROR!!!'
  F = [0/0,0/0,0/0;0/0,0/0,0/0;0/0,0/0,0/0/0/0];
end;
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


[38] F. MORENO-NOGUER, V. LEPETIT, AND P. FUA. Accurate Non-Iterative O(n) Solution to the PnP Problem. In IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, October 2007. 44, 45, 63, 76


[40] MARTIN BUJNAK, ZUZANA KUKELOVA, AND TOMÁS PAJDLA. A general solution to the P4P problem for camera with unknown focal length. In CVPR, 2008. 45, 63, 76

[41] LIHONG ZHI AND JIANLIANG TANG. A Complete Linear 4-Point Algorithm for Camera Pose Determination 1). 45


REFERENCES


REFERENCES


