DIRECT BUNDLE ADJUSTMENT OF VIDEO

A Thesis in
Electrical Engineering

by

John P. Morgan Jr.

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2016
The thesis of John P. Morgan Jr. was reviewed and approved* by the following:

Richard L. Tutwiler
Associate Professor of Acoustics
Thesis Adviser

William E. Higgins
Distinguished Professor of Electrical Engineering

Kultegin Aydin
Professor of Electrical Engineering
Head of the Department of Electrical Engineering

*Signatures are on file in the Graduate School.
Abstract

Given a set of 3D points seen by multiple cameras, bundle adjustment is the process of jointly optimizing the location of the 3D points and the pose of each camera. Often the image coordinates of the points are known, usually via feature detection and matching, and the reprojection error can be minimized. Here the focus is instead on intensity-based methods that directly minimize the intensity residual between viewpoints. These methods require a good initialization, which is implicitly provided by video because the transformation between frames is small allowing each frame to initialize the next frame. The goal of this thesis is to analyze the feasibility of intensity-based bundle adjustment for dense reconstructions and compare this to recent work on multi view stereo.

The proposed method optimizes the inverse depth of a reference frame rather than explicit 3D points, which leads to a significantly more efficient algorithm. Dense reconstructions are possible because the sparsity of the bundle adjustment problem can be exploited to achieve linear runtime with respect to the number of points. The greatest advantage over existing algorithms is that special initialization cases are not required. Each point is tracked through the video, and therefore the inverse depth and camera pose are always a feasible explanation for what was observed. It will be shown that very few frames are required to converge to an accurate estimate. Dense reconstructions do not run in real-time; however, the method is equally applicable with sparse sets of points for real-time pose estimation.
# Contents

List of Figures ................................................................. v
List of Tables ................................................................. vi
List of Symbols ................................................................. vii

Chapter 1. INTRODUCTION .............................................. 1
1.1 Problem Statement ....................................................... 1
1.2 Related Work ............................................................ 1
1.3 Outline ................................................................. 2

Chapter 2. BACKGROUND ............................................... 4
2.1 Rigid Transformations .................................................. 4
  2.1.1 Minimal Representation of Rotation ................................ 4
  2.1.2 Minimal Representation of Rigid Transformations ............. 5
2.2 Two View Geometry .................................................... 6
2.3 Non-Linear Least Squares ............................................ 7
  2.3.1 Unconstrained Solution ........................................... 8
  2.3.2 Bound Constraints ................................................ 8

Chapter 3. MONOCULAR SLAM ......................................... 11
3.1 Stereo Matching ....................................................... 11
  3.1.1 Point Correspondences ........................................... 11
  3.1.2 Triangulation ..................................................... 12
  3.1.3 Bayesian Estimation of Depth .................................. 13
3.2 Pose Estimation ....................................................... 15
  3.2.1 Compositional Approach ....................................... 16
  3.2.2 Inverse Compositional Approach ............................... 16
  3.2.3 Jacobian of the Warp Function ................................. 17
  3.2.4 Implementation Details ......................................... 17
3.3 Joint Estimation of Pose and Inverse Depth ....................... 19
  3.3.1 Partitioned Non-Linear Least Squares ......................... 20
  3.3.2 Jacobian of the Warp Function ................................. 22
  3.3.3 Implementation Details ......................................... 24

Chapter 4. RESULTS ....................................................... 25
4.1 Sample Reconstructions ............................................. 25
4.2 Experimental Results ................................................. 28

Chapter 5. CONCLUSION ................................................ 34

Appendix A. Inverse Compositional Pose Estimation Algorithm .. 35
Appendix B. Joint Estimation Algorithm ................................ 36

References ................................................................. 37
List of Figures

2.1 Two view geometry ......................................................... 7
2.2 General and box constraint projection .................................. 9
3.1 Epipolar geometry ......................................................... 12
3.2 Triangulation ............................................................... 13
3.3 Effect of variance on depth distribution ............................... 14
3.4 Effect of inlier ratio on depth distribution ............................. 14
3.5 Homography ............................................................... 18
3.6 Relation of epipolar line to inverse depth Jacobian .................... 23
4.1 Over table groundtruth depth and reference frame .................... 26
4.2 Fast motion groundtruth depth and reference frame .................. 26
4.3 Freiburg2 xyz depth and reference frame ............................... 26
4.4 Over table estimated depth and back projected point cloud ............ 27
4.5 Fast motion estimated depth and back projected point cloud ........... 27
4.6 Freiburg2 xyz estimated depth and back projected point cloud ......... 27
4.7 Over table precision and completeness ................................. 29
4.8 Fast motion precision and completeness ............................... 30
4.9 Over table convergence of precision .................................... 31
4.10 Fast motion true and estimated trajectory .............................. 32
4.11 Over table runtime ......................................................... 33
List of Tables

4.1 Absolute trajectory error .......................................................... 30
List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega$</td>
<td>Bound constraints on $\beta$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Vector of parameters being optimized</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Levenberg-Marquardt update to $\beta$</td>
</tr>
<tr>
<td>$\delta_a$</td>
<td>Levenberg-Marquardt update to pose</td>
</tr>
<tr>
<td>$\delta_b$</td>
<td>Levenberg-Marquardt update to depth</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>Residual</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Angle of rotation about axis $\omega$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Levenberg-Marquardt damping factor</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Axis-angle representation of rotation</td>
</tr>
<tr>
<td>$[\omega]_x$</td>
<td>Skew-symmetric representation of $\omega$</td>
</tr>
<tr>
<td>$A$</td>
<td>Portion of Hessian corresponding to the pose update $a$</td>
</tr>
<tr>
<td>$A'$</td>
<td>Damped $A$ matrix</td>
</tr>
<tr>
<td>$A(\beta)$</td>
<td>Active set of parameters that will not be updated</td>
</tr>
<tr>
<td>$B$</td>
<td>Portion of Hessian corresponding to the inverse depth $b$</td>
</tr>
<tr>
<td>$B'$</td>
<td>Damped $B$ matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>Portion of Hessian corresponding to the pose update $a$ and inverse depth $b$</td>
</tr>
<tr>
<td>$E$</td>
<td>Sum of squared residuals</td>
</tr>
<tr>
<td>$I$</td>
<td>Intensity image</td>
</tr>
<tr>
<td>$H$</td>
<td>Hessian matrix of second derivatives</td>
</tr>
<tr>
<td>$H'$</td>
<td>Damped Hessian matrix</td>
</tr>
<tr>
<td>$\tilde{H}$</td>
<td>Hessian matrix without active parameters</td>
</tr>
<tr>
<td>$H_\varepsilon$</td>
<td>Hessian of $\varepsilon$ with respect to $\beta$</td>
</tr>
<tr>
<td>$H_E$</td>
<td>Hessian of $E$ with respect to $\beta$</td>
</tr>
<tr>
<td>$\tilde{H}_E$</td>
<td>Gauss-Newton approximation of $H_E$</td>
</tr>
<tr>
<td>$\tilde{H}'_E$</td>
<td>Damped Gauss-Newton approximation of $H_E$</td>
</tr>
<tr>
<td>$\tilde{H}''_E$</td>
<td>Damped Gauss-Newton approximation of $H_E$ without active parameters</td>
</tr>
<tr>
<td>$H_{i\to j}$</td>
<td>Homography that maps points in frame $i$ to frame $j$</td>
</tr>
<tr>
<td>$J$</td>
<td>Jacobian matrix of first derivatives</td>
</tr>
<tr>
<td>$J_\varepsilon$</td>
<td>Jacobian of $\varepsilon$ with respect to $\beta$</td>
</tr>
<tr>
<td>$J_\omega$</td>
<td>Jacobian of $\tilde{\omega}$ with respect to $\omega$</td>
</tr>
<tr>
<td>$J_{a,i}$</td>
<td>Jacobian of $\varepsilon_i$ with respect to $a$</td>
</tr>
<tr>
<td>$J_{b,i}$</td>
<td>Jacobian of $\varepsilon_i$ with respect to $b_i$</td>
</tr>
<tr>
<td>$J_v$</td>
<td>Jacobian of $\tilde{\tau}$ with respect to $\nu$</td>
</tr>
<tr>
<td>$J_{z,i}$</td>
<td>Jacobian of $\tilde{\tau}$ with respect to $b_i$</td>
</tr>
<tr>
<td>$K$</td>
<td>Intrinsic camera matrix</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of residuals being optimized</td>
</tr>
<tr>
<td>$P(\beta)$</td>
<td>Projection of $\beta$ onto $\Omega$</td>
</tr>
<tr>
<td>$R$</td>
<td>Rotation matrix</td>
</tr>
<tr>
<td>$\text{SE}(n)$</td>
<td>$n$ dimensional special Euclidean group</td>
</tr>
<tr>
<td>$\text{SO}(n)$</td>
<td>$n$ dimensional special orthogonal group</td>
</tr>
<tr>
<td>$T$</td>
<td>Transformation matrix</td>
</tr>
</tbody>
</table>
\( V \) Exponential map of translational component

\( W(\cdot) \) Warp function that maps coordinates in frame 1

\( X \) Point in 3-space

\( \bar{X} \) Homogeneous representation of a point in 3-space

\( a \) Lie algebra of the pose update being optimized

\( b \) Vector of inverse depths being optimized

\( g_a \) Gradient of \( E \) with respect to pose update \( a \)

\( g_b \) Gradient of \( E \) with respect to inverse depth \( b \)

\( g \) Gradient of \( E \) with respect to \( \beta \)

\( \dot{g} \) Gradient \( g \) without active parameters

\( g_a \) Gradient of \( E \) with respect to pose update \( a \)

\( g_b \) Gradient of \( E \) with respect to inverse depth \( b \)

\( r_x \) \( x \) component of \( \omega \)

\( r_y \) \( y \) component of \( \omega \)

\( r_z \) \( z \) component of \( \omega \)

\( \text{se}(3) \) Lie algebra

\( \text{so}(3) \) Skew-symmetric group

\( t \) Translation

\( v \) Translational component of Lie algebra

\( x \) Image coordinate

\( \bar{x} \) Homogeneous representation of image coordinate \( x \)

\( y \) Image coordinate returned by the warp function \( W \)

\( \bar{y} \) Homogeneous representation of image coordinate \( y \)

\( z \) Depth
Chapter 1

Introduction

1.1 Problem Statement

This work aims to analyze the feasibility of intensity-based bundle adjustment for simultaneous localization and mapping (SLAM). The goals of SLAM are twofold: maintain a 3D map of the scene and track the sensor pose within that map. Here the map will be represented as the depth of each pixel in some reference view. The sensor is a single moving camera, which is referred to as monocular SLAM or structure from motion. Monocular reconstructions add the challenge of scale ambiguity, which means the absolute scale of the reconstruction cannot be determined and the relative scale is subject to drift.

The proposed approach directly uses image intensity to jointly optimize pose and structure as opposed to other methods that update pose and structure sequentially. Intensity-based (direct) methods are advantageous because they utilize all of the image information, whereas feature based methods are limited to the information specific to the feature (e.g. corners). Only direct methods are considered here. The pose accuracy, reconstruction accuracy, and runtime of the proposed method will be evaluated. Runtime is important because real-time SLAM is of particular interest for robotic navigation and interactive virtual or augmented reality.

The use of video is an important constraint because each frame initializes the next frame, which requires the transformation between frames to be small. Images taken at the frame rate of video implicitly satisfy this constraint. The focus will be entirely on local segments of video used to update a single depth map; large scale reconstructions that require multiple depth maps and loop closure are not considered. A local segment of video is a sequence of frames that have sufficient overlap with the reference frame.

1.2 Related Work

Several direct monocular SLAM algorithms have recently been proposed [1,2,3]. The authors of [1] develop a cost volume that stores the average intensity residual for each pixel in a reference image at a discrete number of inverse depths; a regularized depth map can be obtained from this volume by using a variational approach. Pose estimation is interleaved with depth estimation and is carried out by minimizing the intensity residual between frames. The algorithm must be bootstrapped because building the cost volume requires knowledge of the pose, but estimating the pose requires knowledge of the depth (and hence the cost volume). Real-time performance is achieved using GPU acceleration.

The authors of [2] directly estimate an inverse depth map; however, each depth measurement is represented as the mixture of a Gaussian and uniform distribution [4]. Ideally the Gaussian
distribution is centered at the true depth while the uniform distribution accounts for erroneous estimates. The individual estimates used to update the model are obtained via stereo matching with each new frame. Pose estimation is carried out in the same manner as [1]; though only a sparse set of FAST features [5] are used. Again bootstrapping is necessary because stereo matching requires known pose whereas pose estimation requires known depth. Real-time performance is achieved using GPU acceleration.

The authors of [3] present a large-scale algorithm that maintains several keyframes, each scaled such that the average inverse depth is one. Only pixels with sufficient image gradient are considered. Local tracking is performed similarly to [1] using the nearest keyframe; however, the intensity residual is variance normalized. Depth measurements are represented as a Gaussian distribution and individual estimates are obtained via stereo matching. Interestingly, the authors were able to initialize the depth randomly rather than bootstrapping, though convergence requires sufficient camera translation in the first seconds of video and takes many frames. Real-time performance is achieved without GPU acceleration.

All of these algorithms update the depth and pose separately even though each update relies on the other. For this reason initialization must be considered separately or relies on special conditions, both of which are undesirable. Jointly optimizing depth and pose eliminates the need for special initialization cases and is referred to as bundle adjustment [6], but it is rarely used for dense reconstructions due to the size of the optimization problem. The authors in [7] propose a method to apply range constraints to the bundle adjustment parameters, which can be used to enforce positive depth. The goal of this work is to show that bundle adjustment is feasible for SLAM when properly formulated and sparseness is exploited.

1.3 Outline

Chapter 2 presents the necessary background for developing the results in chapter 3. First the Lie algebra of rigid transformations is introduced, which is a minimal representation that can be used during optimization. Next, a brief overview of the pinhole camera model is given that will be used to develop the residuals in chapter 3. To conclude it is useful to develop the general solution of non-linear least squares with and without bound constraints because all of the optimizations will be posed as non-linear least squares.

Chapter 3 discusses how to update depth and pose individually and then presents the proposed joint optimization. To start, stereo matching for depth estimation and direct image alignment for pose estimation are covered. Direct image alignment is then extended to the joint estimation of inverse depth and pose. It is important to consider depth and pose separately because the proposed method must address the shortcomings. Additionally, pose estimation is used to refine the initialization prior to joint optimization, and stereo matching can be used to create a dense reconstruction if only a sparse set of points were jointly optimized.

Chapter 4 summarizes the results. Qualitative reconstructions of datasets from [8] and [9] are presented followed by a quantitative analysis of the reconstruction accuracy, pose accuracy, and runtime. To evaluate the reconstruction only the synthetic datasets of [8] are used; synthetic data
is necessary because a 100% complete depth map is required in addition to groundtruth pose. The reconstruction accuracy of the proposed method will be compared to [2]. To evaluate pose, the absolute trajectory error of each dataset is presented. Finally, linear runtime with respect to the number of points will be demonstrated by performing reconstructions of various densities. It will also be shown that the reconstruction accuracy does not suffer if less points are used.

Chapter 5 will offer some conclusions and directions for future work. The algorithms for pose and joint estimation are presented in detail in appendices A and B.
Chapter 2

Background

This chapter will present the relevant background needed to develop the results in chapter 3. To start, section 2.1 defines the notion of camera pose and presents the Lie algebra which will be used during optimization. Section 2.2 presents the relevant two view geometry and establishes a consistent frame of reference. This is important because any two frames in the video could be chosen. All of the optimization problems will be posed as non-linear least squares, so section 2.3 concludes by discussing how to solve such problems in general with or without bound constraints.

2.1 Rigid Transformations

The pose of a camera refers to its rotation $R \in \text{SO}(3)$ and translation $t \in \mathbb{R}^3$ with respect to some global frame of reference where $\text{SO}(n) = \{ R \in \mathbb{R}^{n \times n} \mid RR^T = I, \det(R) = 1 \}$ denotes the special orthogonal group. The combination of a rotation and translation is referred to as a rigid transformation because it preserves the Euclidean distance between any two points. Rigid transformations belong to the special Euclidean group $\text{SE}(n)$ defined as follows

$$\text{SE}(n) = \left\{ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \mid R \in \text{SO}(n), t \in \mathbb{R}^n \right\} \quad (2.1)$$

If homogeneous coordinates are used, then the transformation of some point $X \in \mathbb{R}^3$ by a rigid transformation $T \in \text{SE}(3)$ can be expressed as a matrix multiplication

$$T \bar{X} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} = RX + t \quad (2.2)$$

From here on homogeneous coordinates will be denoted by an overbar. During optimization it is necessary to have a minimal representation of $\text{SE}(3)$ transformations; here the Lie algebra $\text{se}(3)$ will be used as in [1,2,3]. However, first the minimal representation of rotation by the skew-symmetric group $\text{so}(3) = \{ S \in \mathbb{R}^{3 \times 3} \mid S^T = -S \}$ must be introduced.

2.1.1 Minimal Representation of Rotation

Let $\theta$ be the angle of rotation about some axis $\omega = [\tau_x \; \tau_y \; \tau_z]^T$ with $\| \omega \| = \theta$. $\omega$ can be obtained from a rotation matrix $R$ as follows

$$\omega := \log(R) = \frac{\theta}{2 \sin(\theta)} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \quad \text{with} \quad \theta = \cos^{-1} \left( \frac{\text{trace}(R) - 1}{2} \right) \quad (2.3)$$
where $R_{rc}$ indicates the element at row $r$ and column $c$. This is referred to as axis-angle representation, and further analysis can be found in [10]. The skew-symmetric matrix \([\omega]_x \in \text{so}(3)\) is now defined as

\[
[\omega]_x = \begin{bmatrix}
0 & -r_z & r_y \\
 r_z & 0 & -r_x \\
 -r_y & r_x & 0
\end{bmatrix}
\]  

(2.4)

Conversely, a skew-symmetric matrix can be mapped to a rotation matrix using an exponential map $\text{exp:so}(3) \rightarrow \text{SO}(3)$. The matrix exponential of a skew-symmetric matrix takes the following closed-form, known as Rodrigues’ rotation formula

\[
R := \exp(\omega) = I + \frac{\sin(\theta)}{\theta} [\omega]_x + \frac{1 - \cos(\theta)}{\theta^2} [\omega]_x^2 \text{ with } \theta = ||\omega||
\]  

(2.5)

The matrix $R$ is orthogonal and has $\det(R) = 1$, so it is a valid rotation. Additionally, the map is surjective onto $\text{SO}(3)$, which means every rotation has an axis angle representation [10]. Thus the rotation can be minimally parameterized by $\omega$.

**2.1.2 Minimal Representation of Rigid Transformations**

Using the skew-symmetric group $\text{so}(3)$, the Lie algebra of the special Euclidean group $\text{SE}(3)$ can now be defined as

\[
\text{se}(3) = \left\{ \begin{bmatrix} \omega \\ v \end{bmatrix} \mid [\omega]_x \in \text{so}(3), v \in \mathbb{R}^3 \right\}
\]  

(2.6)

and can be mapped to a transformation matrix using an exponential map $\text{exp:se}(3) \rightarrow \text{SE}(3)$

\[
T := \exp \begin{bmatrix} \omega \\ v \end{bmatrix} = \begin{bmatrix} \exp(\omega) & Vv \\ 0 & 1 \end{bmatrix}
\]  

(2.7)

\[
V = I + \frac{\theta - \sin(\theta)}{\theta^3} [\omega]_x + \frac{1 - \cos(\theta)}{\theta^2} [\omega]_x^2 \text{ with } \theta = ||\omega||
\]

During optimization updates will be calculated in $\text{se}(3)$ and integrated in $\text{SE}(3)$, so the inverse mapping is not used. However, for completeness the inverse mapping $\text{log:SE}(3) \rightarrow \text{se}(3)$ is

\[
\begin{bmatrix} \omega \\ v \end{bmatrix} := \log(T) = \begin{bmatrix} \log(R) \\ V^{-1}v \end{bmatrix}
\]  

(2.8)

\[
V^{-1} = I - \frac{1}{2} [\omega]_x + \frac{2 \sin(\theta) - \theta (1 + \cos(\theta))}{2\theta^2 \sin(\theta)} [\omega]_x^2 \text{ with } \theta = ||\omega||
\]

If $\theta \approx 0$ then the following approximations can be made: $\sin(\theta) \approx \theta$, $(1 - \cos(\theta))/\theta^2 \approx 1/2$, and the third terms in $R$ (2.5) and $V$ (2.7) can be ignored yielding
With this simplification the derivative of the small angle transformation matrix with respect to \( \omega \) and \( v \) is trivial. This will be a useful simplification when calculating the Jacobian matrices in chapter 3.

### 2.2 Two View Geometry

A video is a sequence of frames \( I_j : x_j \rightarrow \mathbb{R} \) where \( x_j \in \mathbb{R}^2 \) is the image coordinate. Let \( R_j \in \text{SO}(3) \) and \( t_j \in \mathbb{R}^3 \) be the rotation and translation of the camera during frame \( j \) with respect to the global frame of reference. Assuming a pinhole camera model with known intrinsic camera matrix \( K \in \mathbb{R}^{3 \times 3} \) (from prior calibration), a point \( X_i \in \mathbb{R}^3 \) in the global frame of reference can be projected onto frame \( j \) as follows

\[
\bar{x}_{j,i} = [wu \quad wv \quad w]^T = KR_j^T(X_i - t_j)
\]

\[
X_i = z_{j,i}R_jK^{-1}[X_{j,i} \quad 1]^T + t_j
\]

The rotation and translation transform \( X_i \) to frame \( j \)’s frame of reference and the intrinsic matrix performs the projection from 3-space to image coordinates. Conversely, image coordinate \( x_{j,i} \) can be back projected to \( X_i \) as follows

\[
l_j(\bar{x}_{j,i}) = l_j(proj(\bar{x}_{j,i})) = l_j(x_{j,i})
\]

Figure 2.1 shows the two view scenario considered from here on. The reference frame will be denoted as frame 1 and the index of subsequent frames will be \( j \). Only the relative pose is required, so without loss of generality frame 1’s pose is chosen as \( R_1 = I_3 \) and \( t_1 = 0 \). This simplifies equations (2.10) and (2.11) considerably.

The reconstruction goal is to estimate the depth of every image coordinate in frame 1. If the depth is known, an image coordinate in frame 1 can be located in frame \( j \) by performing a back projection (2.11) from frame 1 followed by a projection (2.10) onto frame \( j \). With \( R_1 = I_3 \) and \( t_1 = 0 \) this becomes

\[
\bar{x}_{j,i} = KR_j^T(z_{1,i}K^{-1}\bar{x}_{1,i} - t_j)
\]
Equation (2.13) can alternately be formulated in terms of the inverse depth \([11]\), which offers several advantages. First, points at infinity can be considered because their inverse depth is simply zero. This drastically increases the range of depths that can be optimized. Second, when the transformation between frames is small, the mapping from frame 1 to frame \(j\) is more linear with respect to the inverse depth \([11]\). In fact, the joint optimization presented here only converges when inverse depth is used. Dividing the homogeneous coordinate \(\tilde{x}_{j,i}\) by the depth \(z_{1,i}\) yields the new coordinate

\[
\tilde{x}_{j,i}' = KR_j^T(K^{-1}\tilde{x}_{1,i} - z_{1,i}^{-1}t_j)
\]  

Homogeneous coordinates are invariant to scale, so (2.14) represents the same image coordinate as (2.13) in terms of the inverse depth. Points at infinity \((z_{1,i}^{-1} = 0)\) arise when there is no translational parallax; from equation (2.14) it is clear that these points cannot be used to determine translation, but are still useful for determining the rotation.

### 2.3 Non-Linear Least Squares

Both pose and joint estimation are formulated as non-linear least squares problems where the goal is to minimize a sum of squared residuals. The general solution to these problems will be presented here and the relevant residuals will be discussed in sections 3.2 and 3.3. Often it is useful to apply bound constraints to the parameter vector, for example the depth is constrained to be positive in section 3.3. The authors of [7] recommend the gradient projection method, which is presented here. Let \(\beta \in \mathbb{R}^m\) be the parameter vector and \(\epsilon(\beta) \in \mathbb{R}^n\) be the residual vector, which is non-linear with respect to \(\beta\). The bound constrained non-linear least squares problem is then

\[
\min_\beta \left\{ E(\beta) = \frac{1}{2} \epsilon(\beta)^T \epsilon(\beta) \mid \beta \in \Omega \right\} \quad \text{with} \quad \Omega = \{ \beta \in \mathbb{R}^m \mid l \leq \beta \leq u \}
\]  

(2.15)
2.3.1 Unconstrained solution

Before developing the constrained solution, the unconstrained case \((l = -\infty, u = \infty)\) will be considered. Let \(f_\varepsilon = d\varepsilon / d\beta \in \mathbb{R}^{n \times m}\) be the Jacobian matrix of first derivatives of \(\varepsilon(\beta)\). The unconstrained solution to (2.15) is obtained by setting the derivative of \(E(\beta)\) to zero

\[
\frac{dE}{d\beta} = g = f_\varepsilon^T \varepsilon(\beta) = 0
\]  

(2.16)

Equation (2.16) is a non-linear system of equations that can be solved iteratively using the Levenberg-Marquardt algorithm. Let \(H_\varepsilon = d^2\varepsilon / d\beta^2 \in \mathbb{R}^{m \times m \times n}\) and \(H_E = d^2 E / d\beta^2 \in \mathbb{R}^{m \times m}\) be the Hessian matrices of second derivatives of \(\varepsilon(\beta)\) and \(E(\beta)\) respectively. Starting with an initial guess \(\beta^{(0)}\) and damping factor \(\lambda\), updates are calculated as

\[
\left(\tilde{H}_E + \lambda \text{diag}(\tilde{H}_E)\right) \delta = -g \quad \text{with} \quad \tilde{H}_E = J_\varepsilon^T J_\varepsilon \equiv (J_\varepsilon^T J_\varepsilon + H_\varepsilon \varepsilon(\beta)) = H_E
\]

\[
\beta^{(k+1)} = \beta^{(k)} + \delta
\]

(2.17)

\(\tilde{H}_E \in \mathbb{R}^{m \times m}\) is the Gauss-Newton approximation of the true Hessian, which is obtained by ignoring the \(H_\varepsilon \varepsilon(\beta)\) term in \(H_E\). This is desirable because \(H_\varepsilon\) is a third order tensor that is expensive to compute and tends to zero if the function is locally linear or the residual is small.

Updates are only kept when they improve the error, which means the undamped Hessian \(\tilde{H}_E\) and gradient vector \(g\) only need to be recalculated when the error decreases and the update is kept. In the above discussion the damping factor \(\lambda\) was kept constant, but to select the best value in practice \(\lambda\) is varied heuristically. After each update if the error improves the damping factor is decreased, otherwise the damping factor is increased. Note that \(\lambda \to 0\) corresponds to Gauss-Newton iteration, whereas \(\lambda \to \infty\) corresponds to steepest descent. This is desirable because the Gauss-Newton algorithm converges quadratically near the minimum, but steepest descent behaves better far from the minimum.

2.3.2 Bound Constraints

Equation (2.17) must now be modified such that the bound constraints are enforced. Define the projection \(P: \mathbb{R}^m \to \Omega\) of parameter vector \(\beta\) into convex set \(\Omega\) as the point in \(\Omega\) that is nearest to \(\beta\). Because \(\Omega\) is simply defined by a set of box constraints, this reduces to clamping each element of \(\beta\) to the upper or lower bound

\[
P(\beta) = \arg\min_p \{||p - \beta|| \mid z \in \Omega\} = \min\{\max\{\beta, l\}, u\}
\]

(2.18)

where min and max are applied element-wise. Figure 2.2 illustrates a non-trivial projection onto some general set and the much simpler clamping onto a bound constrained set.
The parameter vector is projected into \( \Omega \) after each update to ensure the bound constraints are always satisfied. When calculating the update there are two cases to consider: parameters that are strictly within bounds \( \beta_i \in (l_i, u_i) \) and parameters that are at a bound \( \beta_i \in \{l_i, u_i\} \). If \( \beta_i \) is within bounds no special action needs to be taken. If \( \beta_i \) is at a bound, then there are two cases when it should not be updated: \( \beta_i \) is at the lower bound and \( \delta_i \) is negative or \( \beta_i \) is at the upper bound and \( \delta_i \) is positive. In both cases the update will move the parameter out of the feasible set.

For convenience, denote the damped Hessian by \( H' = H + \lambda \text{diag}(H) \). The update to \( \beta_i \) is \( \delta_i = -(\bar{H}_E^{-1} g)_i \), which follows directly from equation (2.17). If a parameter \( \beta_i \) is not going to be updated, then the \( i \)-th column of \( J_\varepsilon \) should be set to zero. However, \( \bar{H}_E' \) (and hence \( \delta \)) depends on \( J_\varepsilon \), which means the sign of \( \delta_i \) must be known before calculating \( \delta \). Unfortunately the effect of \( \bar{H}_E^{-1} \) on the direction of the gradient \( g \) is nontrivial, so determining the sign of \( \delta \) ahead of time is not feasible.

For this reason, the gradient projection method instead considers the steepest descent update \(- g\) when determining which updates to ignore. The active set \( A(\beta) \) of parameters that will not be updated is defined as

\[
A(\beta) = \{ i \mid \beta_i = l_i, g_i \geq 0 \text{ or } \beta_i = u_i, g_i \leq 0 \} \quad (2.19)
\]

Ignoring the updates in equation (2.19) will not guarantee that \( \delta \) is a feasible direction; however, the active set will converge to the true active set in a finite number of iterations [12]. Thus, when used in conjunction with second order updates, the gradient projection method is a heuristic that can improve the convergence rate.

If a parameter is in the active set, the column of \( J_\varepsilon \) corresponding to \( d\varepsilon/d\beta_i \) is set to zero. From (2.16) and (2.17) it is easy to verify that \( g_i \) and the \( i \)-th column and row of \( \bar{H}_E = J_\varepsilon J_\varepsilon^T \) will be zero. However, for \( \delta \) to exist \( \bar{H}_E \) must be non-singular, which is not the case if any of the columns are zero. To avoid this the \( i \)-th diagonal element can be set to any positive value

\[
\bar{H}_{rc} = \begin{cases} 
\bar{H}_{rc}, & r \notin A(\beta) \text{ and } c \notin A(\beta) \\
1, & r = c \\
0, & \text{else}
\end{cases}
\quad \hat{g}_r = \begin{cases} 
g_r, & r \notin A(\beta) \\
0, & \text{else}
\end{cases}
\quad (2.20)
\]
By choosing a positive value the Hessian remains positive definite; in this case, the diagonal elements are chosen to be 1. This does not change the result because $g_i = 0$ and the $i^{th}$ row and column are zero everywhere but the diagonal yielding

$$\delta_i = -\left(\bar{H}_E^{-1}\bar{g}\right)_i = -g_i/(\bar{H}_E'_{ii}) = 0 \quad \text{if} \quad i \in A(\beta) \quad (2.21)$$

The gradient projection update is summarized by equation (2.22) below

$$\bar{H}_E'\delta = -\bar{g} \quad (2.22)$$

$$\beta^{(k+1)} = p(\beta^{(k)} + \delta)$$

As in the unconstrained case, updates are only kept when they improve the error and the damping factor $\lambda$ is varied heuristically (see section 2.3.1 for discussion).
Chapter 3

Monocular SLAM

This chapter will develop depth and pose estimation separately and then present the proposed joint optimization. Section 3.1 is an overview of stereo matching and section 3.2 introduces direct image alignment for pose estimation. Both [2] and [3] use these methods sequentially; however, it will be clear that there is interdependence between depth and pose which makes initialization difficult. Section 3.3 presents the joint optimization and addresses the issue of initialization.

3.1 Stereo Matching

Given two images \( I_1 \) and \( I_j \) at different viewpoints with known transformation between views, the location of points visible in both images can be triangulated. The first step is to find the coordinate in each image that corresponds to the same 3D point. This search can be constrained using knowledge of the epipolar geometry, which is discussed next. The second step is to triangulate the depth of each correspondence, and the final step is to incorporate each new estimate into the global measurement.

3.1.1 Point Correspondences

With just a single view the point \( X_l \in \mathbb{R}^3 \) could lie anywhere on the ray from the camera center through the image coordinate \( x_{l,i} \), which can be seen from the back projection (2.11) when the depth \( z \) is unknown. Therefore, the search for point correspondences can be constrained to the epipolar line, which is the projection of this ray onto a second view. The search is further constrained by selecting a minimum and maximum depth. All of this is visualized in figure 3.1 where the black arrow is the ray of possible locations for \( X_l \), \( X_{\text{min}} \) and \( X_{\text{max}} \) are the back projected search bounds, and the epipolar line segment to be searched is \( [x_{j,\text{min}}, x_{j,\text{max}}] \).

It is assumed that the 3D point has similar appearance in each image. The epipolar line can be searched at the resolution of the image with optional subpixel interpolation; however, subpixel accuracy is implicitly achieved by filtering multiple estimates in the final step. A single pixel is not enough information to obtain a unique match, so instead a local region of the image must be used.

The authors of [3] correlate line segments parallel to the epipolar lines in each image. When the transformation is small this is a good approximation of the projective mapping between frames; however, it is not a good approximation as the frames get further apart. The primary advantage of using a 1D correlation is speed. Additionally, it is simple enough to perform subpixel interpolation.
A more robust approach is to use a 2D correlation to find image correspondences. The similarity of two image patches $P_k \in \mathbb{R}^{h \times w}$ can be quantified using a normalized cross correlation (NCC)

$$\text{NCC score} = \frac{1}{wh} \sum_{v=0}^{h} \sum_{u=0}^{w} \frac{(P_1(u, v) - \mu_1)(P_2(u, v) - \mu_2)}{\sigma_1 \sigma_2} \in [-1, 1] \quad (3.1)$$

where $\mu_k$ is the mean and $\sigma_k$ is the standard deviation of patch $P_k$. Normalization is achieved by subtracting the mean and dividing by the standard deviation, which decreases sensitivity to changes in illumination. This is important when observing the scene from multiple viewpoints because the lighting may vary.

To further improve the quality of matches a full or affine homography $H_{k \rightarrow i} \in \mathbb{R}^{3 \times 3}$ can be applied to one of the image patches. The homography accounts for the projective mapping between frames; however, it is only valid for coplanar points. For small image patches it is acceptable to assume the scene is locally planar. When the transformation between frames is small the homography is simply a 2D translation, which is already determined from the point correspondence. For larger transformations the homography will warp the image patch; an example is scaling when the camera gets closer or further from a point.

### 3.1.2 Triangulation

Once the coordinate on the epipolar line with the highest correlation score has been found, the depth can be triangulated. Consider the triangle formed by $t_1 = 0$, $t_j$, and $X_i$ in figure 3.2. $X_i$ is unknown, but the rays $t_1 \rightarrow X_i$ and $t_j \rightarrow X_i$ are known from back projection (2.11). The angles $\theta_1$ and $\theta_2$ can be solved for using the following cross products

$$\| K^{-1} \mathbf{x}_{1,i} \times R_j K^{-1} \mathbf{x}_{j,i} \| = \| K^{-1} \mathbf{x}_{1,i} \| \| R_j K^{-1} \mathbf{x}_{j,i} \| \sin(\theta_1)$$

$$\| -t_j \times R_j K^{-1} \mathbf{x}_{j,i} \| = \| t_j \| \| R_j K^{-1} \mathbf{x}_{j,i} \| \sin(\theta_2) \quad (3.2)$$
Figure 3.2: Relevant quantities for the triangulation of $X_i$.

Applying the law of sines

$$\|X_i\|/\sin(\theta_2) = \|t_j\|/\sin(\theta_1)$$

(3.3)

Let $K^{-1}x_{1,i} = [wu \ v]$. Combining the law of sines (3.3) with the cross product definitions of $\sin(\theta_1)$ and $\sin(\theta_2)$ in (3.2) yields

$$z_{1,i} = w \frac{\|X_i\|}{\|K^{-1}x_{1,i}\|} = w \frac{\|t_j \times R_j K^{-1}x_{j,i}\|}{\|K^{-1}x_{1,i} \times R_j K^{-1}x_{j,i}\|}$$

(3.4)

Equation (3.4) only applies to coordinates on the epipolar line because $t_1 \rightarrow X_i$ and $t_j \rightarrow X_i$ must intersect. If this were not the case the analysis would need to be extended to finding where the rays are nearest or to optimizing $X_i$ such that the reprojection error is minimized.

### 3.1.3 Bayesian Estimation of Depth

The final step is to filter the depth estimates. The authors of [3] simply maintain a Gaussian distribution using a Kalman filter. The authors of [2] use the distribution presented in [4], which represents each measurement as the mixture of a Gaussian and uniform distribution. Assuming there are good measurements normally distributed around the true depth that occur with probability $\pi$ and outlier measurements uniformly distributed over $[z_{min}, z_{max}]$ that occur with probability $1 - \pi$, the pdf of depth $z$ can be formulated as follows

$$p(z|\mu, \pi) = \pi N(z|\mu, \sigma^2) + (1 - \pi)U(z|z_{min}, z_{max})$$

(3.5)

where $\mu$ and $\sigma$ are the mean and standard deviation of the normal distribution. $\mu$ can be considered the current depth measurement. Figures 3.3 and 3.4 show some example distributions. Note that the uniform distribution dominates if the inlier ratio is low or the variance is high, which is undesirable.
The goal is now to update the depth measurement \( \mu \) and inlier ratio \( \pi \) with each new estimate \( z_i \). Assuming each estimate is independent, the posterior can be formulated as follows

\[
p(\mu, \pi | z_1 \ldots z_N) \propto p(\mu, \pi) \prod_{i=1}^{N} p(z_i | \mu, \pi)
\]  

(3.6)

where \( p(\mu, \pi) \) is the initial guess, or prior, of the depth measurement and inlier ratio. The authors of [4] recommend a prior where the mean is half way between \( z_{\text{min}} \) and \( z_{\text{max}} \) and \( z_{\text{max}} - z_{\text{min}} \) is six standard deviations.

One approach to calculate Eqn. (3.6) would be to create a 2D histogram over all depths and inlier ratios. While this would allow multiple modes to be represented, storing a 2D histogram at every location in an image requires too much memory and computation. Instead the authors of [4] propose a unimodal parametric approximation for the posterior

\[
p(\mu, \pi | z_1 \ldots z_i) \cong \text{Beta}(\pi | a_i, b_i)N(\mu | \mu_0, \sigma_i^2)
\]  

(3.7)

where \( a_i \) and \( b_i \) are the beta distribution parameters. Given a new estimate \( z_{i+1} \) with variance \( \tau_{i+1}^2 \), the parameters of Eqn. (3.7) must be updated. The reader is referred to [4] for details on the update.

The variance of an estimate can be approximated as follows. Let \( x_{j,i} \) be the image coordinate with the highest correlation score. Starting from \( x_{j,i} \), take an additional step along the epipolar line to obtain \( x_{j,i}^+ \). Next, triangulate \( x_{j,i}^+ \) to obtain \( X_i^+ \), and finally take the Euclidean distance \( \|X_i - X_i^+\|_2 \) as one standard deviation. Note that small transformations yield imprecise depth.
estimates that should be reflected in the variance. If the transformation is small, then the denominator in equation (3.4) for triangulation approaches zero. Hence a small step in image coordinates will lead to a large jump in depth and the variance will be high, which reflects the imprecision as desired.

The posterior is updated with every new frame until it diverges or converges. Divergence occurs when the inlier ratio \( a_i/(a_i + b_i) \) is below some threshold \( \eta_{\text{outlier}} \), if this happens the measurement is reset to the prior and updating resumes as normal. Convergence occurs when the variance is small and the inlier ratio is above some threshold \( \eta_{\text{inlier}} \). After a measurement converges it is no longer updated. Ceasing updates not only alleviates computation, but also reduces the effect of occlusion. As the transformation between frames becomes larger there is a higher chance of occlusion and hence poorer estimates; therefore ceasing updates before there is too much movement is a better approach than updating indefinitely.

Estimates as described by Eqn. (3.7) are maintained for every pixel in the reference frame to generate a dense depth map. After every frame in the video has been incorporated, the final depth map is the set of all measurements that have converged. The discussion in this section is equally applicable to inverse depth.

### 3.2 Pose Estimation

As in subsection 3.1.1, it is reasonable to assume that a point \( X_i \in \mathbb{R}^3 \) projected onto two nearby frames will have similar appearance in each image. In this case each point is represented by its image coordinate \( x_{1,i} \in \mathbb{R}^2 \) and depth \( z_i \in \mathbb{R} \) in frame 1. If the depth is assumed to be known, then the goal is to minimize the intensity residual between the projected locations by updating the camera pose. Let \( \beta = [\omega^T \quad v^T]^T \in \text{se}(3) \) be the update parameters, which are applied with an exponential map (2.7)

\[
\exp(\beta) T = \begin{bmatrix} \exp(\omega) & Vv \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \exp(\omega) R & \exp(\omega) t + Vv \\ 0 & 1 \end{bmatrix}
\]  

(3.8)

From here on \( T \) will be used interchangeably with \( R \) and \( t \). The mapping from frame 1 to frame \( j \) in terms of inverse depth is given by equation (2.14), and applying the update in (3.8) yields the following warp function

\[
W(x_{1,i}, z_i; T, \beta) = \text{proj} \left( KR^T \exp(\omega)^T \left( K^{-1} \bar{x}_{1,i} - z_i^{-1} \exp(\omega) t + Vv \right) \right)
\]  

(3.9)

It is important to emphasize that (3.9) assumes frame 1 has no rotation or translation. If this is not the case, the rotation and translation of frame 1 can be removed prior to estimation and reapplied globally afterwards. The goal will now be to track the image coordinates \( x_{1,i} \) from frames to frame based on the allowable warp in (3.9). This is referred to as direct image alignment and will be used for both pose and joint estimation. For pose estimation there are two approaches: compositional and inverse compositional, which are described next.
### 3.2.1 Compositional Approach

In section 2.3 it was assumed that the parameter vector could be updated additively by $\beta^{(k+1)} = \beta^{(k)} + \delta$. However, $\text{se}(3)$ is a tangent space of $\text{SE}(3)$, and therefore this would only be a good approximation if $\beta^{(k)} - \beta^{(0)}$ remained very small. A more appropriate update is to apply the transformation in $\text{SE}(3)$ with the following composition

$$
\beta^{(k+1)} = \delta \circ \beta^{(k)} = \log(\exp(\delta) \exp(\beta^{(k)}))
$$

Applying bound constraints to the compositional update in (3.10) is non-trivial, but the unconstrained update (2.17) can still be used if the additive update is replaced by the compositional update. If the transformation between successive frames in a video is assumed to be small, then a valid initial estimate is simply $\beta^{(0)} = \log(T_{j-1})$. The logarithmic mapping can be avoided by instead updating $T_j$ and fixing $\beta$ at 0. The resulting residual and update rule are

$$
\begin{align*}
\epsilon_i &= I_1(x_{1,i}) - I_j\left(W(x_{1,i}, z_i; T_j^{(k)}, 0)\right) \\
T_j^{(k+1)} &= \exp(\delta) T_j^{(k)} \quad \text{with} \quad T_j^{(0)} = T_{j-1}
\end{align*}
\tag{3.11}
$$

This is referred to as the compositional approach [13]. Applying the chain rule to $\epsilon_i$ in (3.11), the resulting Jacobian is

$$
J_{\epsilon_i} = -\nabla I_j\left(W(x_{1,i}, z_i; T_j^{(k)}, 0)\right) T_j T\left(\frac{\partial W(x_{1,i}, z_i; T_j^{(k)}, \beta)}{\partial \beta} \right)_{\beta=0}
\tag{3.12}
$$

Notice that the Jacobian depends on $T_j^{(k)}$, which means it must be recomputed at every iteration. A more efficient approach is discussed next.

### 3.2.2 Inverse Compositional Approach

Rather than directly computing the update for $T_j$, an alternate approach is to calculate an update for $T_1$ and then apply the inverse of that update to $T_j$. Thus the Jacobian never changes allowing the Jacobian and Hessian matrices to be pre-computed. This is referred to as the inverse compositional approach [13]. The residual and inverse update rule are

$$
\begin{align*}
\epsilon_i &= I_1\left(W(x_{1,i}, z_i; I, \beta)\right) - I_j\left(W(x_{1,i}, z_i; T_j^{(k)}, 0)\right) \\
T_j^{(k+1)} &= \exp(\delta)^{-1} T_j^{(k)} \quad \text{with} \quad T_j^{(0)} = T_{j-1}
\end{align*}
\tag{3.13}
$$

where $\delta$ is calculated as in (2.17). The Jacobian is again obtained using the chain rule.
\[ J_{e,i} = \nabla I_1(x_{1,i})^T \left( \frac{\partial W(x_{1,i}, z_i; l, \beta)}{\partial \beta} \right) \bigg|_{\beta=0} \quad (3.14) \]

Notice that the Jacobian no longer depends on \( T_j^{(k)} \) and therefore does not change. The final detail is how to compute \( dW/d\beta \) in (3.12) and (3.14).

### 3.2.3 Jacobian of the Warp Function

Using the small angle assumption (2.9), the warp function \( W \) (3.9) can be simplified. This is valid because \( \beta = 0 \) and the incremental updates are expected to be small. For convenience, the warped homogeneous coordinate will be denoted by \( \overline{y} \in \mathbb{R}^3 \)

\[ \overline{y} = KR^T \left( (l + [\omega]_x)^T K^{-1} \overline{x}_{1,i} - z_i^{-1}(t + (l + [\omega]_x)^T(l + [\omega]_x/2)v) \right) \]

\[ W(x_{1,i}, z_i; T, \beta) \approx \text{proj}(\overline{y}) \quad (3.15) \]

Rather than directly taking the derivative of the 2D coordinate, it is easiest to first consider the homogeneous coordinate \( \overline{y} \) and then apply the quotient rule. Let \( J_\omega \in \mathbb{R}^{3 \times 3} \) be the derivative of \( \overline{y} \) with respect to \( \omega \) and \( J_v \in \mathbb{R}^{3 \times 3} \) be the derivative of \( \overline{y} \) with respect to \( v \)

\[ J_\omega = \frac{\partial \overline{y}}{\partial \omega} \bigg|_{\beta=0} = KR^T \left[ \frac{\partial [\omega]_x^T}{\partial r_x} K^{-1} \overline{x}_{1,i} \quad \frac{\partial [\omega]_x^T}{\partial r_y} K^{-1} \overline{x}_{1,i} \quad \frac{\partial [\omega]_x^T}{\partial r_z} K^{-1} \overline{x}_{1,i} \right] \]

\[ J_v = \frac{\partial \overline{y}}{\partial v} \bigg|_{\beta=0} = -z_i^{-1}KR^T \quad (3.16) \]

Applying the quotient rule

\[ \frac{\partial W(x_{1,i}, z_i; T, \beta)}{\partial \beta} \bigg|_{\beta=0} = \frac{1}{\overline{y}_3^2} \begin{bmatrix} \overline{y}_3 & 0 & -\overline{y}_1 \\ 0 & \overline{y}_3 & -\overline{y}_2 \end{bmatrix} [J_\omega \quad J_v] \quad (3.17) \]

As was mentioned in section 2.2, one reason to use inverse depth is that points at infinity remain useful for estimating rotation. It is clear from (3.16) that if \( z_i^{-1} = 0 \), then \( J_v = 0 \) but \( J_\omega \) does not change. When the Jacobian is zero it does not contribute to the update, thus points at infinity will not affect the translation but do contribute to the rotation. If the depth or 3D point were used directly, then points at infinity would cause \( J_\omega \) to become infinitely large rather than \( J_v \) going to zero.

### 3.2.4 Implementation Details

The first thing to notice is that the Jacobian matrices in (3.12) and (3.14) directly depend on the image gradient. Therefore, image regions that have little to no gradient should not be used for pose estimation. Intuitively this makes sense because the optimization is attempting to track each
image coordinate $x_{1,i}$ through the video, and therefore there needs to be enough image content to obtain a unique match. The content is quantified by image gradient.

The discussion so far has been for a single image coordinate. However, a single coordinate is not enough information to obtain a unique match. For this reason the coordinate is extended to a local region, or patch, of the image. Let $w \in \mathbb{Z}$ be the integer width of the patch and $h \in \mathbb{Z}$ be the integer height of the patch. Define the set $P_{j,i}$ of all coordinates in the patch centered at coordinate $x_{j,i}$ as

$$P_{j,i} = \left\{ x_{j,i} + \left[ k - \frac{w + 1}{2}, l - \frac{h + 1}{2} \right]^T \mid k \in [1,w], l \in [1,h] \right\}$$

(3.18)

If the scene is assumed to be locally planar, then the mapping of a patch in frame 1 to frame $j$ is defined by a homography. The warp function (3.9) is in fact a homography if the patch plane is additionally assumed to be parallel to the image plane (all points have the same depth). This is desirable because the projective mapping is implicitly accounted for. Figure 3.5 illustrates the relevant quantities for defining the homography in lemma 3.1 below.

The residual (3.13) and Jacobian (3.14) of the inverse compositional update then become

$$\varepsilon_i(\beta) = I_1 \left( W(P_{1,i}; z_{1,i}; l, \beta) \right) - I_j \left( W(P_{1,i}; z_{1,i}; T_j^{(k)}, 0) \right) \in \mathbb{R}^{hw}$$

(3.19)

$$J_{\varepsilon,i} = \nabla I_1(P_{1,i})^T \left( \frac{\partial W(P_{1,i}; z_{1,i}; l, \beta)}{\partial \beta} \right)_{\beta=0} \in \mathbb{R}^{hw \times 6}$$

The only difference is that the residual and Jacobian must be evaluated at more coordinates, which motivates smaller patch sizes.

Figure 3.5: Relevant quantities for determining the homography.
Lemma 3.1 The warp function (3.9) is the homography \( H_{1 \rightarrow j} \) of points on a plane parallel to the reference image plane at depth \( z_i \).

For convenience let \( R' = \exp(\omega) R \) and \( t' = \exp(\omega) t + Vv \). The warp function is then

\[
W(x_{1,i}, z_i; T, \beta) = \text{proj} \left( KR'T(K^{-1}x_{1,i} - z_i^{-1}t') \right)
\]  

(3.20)

If the plane being projected onto each image is parallel to the reference image plane and the surface normal \( n \in \mathbb{R}^3 \) is chosen to be facing away from the camera, then \( n = [0 \ 0 \ 1]^T \). The specifics of the intrinsic matrix are not important, but it is important to note that the third row is \([0 \ 0 \ 1]\) yielding the following

\[
n^TK^{-1}x_{1,i} = [0 \ 0 \ 1] \begin{bmatrix} K_{11}^{-1} & K_{12}^{-1} & K_{13}^{-1} \\ K_{21}^{-1} & K_{22}^{-1} & K_{23}^{-1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1,i} \\ 1 \end{bmatrix} = 1
\]  

(3.21)

Thus (3.20) can be written as

\[
W(x_{1,i}, z_i; T, \beta) = \text{proj}(KR'T(I - z_i^{-1}t'n)K^{-1}x_{1,i}) = \text{proj}(H_{1 \rightarrow j}x_{1,i})
\]  

(3.22)

To extend the “reach” of a coordinate without increasing the patch size, the image can be downsampled and pose estimation can be run on the downsampled image. The only change is that the coordinates and intrinsic matrix \( K \) must be appropriately scaled to match the new image size. Typically a pyramid of downsampled images is created and the pose is successively refined at each level starting from the smallest image.

Algorithm 1 in appendix A presents the inverse compositional Levenberg-Marquardt algorithm, which uses the residual and inverse update in (3.13).

### 3.3 Joint Estimation of Pose and Inverse Depth

To estimate pose in section 3.2 it was assumed that the depth was known. However, in order to estimate depth in section 3.1 it was assumed that pose was known. This interdependence makes it very difficult to initialize SLAM algorithms that require both pose and depth. The authors in [2] begin by bootstrapping: FAST [5] features are tracked for a few frames, an essential matrix is computed from the tracked correspondences, the essential matrix is decomposed into a rotation and translation, and an initial depth map is obtained via stereo matching. After initialization, pose and depth updates are interleaved with each new frame.

The initialization obtained by bootstrapping is typically poor because there is no guarantee that there was enough translation to accurately initialize the depth. In fact, if the camera motion was pure rotation then nothing can be said about the depth. Interestingly the authors in [3] were able...
to initialize the depth randomly and still achieve convergence. However, they observed that sufficient translation is needed in the first few seconds and that multiple keyframe propagations are required before convergence.

Jointly optimizing the pose and structure is referred to as bundle adjustment, and for the purpose of SLAM it eliminates the need for special initialization cases. Typically the explicit 3D points are optimized; however, here the bundle adjustment will be efficiently formulated in terms of inverse depth.

### 3.3.1 Partitioned Non-Linear Least Squares

The Levenberg-Marquardt algorithm will be used without bound constraints on the pose and with bound constraints on the depth. Let $N$ be the number of depth estimates. If (2.17) or (2.22) were used the normal equations would have $N + 6$ parameters with $N \sim 1e5$ for dense updates, and directly solving such a system of equations would be $\mathcal{O}(N^3)$. The complexity can be reduced to $\mathcal{O}(N)$ by exploiting the sparsity of the Hessian matrix in what is referred to as the partitioned or sparse Levenberg-Marquardt algorithm [14]. To start, the parameter and update vectors are partitioned as follows

$$
\beta = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \right| a = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathfrak{se}(3), b \in \mathbb{R}^N \right\} \quad \delta = \begin{bmatrix} \delta_a \\ \delta_b \end{bmatrix}
$$

(3.23)

where $a$ is the pose update and $b$ is the inverse depth. The new warp function is almost identical to (3.9) except the inverse depth is now a parameter

$$
W(x_{1,i}; T, \beta) = \text{proj} \left( KR^T \exp(\omega)^T \left( K^{-1} \bar{x}_{1,i} - b_l(\exp(\omega) t + Vv) \right) \right)
$$

(3.24)

As was discussed in section 3.2.1, the pose should be updated using a compositional approach. On the other hand, depth can be updated additively. The inverse compositional approach can no longer be used because if $R = I$ and $t = 0$, the Jacobian with respect to depth will be zero. Therefore, the compositional approach must be used. To simplify the notation define

$$
x_{j,i}^{(k)} = W(x_{1,i}; T_j^{(k)}, \beta^{(k)})
$$

(3.25)

Keep in mind that $a$ is fixed at zero. Recalling that $P(b)$ is the projection (2.18) of $b$ onto the bound constraints $0 \leq b \leq b_{max}$, the residual and update are then

$$
\varepsilon_i = I_1(x_{1,i}) - I_j(x_{j,i}^{(k)})
$$

$$
T_j^{(k+1)} = \exp(\delta_a) T_j^{(k)} \quad \text{with} \quad T_j^{(0)} = T_{j-1}
$$

$$
b^{(k+1)} = P(b^{(k)} + \delta_b) \quad \text{with} \quad b^{(0)} = z_1^{-1}
$$

(3.26)
The Jacobian matrices corresponding to \( a \) and \( b \) will be kept separate and can be obtained using the chain rule. It is also assumed that the inverse depth of coordinate \( x_{1,i} \) is independent of coordinate \( x_{1,l} \) for \( l \neq i \) yielding the following

\[
J_{a,i} = -\nabla l_j(x_{j,k}^{(k)})^T \left( \frac{\partial x_j^{(k)}}{\partial a} \right)_{a=0} \quad J_{b,i} = -\nabla l_j(x_{j,k}^{(k)})^T \left( \frac{\partial x_j^{(k)}}{\partial b_i} \right)_{a=0} \tag{3.27}
\]

From (3.27) it is not immediately obvious why the two derivatives have been kept separate until the entire Jacobian is constructed from each residual

\[
J_\epsilon = \begin{bmatrix} J_{a,1} & J_{b,1} & \cdots & J_{a,N} & J_{b,N} \\ J_{a,2} & J_{b,2} & \cdots \end{bmatrix} \tag{3.28}
\]

The sparse structure of the Jacobian is now apparent. However, to develop an efficient algorithm for calculating \( \delta \) the Hessian must be considered. In the remainder of this subsection the Hessian is introduced and the partitioned normal equations are derived. Recall from (2.17) that the Gauss-Newton approximation to the Hessian is \( J_\epsilon^T J_\epsilon \) and the gradient is \( J_\epsilon^T \epsilon \). Equation (3.28) results in the following sparse Hessian and gradient

\[
J_\epsilon^T J_\epsilon = \begin{bmatrix} A & C_1 & \cdots & C_N \\ C_1^T & B_1 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ C_N^T & B_N & \cdots & A \end{bmatrix} = \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \quad g = \begin{bmatrix} \sum_{i=1}^N J_{a,i}^T \epsilon_i \\ J_{b,1}^T \epsilon_1 \\ \vdots \\ J_{b,N}^T \epsilon_N \end{bmatrix} = \begin{bmatrix} g_a \\ g_{b,1} \\ \vdots \\ g_{b,N} \end{bmatrix} \tag{3.29}
\]

\[
A = \sum_{i=1}^N J_{a,i}^T J_{a,i} \quad B_i = J_{b,i}^T J_{b,i} \quad C_i = J_{a,i}^T J_{b,i}
\]

The regularized normal equations \( (H + \lambda \text{diag}(H)) \delta = -g \) can now be partitioned as follows

\[
(A + \lambda \text{diag}(A)) \delta_a + C \delta_b = A' \delta_a + C \delta_b = -g_a \tag{3.30}
\]

\[
C^T \delta_a + (B + \lambda \text{diag}(B)) \delta_b = C^T \delta_a + B' \delta_b = -g_b
\]

To solve for \( \delta_a \) the second equation can be multiplied by \( CB'^{-1} \) and subtracted from the first equation. Note that \( B' \) is block diagonal, which simplifies the matrix inverse.

\[
(A' - CB'^{-1}C^T) \delta_a = -g_a + CB'^{-1}g_b \tag{3.31}
\]

The solution for \( \delta_a \) is then used to solve for \( \delta_b \) in the second equation. \( B' \) is block diagonal, so \( \delta_b \) can be calculated element-wise

\[
\delta_{b,i} = -B_i'^{-1} (g_{b,i} + C_i^T \delta_a) \tag{3.32}
\]
Equations (3.31) and (3.32) are the general case for $b_i$ of any dimension. Unfortunately, applying the damping factor is nontrivial, which means that even if updates are not kept $B^{-1}$ and all relevant matrix products must be recalculated. Section 3.3.3 shows how the normal equations can be simplified even further for the special case $b_i \in \mathbb{R}$, which is why it is important to optimize inverse depth rather than 3D points. However, before discussing the specific implementation details $\partial W/\partial a$ and $\partial W/\partial b$ in (3.27) will be derived.

### 3.3.2 Jacobian of the Warp Function

The derivation here is nearly identical to section 3.2.3 with the addition of $\partial W/\partial b$. Using the small angle assumption (2.9), the warp function $W$ in (3.24) can be linearized with respect to $a$. This is valid because $a = 0$ and incremental updates are expected to be small. Denote the homogeneous coordinate by $\overline{y} \in \mathbb{R}^3$

$$\overline{y} = KR^T((I + [\omega]_x)T K^{-1}\overline{x}_{1,i} - b_i(t + (I + [\omega]_x)T(I + [\omega]_x/2)\nu))$$

(3.33)

$$W(x_{1,i}; T, \beta) \approx \text{proj}(\overline{y})$$

Rather than directly taking the derivative of the 2D coordinate, it is easiest to first consider the homogeneous coordinate $\overline{y}$ and then apply the quotient rule. Let $J_\omega \in \mathbb{R}^{3 \times 3}$ be the derivative of $\overline{y}$ with respect to $\omega$, $J_v \in \mathbb{R}^{3 \times 3}$ be the derivative of $\overline{y}$ with respect to $v$, and $J_{z,i} \in \mathbb{R}^{3 \times 1}$ be the derivative of $\overline{y}$ with respect to $b_i$. $J_\omega$ and $J_v$ are the same as in section 3.2.3, but are presented here for completeness.

$$J_\omega = \frac{\partial \overline{y}}{\partial \omega} \bigg|_{a=0} = KR^T \left[ \frac{\partial [\omega]_x^T}{\partial r_x} K^{-1}\overline{x}_{1,i} \quad \frac{\partial [\omega]_x^T}{\partial r_y} K^{-1}\overline{x}_{1,i} \quad \frac{\partial [\omega]_x^T}{\partial r_z} K^{-1}\overline{x}_{1,i} \right]$$

(3.34)

$$J_v = \frac{\partial \overline{y}}{\partial v} \bigg|_{a=0} = -b_i KR^T$$

$$J_{z,i} = \frac{\partial \overline{y}}{\partial b_i} \bigg|_{a=0} = -KR^T t$$

Applying the quotient rule

$$\frac{\partial W(x_{1,i}; T, \beta)}{\partial a} \bigg|_{a=0} = \frac{1}{\overline{y}_3^2} \begin{bmatrix} \overline{y}_3 & 0 & -\overline{y}_1 \end{bmatrix} \begin{bmatrix} J_\omega & J_v \end{bmatrix}$$

(3.35)

$$\frac{\partial W(x_{1,i}; T, \beta)}{\partial b_i} \bigg|_{a=0} = \frac{1}{\overline{y}_3^2} \begin{bmatrix} \overline{y}_3 & 0 & -\overline{y}_1 \end{bmatrix} \begin{bmatrix} J_{z,i} \end{bmatrix}$$

Interestingly, $\partial W/\partial b_i$ in (3.35) is a vector parallel to the epipolar line (see lemma 3.2 below). This implies that $J_{b,i}$ in (3.27) is simply the projection of the image gradient onto the epipolar
line, which has two implications: if the gradient is perpendicular to the epipolar line then the depth update will be zero, otherwise only the component of the gradient parallel to the epipolar line affects the depth update. Figure 3.6 shows the gradient directions that will increase or decrease the depth. Intuitively this makes sense because a change in depth can only shift the coordinate along the epipolar line.

![Figure 3.6: Relevant quantities for determining the epipolar line. The epipolar line is shown in blue and the gradient directions that will increase or decrease z are shown as green and red arrows respectively.](image)

**Lemma 3.2** The derivative $\partial W / \partial b_i$ in (3.35) is a vector parallel to the epipolar line.

The epipolar line in frame $j$ can be interpreted as the intersection of the image plane with the plane formed by $t_1 = 0$, $t_j$, and $X_i$ shown in figure 3.6. This is equivalent to finding the line that passes through the projection of $X_i$ onto frame $j$ and the projection of $t_1$ onto frame $j$, denoted by $y$ and $-K R_j^T t_j$ respectively. The projection of $t_1$ onto frame $j$ is also referred to as the epipole and follows directly from the camera projection (2.10).

The epipolar line $l \in \mathbb{R}^3$ can now be defined by the cross product of the projected coordinate and epipole. Notice that the epipole is equivalent to $J_{z,i}$ in (3.34), yielding

$$l = \bar{y} \times (-K R_j^T t_j) = \begin{bmatrix} 0 & -\bar{y}_3 & \bar{y}_2 \\ \bar{y}_3 & 0 & -\bar{y}_1 \\ -\bar{y}_2 & \bar{y}_1 & 0 \end{bmatrix} J_{z,i} \tag{3.36}$$

The epipolar line satisfies the relation $l_1 x + l_2 y + l_3 = 0$ where the vector $[l_1, l_2]^T$ is perpendicular to the line. To convert (3.36) to a parametric form, solve for a specific point $p$ by setting $y = 0$ and define the vector $v = [-l_2, l_1]^T$ parallel to the line

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = p + tv = \begin{bmatrix} -l_3/l_1 \\ 0 \end{bmatrix} + t \begin{bmatrix} -\bar{y}_3 & 0 & -\bar{y}_1 \\ 0 & -\bar{y}_1 & -\bar{y}_2 \end{bmatrix} J_{z,i} \tag{3.37}$$

Thus it is clear that $v \propto \partial W / \partial b_i$ and the two vectors are parallel. ■
3.3.3 Implementation Details

As was mentioned in section 3.2.1, the partitioned normal equations given by (3.31) and (3.32) can be simplified if \( b_i \) is scalar. It can be seen from (3.27) and (3.29) that in this case \( f_{b,i} \) and \( B_i \) will also be scalar. Thus \( B \) is a diagonal matrix and \( B' = B + \lambda \text{diag}(B) = (1 + \lambda)B \). Equation (3.31) becomes

\[
\left( A' - \frac{1}{1 + \lambda} CB^{-1}C^T \right) \delta_a = -g_a + \frac{1}{1 + \lambda} CB^{-1} g_b
\]  

(3.38)

Equation (3.32) becomes

\[
\delta_{b,i} = -\frac{1}{(1 + \lambda)B_i} \left( g_{b,i} + C_i^T \delta_a \right)
\]

(3.39)

There are two distinct advantages when \( b_i \) is scalar: the inverse of \( B \) is trivial because it is a diagonal matrix and the damping factor is easy to apply. As with pose estimation, joint estimation should only be performed where there is image gradient and should be extended to use image patches rather than single coordinates. All of the derivation above is applicable to image patches; one simply needs to replace all instances of \( x_{1,i} \) with \( P_{1,i} \) (3.18). See section 3.2.4 for discussion.

Unlike pose estimation, it is not recommended to perform joint optimization on downsampled images because sampling the depth is nontrivial. Instead, the downsampled images are used for pose estimation after which the original image is used for joint optimization. Estimating pose on the downsampled images allows the optimization to handle larger transformations and improves the initial estimate during joint optimization. In fact, the update to the mapped coordinate \( x_{j,i} \) during joint optimization is often subpixel.

The gradient projection method is easily applied to \( b \) by setting \( B_i = 1, C_i = 0, \) and \( g_{b,i} = 0 \) for all \( i \in A(b) \) defined by equation (2.19). However, gradient projection is not used because the joint estimation tends to converge in too few iterations. Instead, the active set is reformulated to contain depth gradients that are near zero, which cause the Hessian to become singular

\[
A(b) = \{ i \mid B_i < \epsilon \}
\]

(3.40)

As a final note, \( ||\delta_b|| \) should not be used to check for convergence of the Levenberg-Marquardt algorithm. Depending on the transformation between frames there is potentially a large range of depths that project to the same image coordinate, so \( \delta_b \) may never go to zero and instead fluctuate within the valid range.

Algorithm 2 in appendix B presents the “1D” partitioned Levenberg-Marquardt algorithm, which uses the update and residual in (3.26).
Chapter 4

Results

This chapter will present the results of the proposed method on three datasets: the simulated over table sequence [8], the simulated fast motion sequence [8], and the TUM RGB-D benchmark freiburg2 xyz [9]. Section 4.1 presents the jointly estimated depth map at the highest density and back projected point cloud for all three datasets. In each case 60 frames were used. Section 4.2 is a quantitative analysis of the reconstruction accuracy, pose accuracy, and runtime. Additionally, the jointly estimated depth is compared to the stereo matching algorithm of [2].

4.1 Sample Reconstructions

In all cases direct bundle adjustment was run on intensity images with values from 0 – 255 using a gradient threshold of 10, 5 x 5 image patches (3.18), three pyramid levels (see section 3.2.4), and maximum inverse depth $b_{max} = 20$. As in [3] the depth was initialized randomly. Points that go out of bounds are simply ignored because tracking must be continuous. In order to perform sparser reconstructions, the image is divided into a grid and only the point with the largest gradient magnitude in each grid cell is kept. The density will thus be quantified by the grid cell size; the reconstructions in this section are cell size 1 (every point with enough gradient is optimized).

Figures 4.1 and 4.2 show the groundtruth depth and reference frame of the over table and fast motion datasets [8]. Figure 4.3 shows the Microsoft Kinect depth and reference frame of the freiburg2 xyz dataset [9]. The over table dataset is relatively simple and serves as a good benchmark. The fast motion dataset is characterized by rapid side-to-side motion of the camera and demonstrates the proposed method’s ability to handle large transformations. The freiburg2 xyz dataset is included as an instance of real data collected using the Microsoft Kinect. Groundtruth pose is also available for each of these datasets, which will be discussed in the next section.

Figures 4.4 to 4.6 show the reconstructions from 60 frames. Each reconstruction suffers from erroneous points at object edges due to occlusions. For example, the edge of the desk in the over table dataset: as the desk occludes the floor, points on the floor are “pushed” by the desk edge leading to erroneous depth estimates. Joint estimation was able to reconstruct objects beyond the range of the Microsoft Kinect, which can be seen from the tripod in the background of figure 4.3.
Figure 4.1: Groundtruth depth (left) and reference frame (right) of the over table dataset.

Figure 4.2: Groundtruth depth (left) and reference frame (right) of the fast motion dataset.

Figure 4.3: Microsoft Kinect depth (left) and reference frame (right) of the freiburg2 xyz benchmark.
Figure 4.4: Estimated depth after 60 frames (left) and back projected cloud (right) of the over table dataset.

Figure 4.5: Estimated depth after 60 frames (left) and back projected cloud (right) of the fast motion dataset.

Figure 4.6: Estimated depth after 60 frames (left) and back projected cloud (right) of the freiburg2 xyz dataset.
4.2 Experimental Results

In this section the reconstruction accuracy will be analyzed, followed by analysis of the pose accuracy and runtime. To quantitatively evaluate a reconstruction, the authors of [2] propose to look at the precision and completeness

\[
\text{precision}(e) = \frac{\# \text{ of converged points within } e \text{ of groundtruth}}{\# \text{ of converged points}} \quad (4.1)
\]

\[
\text{completeness}(e) = \frac{\# \text{ of converged points within } e \text{ of groundtruth}}{\text{image size}} \quad (4.2)
\]

The precision measures how accurately the depth has converged, whereas the completeness measures how much of the image has been successfully reconstructed. However, because this is a monocular reconstruction the absolute scale must be resolved before comparing the estimated depth to the groundtruth. Two methods were compared for recovering scale: absolute orientation [15] of the backprojected points and scaling the baseline. The best results were obtained by scaling the baseline as follows

\[
s_j = \frac{t_{j, \text{true}} - t_{1, \text{true}}}{t_{j, \text{est}} - t_{1, \text{est}}} \quad (4.3)
\]

\[
z_{1,j}^{(i)} = s_j z_{1,i}^{(j)} \quad t_j = s_j (t_j - t_1) + t_1
\]

where \(t_{j, \text{true}}\) and \(t_{j, \text{est}}\) are the true and estimated camera centers at frame \(j\) and \(z_{1,j}^{(i)}\) is the depth estimate of point \(i\) after frame \(j\) has been optimized. Based on the formulation in section 2.2, the camera center of frame \(j\) is simply the translation. From here on, whenever pose or depth is compared to groundtruth it is first scaled by \(s_j\) as in (4.3).

Figures 4.7 and 4.8 plot the precision and completeness with an error \(e\) of 0-10cm for the over table and fast motion dataset respectively. Six reconstructions were run for 60 frames with cell sizes of 1, 2, 4, 8, 16, and 32; however, figures 4.7 and 4.8 list the total number of points rather than the cell size. For reference, the precision of REMODE [2] using pose from the joint optimization is compared to the precision of cell size 1. REMODE yields a better estimate in both cases; however, the improvement for the fast motion dataset is marginal. This additionally demonstrates that joint optimization can be used to estimate the pose necessary for stereo matching.

It can be seen that the precision of the reconstruction does not depend on how many points were optimized. This is of paramount importance because it implies that a dense reconstruction is not necessary to recover pose, which will be discussed in detail later on when the pose and runtime are presented. On the other hand, it is clear that the completeness is directly related to the number of points. The fast motion dataset has particularly low completeness because a large portion of the points went out of bounds, which is also apparent from the depth in figure 4.5.
Figure 4.7: Precision (top) and completeness (middle) of the estimated depth after 60 frames for the over table dataset. (bottom) The precision of REMODE [2] run using the joint pose is compared to the precision of the densest reconstruction.
Figure 4.8: Precision (top) and completeness (bottom) of the estimated depth after 60 frames for the fast motion dataset. (bottom) The precision of REMODE [2] run using the joint pose is compared to the precision of the densest reconstruction.
One final point of interest for the reconstruction accuracy is how fast the depth converges to a good estimate. Figure 4.9 shows the precision at frames 10, 20, 30, 40, 50, and 60 of the over table dataset with cell size 1. By frame 30 the depth has converged to a good estimate, which indicates that very little motion is needed to determine the depth. Beyond frame 30 the depth continues to improve, but with diminishing return.

Figure 4.9: Precision of the estimated depth from 10 to 60 frames for the over table dataset with cell size 1 (166055 points).

To evaluate pose, the absolute trajectory error [9] is considered. This is the average Euclidean distance between the true and estimated camera center after alignment via absolute orientation [15]. Table 4.1 shows the error for each dataset with cell size 1, 2, 4, 8, 16, and 32. The best error is bolded. However, the error over all 60 frames is somewhat misleading because initially the depth has not converged and the pose is very inaccurate. For this reason, table 4.1 presents the error for frames 30-60 as well as 1-60. It can be seen that the pose is significantly more accurate once the depth has converged.

This is further illustrated in figure 4.10, which shows the true and estimated trajectory for the fast motion dataset with cell size 1 and 32. Initially the pose deviates drastically from the groundtruth, but soon converges to the correct trajectory. This occurs regardless of how many points are being optimized and was most apparent in the fast motion dataset; the error in table 4.1 drops from a few centimeters to 1-2mm.

Table 4.1 also reveals a slight tendency for denser reconstructions to have more accurate pose; however, the sparse reconstructions still yield a good estimate. Recall that the depth precision was also independent of reconstruction density (figures 4.7 and 4.8). Thus, it is clear that joint estimation is equally applicable to sparse and dense sets of points. This is important because it means pose estimation only requires a sparse set of points that can be optimized in real-time, which will be seen next in the discussion of runtime.
<table>
<thead>
<tr>
<th>cell size</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frames</td>
<td>over table</td>
<td>fast motion</td>
<td>fr2/xyz</td>
<td>over table</td>
<td>fast motion</td>
<td>fr2/xyz</td>
</tr>
<tr>
<td>1-60</td>
<td>0.0109</td>
<td>1.8966</td>
<td>0.2200</td>
<td>0.0076</td>
<td>0.1124</td>
<td>0.1148</td>
</tr>
<tr>
<td></td>
<td>0.0129</td>
<td>5.1742</td>
<td>0.2161</td>
<td>0.0080</td>
<td>0.1127</td>
<td>0.1127</td>
</tr>
<tr>
<td></td>
<td>0.0160</td>
<td>5.1660</td>
<td>0.2045</td>
<td>0.0054</td>
<td>0.1152</td>
<td>0.1099</td>
</tr>
<tr>
<td></td>
<td>0.0130</td>
<td>4.4714</td>
<td>0.2195</td>
<td>0.0070</td>
<td>0.1561</td>
<td>0.1643</td>
</tr>
<tr>
<td></td>
<td>0.0264</td>
<td>1.9600</td>
<td>0.8584</td>
<td>0.1738</td>
<td>0.4277</td>
<td>0.4277</td>
</tr>
<tr>
<td></td>
<td>0.1532</td>
<td>6.0928</td>
<td>1.5088</td>
<td>0.2388</td>
<td>0.3170</td>
<td>0.3170</td>
</tr>
</tbody>
</table>

Table 4.1: Absolute pose error (cm) over 60 frames with 3 levels, patch size 5, and gradient threshold 10.

Figure 4.10: True and estimated trajectory for 60 frames of the fast motion dataset with cell size 1 (top) and cell size 32 (bottom).
To evaluate runtime, the average time per frame with cell size 1, 2, 4, 8, 16, and 32 for the over table dataset is plotted in figure 4.11. A log-log scale is used to see the results more clearly and a line with equation $y = mx$ has been fit to the data points. 33ms (the frame rate of the video) is shown as a red line below which is real-time performance. Clearly the runtime is linear with respect to the number of points as claimed in section 3.3.1. Additionally, up to 1600 points could be optimized in real-time using a 2.4GHz Intel Core i7-3630QM processor.

Figure 4.11: Average runtime of joint estimation using a 2.4GHz Intel Core i7-3630QM verse the number of points for the over table dataset. The red dashed line is 33ms, below which is real-time.
Chapter 5

Conclusion

This thesis has introduced direct bundle adjustment, which densely reconstructs local regions of video by jointly optimizing camera pose and scene depth via direct image alignment. Joint optimization is desirable because it eliminates the need for special initialization cases or bootstrapping; however, bundle adjustment is rarely used for dense reconstructions due to the size of the optimization problem. To achieve reasonable runtimes, sparsity was exploited and the reconstruction was minimally represented as an inverse depth map rather than explicit 3D points. Even with these optimizations, dense reconstructions are not real-time; however, sparse sets of points can be used for real-time camera tracking that is robust to poor initialization cases. It would be interesting to additionally estimate the intrinsic matrix, which would eliminate the need for prior calibration.

Future work will be to create a real-time algorithm that uses direct bundle adjustment for pose estimation and stereo matching for dense reconstruction. This is favorable to GPU acceleration of direct bundle adjustment for two reasons: stereo matching does not need to be limited to regions of high gradient and stereo estimates can be filtered as discussed in section 3.1.3. Stereo estimates with low gradient that cannot be uniquely matched will have high variance and can be ignored, whereas joint optimization requires every point to be trackable. Bayesian estimation eliminates poor estimates (such as occlusions) via variance and inlier ratio, but should not be used with joint estimation because modifying the parameters outside of the optimization has an unpredictable effect on the tracks of each point. In conclusion, joint estimation yields robust and convenient pose estimates; however, stereo matching is superior for dense reconstruction.
Appendix A

Inverse Compositional Pose Estimation Algorithm

**Given:** A set of image coordinates and their depth \( \{x_{1,i}, z_{1,i} : \nabla I_1(x_{1,i}) \geq \tau \} \) with gradient above some threshold \( \tau \) and the pose of the previous frame \( T_{j-1} \in \text{SE}(3) \).

**Objective:** Minimize \( E(T_j) = \varepsilon^T \varepsilon \) with \( T_j \in \text{SE}(3), \varepsilon \in \mathbb{R}^n \), and

\[
\varepsilon_i = I_1 \left( W(x_{1,i}, z_{1,i}; I, 0, \beta) \right) - I_j \left( W(x_{1,i}, z_{1,i}; R_j^{(k)}, t_j^{(k)}, 0) \right)
\]

**Algorithm:** Define the parameter vector \( \delta = [\omega^T \ v^T]^T \) where \( \delta \in \text{se}(3) \) is the Lie algebra (2.6) of the pose update.

(i) Initialize the pose \( T_j = T_{j-1} \) and damping factor \( \lambda = .001 \).

(ii) Compute \( \varepsilon \) and \( J = \partial \varepsilon / \partial \beta \) using (3.14) and (3.17).

(iii) Solve the regularized normal equations for \( \delta \)

\[
(J^T J + \lambda \text{diag}(J^T J)) \delta = -J^T \varepsilon
\]

(iv) Compute the new error \( E(\exp(\delta)^{-1} T_j) \).

(v) If the error decreased: \( T_j = \exp(\delta)^{-1} T_j \). If \( ||\delta||_\infty \) is small enough, stop. Otherwise, \( \lambda = \lambda / 10 \) and go to step (ii).

(vi) If the error increased: If \( ||\delta||_\infty \) is small enough, stop. Otherwise, \( \lambda = 10\lambda \) and go to step (iii).

Algorithm 1: Inverse compositional pose estimation.
Appendix B

Joint Estimation Algorithm

Given: A set of image coordinates \( \{ x_{1,i} : \nabla l_1(x_{1,i}) \geq \tau \} \) with gradient above some threshold \( \tau \), the pose of the previous frame \( T_{j-1} \in SE(3) \), and the current inverse depth estimates \( b = [ z_{1,1}^{-1}, z_{1,2}^{-1}, \ldots, z_{1,n}^{-1}]^T \).

Objective: Minimize \( E(T_j, b) = e^T \epsilon \) with \( T_j \in SE(3), \epsilon \in \mathbb{R}^n \), and \( \epsilon_i \) defined in (3.26)

Algorithm: Define the parameter vectors \( \delta_a = [\omega^T, v^T]^T \) and \( \delta_b \in \mathbb{R}^n \) where \( \delta_a \in se(3) \) is the Lie algebra (2.6) of the pose update and \( \delta_b \) is the inverse depth update.

(i) Initialize the pose \( T_j = T_{j-1} \) and damping factor \( \lambda = .001 \).

(ii) Compute \( e, J_{a,i} = \partial \epsilon_i / \partial a \), and \( J_{b,i} = \partial \epsilon_i / \partial b_i \) using (3.27) and (3.35). Compute the following intermediate values

\[
A = \sum_{i=1}^{N} J_{a,i}^T J_{a,i}, \quad g_a = \sum_{i=1}^{N} J_{a,i}^T \epsilon_i
\]

\[
B_i = \begin{cases} 
J_{b,i}^T J_{b,i}, & \text{if } J_{b,i}^T J_{b,i} \geq \epsilon \\
1, & \text{else}
\end{cases}, \quad C_i = \begin{cases} 
J_{a,i}^T J_{b,i}, & \text{if } J_{a,i}^T J_{b,i} \geq \epsilon \\
0, & \text{else}
\end{cases}, \quad g_b,i = \begin{cases} 
J_{b,i}^T \epsilon_i, & \text{if } J_{b,i}^T J_{b,i} \geq \epsilon \\
0, & \text{else}
\end{cases}
\]

\[
S_1 = \sum_{i=1}^{N} C_i C_i^T / B_i, \quad S_2 = \sum_{i=1}^{N} C_i \epsilon_i b_i / B_i
\]

(iii) Solve the regularized normal equations for \( \delta_a \)

\[
(A + \lambda \text{diag}(A) - S_1 / (1 + \lambda))\delta_a = g_a - S_2 / (1 + \lambda)
\]

(iv) Use \( \delta_a \) to solve for \( \delta_b \)

\[
\delta_{b,i} = (g_{b,i} - C_i^T \delta_a) / ((1 + \lambda)B_i)
\]

(v) Compute the new error \( E(\exp(\delta_a) T_j, \text{mid}\{0, b + \delta_b, b_{max}\}) \).

(vi) If the error decreased: \( T_j = \exp(\delta_a) T_j \) and \( b = \text{mid}\{0, b + \delta_b, b_{max}\} \). If \( \| \delta_a \|_\infty \) is small enough, stop. Otherwise, \( \lambda = \lambda / 10 \) and go to step (ii).

(vii) If the error increased: If \( \| \delta_a \|_\infty \) is small enough, stop. Otherwise, \( \lambda = 10 \lambda \) and go to step (iii).

Algorithm 2: Joint estimation of pose and inverse depth.
References


