High accuracy stereo Structure from Motion through symmetry cost functions

Using cost functions with symmetry terms for high accuracy relative motion estimation

PETER SÖDERGREN

psoderg@kth.se
Master’s Thesis at CSC
Computer Science
Supervisor: Stefan Carlsson
Examiner: Stefan Carlsson
14 April 2014

Svensk titel: Nogrann estimering av relativ stereokamerarörelse genom symmetri i kostnadsfunktioner

TRITA xxx yyyy-nn
Abstract

A large effort, in the field of Computer Vision in general and Structure from Motion (SfM) in particular, has previously been put into developing algorithms for estimating camera pose and triangulate 3D features. Many of these algorithms however are based upon maximum posterior likelihood cost functions which are efficient in terms of time consumption rather than in accuracy. The implementations are thus based on a simplified model in order to make the algorithm more efficient. This project aimed to look at a particular problem, stereo cameras with approximately known relative motion, and the goal was to develop a more accurate cost function. The algorithms presented uses the symmetry of the measured noise in order to produce an estimated relative motion which outperforms the least square ($L_2$) method with outlier rejection. The project did include both synthetic and real test data sets. The project was done at the request of a company and the testing sets were based on typical properties of a projectile tracking system used by the company.
Sammanfattning

Noggrann estimering av relativ stereokamerarörelse genom symmetri i kostnadsfunktioner

Ett vanligt forskningsområde inom Structure from Motion (SfM) är att estimerar den relativa rörelsen mellan kameror. Många av dessa algoritmer förenklar den använda modellen för att sänka tidskonsumtionen av processen istället för att öka precisionen. Detta projekt hade som mål att undersöka ett specifikt problem, stereo kameror med relativ rörelse där den relativa rörelsen var approximativt känd, samt utveckla en mer realistisk kostnadsfunktion för utvärdering ur ett efterhand maximal sannolikhetsperspektiv. Algoritmen som presenteras i rapporten använder symmetrin i bruset för att producera en estimering av den relativa rörelsen som överträffar minsta kvadratmetoder ($L_2$) med förkastning av felaktigt ihopkopplade bildpunkter. Projektet inkludera både verkliga såväl som syntetiska dataset. Projektet utfördes på begäran av ett företag och de syntetiska dataseten var baserade på typiska egenskaper i företagets projekttilspårningssystem.
Contents

1 Introduction 1

2 Related Works 2

3 Background Theory 3
  3.1 The Pin-Hole Camera Model 3
  3.1.1 The Camera Matrix 3
  3.1.2 The Rotation Matrix 3
  3.1.3 Normalized Rotation Vector and Rotation Angle 4
  3.1.4 Lens Distortion 5
  3.2 The Essential Matrix 5
  3.3 The Fundamental Matrix 6
  3.4 Minimization Methods 6
    3.4.1 Gradient Descent 7
    3.4.2 Newton’s Method 8
    3.4.3 Levenberg-Marquardt 8
    3.4.4 Branch and Bound 9
  3.5 Standard Cost Functions 10
    3.5.1 $L_1$ Cost 10
    3.5.2 $L_2$ Cost 10
    3.5.3 $L_\infty$ Cost 10
  3.6 Triangulation 11
    3.6.1 Linear-Eigen Triangulation 11
    3.6.2 $L_2$ Triangulation 11
    3.6.3 $L_\infty$ Triangulation 13
  3.7 Structure from Motion (SfM) 13
    3.7.1 The SVD-Method 14
    3.7.2 Local Iterative Methods 15
    3.7.3 Bundle Adjustment 16
    3.7.4 $L_2$ Minimization 16
    3.7.5 $L_\infty$ Minimization 17
    3.7.6 Object Space Error Minimization 18
## 4 Method

### 4.1 Minimization

- **4.1.1 Minimization Algorithm**
- **4.1.2 Initialization**
- **4.1.3 Triangulation Algorithm**

### 4.2 Synthetic Datasets

- **4.2.1 Gaussian Error Distribution**
- **4.2.2 Rayleigh Error Distribution**
- **4.2.3 Uniform Error Distribution**
- **4.2.4 Independent Uniform Error Distribution**
- **4.2.5 Outliers**

### 4.3 Real Datasets

### 4.4 Cost Function Objectives

- **4.4.1 Outlier Insusceptibility**
- **4.4.2 Error Representation**
- **4.4.3 Pseudo Global Convergence with Good Initialization**

### 4.5 Cost Functions

- **4.5.1 \( L_2 \) with Outlier Tolerance (\( L_2\)-WOT)**
- **4.5.2 \( L_2 \) with Outlier Rejection (\( L_2\)-WOR)**
- **4.5.3 \( L_2 \) Difference**
- **4.5.4 Variance Difference**
- **4.5.5 \( \beta \) Values of the Symmetry Functions**

### 4.6 Measurements

## 5 Results

### 5.1 Synthetic Data

- **5.1.1 Basic Functions - Noisy Reprojection Error Measurement**
- **5.1.2 Basic Functions - Real Projection Error Measurement**
- **5.1.3 Performance of \( L_2\)-WOT**
- **5.1.4 Difference Functions with SVD Initialization**
- **5.1.5 Difference Functions with \( L_2\)-WOR Initialization**
- **5.1.6 Convergence**
- **5.1.7 Error Representation**
- **5.1.8 Iterations**

### 5.2 Real Data

- **5.2.1 Noisy Reprojection Error**
- **5.2.2 Solution Spread**
- **5.2.3 Solution Difference**
- **5.2.4 Comparison with the company’s old method**

### 5.3 Symmetry Tests

## 6 Discussion

### 6.1 Limitations of the Test

### 6.2 Limitations of the Scope
Chapter 1

Introduction

The field of Structure from Motion (SfM) in Computer Vision has for a long time been focused on developing algorithms which increase accuracy when estimating relative motion from point correspondences. Initially and for a long time this was done by using the 8-point-algorithm which is based around estimating the essential and the fundamental matrix. [1] Today there are three different groups of relative motion estimation methods, SVD-methods, local iterative methods, and global iterative methods.

SVD-methods as explained previously approximates the fundamental or essential matrix and extracts the relative motion from those matrices. SVD-methods requires point correspondences with low noise. To achieve good results when noise is present it is common to run the method several times, each time labelling a random set of point correspondences as outliers. Local iterative methods finds the local minima to a specified cost function. Local iterative methods are typically good for finding a single minimum but are as the name suggest susceptible to local minima. Global iterative methods finds the global minima of the specified cost function. These methods produces the best results for a given cost function but are difficult if not impossible to create for most cost functions.

This project was focused on finding new cost functions which approximate the relative motion of stereo cameras better than the standard $L_2$ cost function, using local iterative methods. The project was done at a company which wanted to investigate how they could improve the relative motion estimation in their projectile tracking system. The system was based upon two stereo cameras with known camera intrinsics and approximately known relative motion. The point correspondences were produced by the projectiles tracked in the system and thus known to be low in outliers.
Chapter 2

Related Works

Using local iterative methods in order to improve motion estimation is quite common in many SfM implementations. The most common implementation of this group of methods is bundle adjustment. Bundle adjustment is a local iterative method where the solution space does not only include the relative motion of the cameras but also the elements of each 3D-coordinate. Bundle adjustment does have a long history in the field of photogrammetry but has only relatively recently been introduced in computer vision. [2] Currently there are complete and working packages for bundle adjustment, utilizing Levenberg-Marquardt. [3] [4] These methods implement least square methods due to speed optimization possibilities associated with least square methods. The possibility for high speed has made bundle adjustment viable for real time scenarios and very large SfM problems. [5] [6]

Bundle adjustment and the local iterative method used in this project do have much in common but the local iterative method used in this project did only include the relative motion in the solution space. The 3D-points was not included in the solution space but instead triangulated during each iteration. The choice to use the smaller solution space was due to Gauss-Newton optimizations, only being available for $L_2$ rather than general cost functions. The accuracy experienced during motion estimation does also become more representative of the accuracy during later real time triangulation when triangulating is performed at each iteration. The triangulations were however the source of most of the time consumption during the executions.
Chapter 3

Background Theory

3.1 The Pin-Hole Camera Model

The pin hole camera model is the underlying model in much of computer vision. However there are a few variations of the model and thus a clarification is needed, in order for the reader to follow the reasoning of this report more easily. Homogeneous coordinates are used throughout the thesis due to their numerical stability near the infinity.

\[ u \in \mathbb{P}^2 \]
\[ X \in \mathbb{P}^3 \]
\[ u = K [ R \ T ] X \]

3.1.1 The Camera Matrix

The intrinsic camera matrix is a 3 by 3 matrix which consists of five variables, the two scaled versions of the focal length \( f_x \) and \( f_y \), the skew parameter \( \gamma \) and the two camera centre coordinates \( c_x \) and \( c_y \).

\[
K = \begin{bmatrix}
  f_x & \gamma & c_x \\
  0 & f_y & c_y \\
  0 & 0 & 1
\end{bmatrix}
\]

The camera matrix intrinsics can be calibrated in several different ways. [7] [8] This thesis does not focus on camera calibration and thus assumes that any instance of \( K \) is always known with reasonable accuracy.

3.1.2 The Rotation Matrix

The rotation matrix is a 3 by 3 matrix which describes the rotation of the world relative to the camera. The rotation matrix is orthogonal which means that it is
column and row wise orthonormal. This is a good property of the rotation matrix since its inverse is simply its transpose:

$$R^{-1} = R^T$$

The rotation imposed by rotation matrix is not easily comprehensible and thus the rotation matrix in this report is often presented as euler angles. The euler angles are yaw, describing the rotation around the z-axis, pitch describing the rotation around the y-axis, and roll describing the rotation around the x-axis. The rotation matrices for the presented sub rotations are presented below.

$$R_{\text{yaw}}(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

$$R_{\text{pitch}}(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix}$$

$$R_{\text{roll}}(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_{\text{roll}}(\gamma)R_{\text{pitch}}(\beta)R_{\text{yaw}}(\alpha)$$

Using euler angles can sometimes lead to problems with singularities e.g. when the pitch aligns the roll and yaw rotation. This was during this project not of any concern since any rotation angle were never be close to $90^\circ$ in any direction. The choice to use euler angles was made since it was used internally at the company at which the project was carried out. In order to extend the results of this thesis to include all possible angles without any singularities one can use another rotation system instead e.g. quaternions or a normalized rotation vector and a rotation angle. The normalized rotation vector and a rotation angle is covered in section 3.1.3.

### 3.1.3 Normalized Rotation Vector and Rotation Angle

The normalized rotation vector and rotation angle is covered by this report since it was used when comparing the difference between rotation results. This rotation notation consists of two parts, the rotation vector and the rotation angle. The rotation vector is used to define the axis around which the rotation is taking place and the rotation angle defines the amount of rotation. The rotation vector is normalized in order to have a reference when comparing different rotation vectors in
terms of distance. It should be noted that there is an alternative notation in which
the length of the rotation vector indicates the angle of the rotation. In this project
the following notation was used.

\[ \omega \text{ } \]

### 3.1.4 Lens Distortion

One part of camera calibration is to estimate lens distortion. This distortion is
usually modelled as an exponential function originating at the principal point of
the camera. A simple model for calibrated camera coordinates where \(x,y\) are the
undistorted coordinates and \(\hat{x}, \hat{y}\) are the distorted coordinates is shown below. [7]

\[
\begin{align*}
    r &= x^2 + y^2 \\
    \hat{x} &= x + x(k_1r + k_2r^2) \\
    \hat{y} &= y + y(k_1r + k_2r^2)
\end{align*}
\]

This is a very simple model. More advanced models include fourth degree distortion
as well, however the best models are usually based on real metric measurements. [9]
In this report the coordinates were assumed to not be distorted since this was not
the focus of the project.

### 3.2 The Essential Matrix

The essential matrix is a 3 by 3 matrix that forms the basis of epipolar geometry.
The essential matrix relates a point in one camera to a line in the other camera.
If we take two homogeneous calibrated camera coordinates \(u\) and \(v\) directed at the
same object in 3D-space, the essential matrix provides the following equality.

\[
u, v \in \mathbb{P}^2
\]

\[
v^TEu = 0
\]

The essential matrix can be explained as the the matrix which describes the inverse
of one pin hole camera equation followed by the application of the other. First state
the following pin hole camera equations and their epipoles, \(e\).

\[
\begin{align*}
u &= \begin{bmatrix} I & 0 \end{bmatrix} X \\
v &= \begin{bmatrix} R & T \end{bmatrix} X \\
e &= -RT^T \\
e' &= \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = T
\end{align*}
\]

The essential matrix can also be described as the line joining the epipole and back
projection of the 3D feature \(X\) in camera 2. The position of feature \(X\) can thus
from one point in the first camera be restricted to the line originating from camera centre 1 and following the path of u in \( \mathbb{R}^3 \). The cross product of the projected 3D feature in camera 2 and the epipole of camera 1 projected in camera 2 defines this line. By extracting \( u \), the essential matrix can be specified. The essential matrix still has an unknown scale factor but since a line equation holds true regardless of scale it can be disregarded.

\[
X = \begin{bmatrix} \lambda u \\ 1 \end{bmatrix}
\]

\[
v = \lambda Ru + T
\]

\[
l = [e']_x v = [T]_x \lambda (Ru + T) = \lambda [T]_x Ru
\]

\[
E = [T]_x R
\]

Where \([e']_x\) and \([T]_x\) is the cross product matrix representation of \( e' \) and \( T \).

The essential matrix is very important since it is possible to extract the relative camera motion from it. This extraction is long, tedious and not covered in detail in this report. However it is covered in great detail in Multiple View Geometry. [10] The extraction is non trivial due to the fact that there are four combinations of rotation and translation pairs which can produce the same essential matrix. The translation can be positive or negative and the rotation can be rotated 180° around the line joining the camera centres.

### 3.3 The Fundamental Matrix

The fundamental matrix is an extension of the essential matrix. The difference is that the fundamental matrix relates an uncalibrated point to an uncalibrated line in the other camera instead of a calibrated point to a calibrated line. The relationship between the essential matrix and the fundamental matrix is shown below.

\[
F = K_2^{-T} E K_1^{-1}
\]

It is obvious from the relationship that it is impossible to extract the relative motion between the two cameras from the fundamental matrix, unless further information about the environment or camera matrices are acquired.

### 3.4 Minimization Methods

Minimization methods are methods which minimizes a function \( f \) based on searching over some set \( A \). The set \( A \) is typically subject to some constraints, in the case of
SfM this is often the chirality constraint.

\[
\begin{align*}
\min f(x) \\
x \in A \\
g_i(x) > 0 \\
i = 1...n
\end{align*}
\]

Most minimization methods work iteratively and does only find the global minimum on continuous functions and where the set A does only contain one local minimum, the global minimum. This section covers four methods three of which works as described above. These are gradient descent, Newton’s Method and the Levenberg-Marquardt algorithm. The last method in this section is Branch and Bound and while it still is iterative it always finds the global minimum regardless if there are several local minima.

### 3.4.1 Gradient Descent

Gradient descent is the most basic minimization method. It utilizes the gradient as a direction vector so the next point which is evaluated is based on the current point plus a vector pointing in the direction of the steepest descent.

\[
p_i = \|\nabla f(x_i)\| \\
x_{i+1} = x_i - \alpha p_i
\]

**Line Search**

The last line of the algorithm introduces an unknown variable \( \alpha \), the step length, which is determined by line search. Line search can be done in many different ways but in this project it was done by using Wolfe’s conditions. These conditions impose conditions on the function value and gradient in the new position based upon the function value and gradient in the current position. The conditions are the following.

\[
\begin{align*}
f(x_i + \alpha p_i) & \leq f(x_i) + c_1 \alpha (\nabla f(x_i))^T p_i \\
(\nabla f(x_i + \alpha p_i))^T \delta x & \geq c_2 (\nabla f(x_i))^T p_i \\
0 & < c_1 < c_2 < 1
\end{align*}
\]

The first condition ensures that the new value decrease is less than a scaled version of the tangent line. The second condition ensures that the new gradient is at least larger than a scaled version of the current gradient. The first condition ensures that the cost decrease is good enough and the second condition ensures that the movement along the line is far enough. Typical values, and the values used in the project, of \( c_1 \) and \( c_2 \) are \( c_1 = 10^{-4} \) and \( c_2 = 0.9 \). [11]
3.4.2 Newton’s Method

Newton’s method is based around using both first and second order information. The function \( f(x_i + \delta x) \) can be described by the Taylor expression shown below

\[
f(x_i + p_i) = f(x_i) + p_i f'(x_i) + 0.5 p_i^2 f''(x_i)
\]

By differentiating this Taylor expression with respect to the step, \( p_i \), and setting it to 0 it is possible to find an optimal point for the function by using the equation for \( p_i \) expressed below.

\[
x_{i+1} = x_i + p_i
\]

\[
p_i = -\frac{f'(x_i)}{f''(x_i)}
\]

For multiple variables this relationship can be expressed by using the Hessian and the gradient. The steps to reach the next position \( x_{i+1} \) is analogous with the single variable case.

\[
H = \begin{bmatrix}
\frac{\delta f}{\delta x_1 \delta x_1} & \cdots & \frac{\delta f}{\delta x_1 \delta x_n} \\
\vdots & \ddots & \vdots \\
\frac{\delta f}{\delta x_n \delta x_1} & \cdots & \frac{\delta f}{\delta x_n \delta x_n}
\end{bmatrix}
\]

\[
f(x_i + p_i) = f(x_i) + p_i \nabla f(x_i) + 0.5 p_i^T H(x_i) p_i
\]

\[
p_i = -H(x_i)^{-1} \nabla f(x_i)
\]

The benefits of using Newton’s method is that it is usually faster than gradient descent since it uses second order information. However when the problem is not sufficiently close to convex, Newton’s method may give non optimal directions. Once the direction is found, a line search is initiated which was explained in section 3.4.1.

Newton’s method can be slow to compute for many variables as the computing time of the Hessian is \( O(n^2) \) where \( n \) is the number of variables. Quasi Newton methods attempts to solve this by relaxing the computation of the Hessian. The most popular of the quasi Newton method is the BFGS method and it simplifies the computation of the Hessian by using the curvature experienced in the most recent step. [11] The Gauss-Newton method can also be used to simplify the calculation of the Hessian, however it is only applicable to least square problems.

3.4.3 Levenberg-Marquardt

The Levenberg-Marquardt algorithm is a common algorithm when solving least square problems. The Levenberg-Marquardt algorithm can be described as Newton’s method with a trust region. When the algorithm detects the local space as quadratic the step taken during that iteration is similar to the step taken by Newton’s method. But when the algorithm detects the local space as non quadratic the step taken is similar to the step computed by gradient descent.
The Levenberg-Marquardt algorithm is usually used in conjunction with a Jacobian representation of the Hessian, which was briefly mentioned in section 3.4.2. The following formula decides the step direction.

\[
J = \begin{bmatrix}
\frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n}
\end{bmatrix}
\]

\[(J^TJ + \lambda I)p_i = -J^T x_i\]

The calculation of \(\lambda\) is complex and since the results of this project does not rely on an implemented Levenberg-Marquardt algorithm it is not included. The information on how to calculate \(\lambda\) can however be found in literature on the subject. [11]

### 3.4.4 Branch and Bound

Branch and bound is an algorithm used to find a global optima when searching a solution space. This technique is especially useful when searching non-convex spaces since the other techniques covered in this section only finds a local minima when searching non-convex spaces. Branch and bounds works by using a function which can output a lower bound for the minimum of a certain subspace. By iteratively dividing the current search spaces into subspaces this enables discarding of some of the subspaces due to their lower bound being higher than a minimum already discovered.

In order for the branch and bound strategy to work, the function which outputs the lower bound to the minimum of a subspace must produce a lower bound which becomes closer to the actual minimum of the subspace when the subspace is smaller and effectively becoming the actual minimum when the subspace size goes toward zero. If these conditions are met then the algorithm will incrementally produce a smaller and smaller total size of the collection of viable space and in the end a solution can be found to a desired accuracy.

Branch and bound are used to solve many kinds of Computer Vision problems since the strategy provides optimal solutions. [12] However these algorithms can become very slow especially if the function used to described the minimum of a subspace is not efficient enough. An inefficient bounding function will lead to a subspace being less likely to be discarded and thus more subspaces in the next divide. An inefficient bounding function will thus lead to longer executions since all calculations must be done for all of the subspaces during the current iteration. Because of this problem, some of the algorithms uses relaxations of the problem in order to discard more subspaces for each branching. [13]
3.5 Standard Cost Functions

Cost functions are used in both triangulation and structure from motion algorithms. This section covers the standard cost functions which are most frequently used for these problems. The section also presents the calculations, since it is needed to fully understand the comparison between the cost functions in the report. All of the standard cost functions are based on different norms of reprojection errors, the 1-norm, 2-norm and \( \infty \)-norm in particular. The reprojection error is calculated as follows.

\[
\begin{align*}
    u_1 &= \begin{bmatrix} 1 & 0 \end{bmatrix} X \\
    v_1 &= \begin{bmatrix} R & T \end{bmatrix} X \\
    r_1 &= \| u_1 - u_0 \| \\
    r_2 &= \| v_1 - v_0 \| 
\end{align*}
\]

Where \( u_0 \) and \( v_0 \) are the originally detected corresponding 2D-coordinates.

3.5.1 \( L_1 \) Cost

\( L_1 \) is the most simple of the standard cost functions and is computed using the sum of the error distance of both cameras combined. The formula is presented below and the extension to several cameras is also included.

\[
\text{cost}(L_1) = \left\| r_1 \ r_2 \right\|_1 = \sum_{i=1}^{n} |r_i| 
\]

3.5.2 \( L_2 \) Cost

The \( L_2 \) cost function is the cost function most frequently used in SfM and is computed using the 2-norm of the reprojection error vector. The formula which is presented below is the square of \( L_2 \) which is used instead of \( L_2 \) since it removes the square root operation. Extension to several cameras is also included.

\[
\text{cost}(L_2) = \left\| r_1 \ r_2 \right\|_2^2 = \sum_{i=1}^{n} |r_i|^2 
\]

3.5.3 \( L_\infty \) Cost

\( L_\infty \) is a cost function which represents the infinity norm of the error distances. The infinity norm can more easily be understood as the maximum of the error distances.
The formula is presented below and the extension to several cameras is also included.

\[
\text{cost}(L_\infty) = \left\| r_1 \quad r_2 \right\|_\infty \\
= \sqrt{\sum_{i=1}^{n} |r_i|^\infty} \\
= \max(r_1, ..., r_n)
\]

3.6 Triangulation

Triangulation in Computer Vision is the process of estimating a 3D position from noisy 2D correspondence measurements. This section will primarily focus on stereo triangulation since this was the focus of the project. Triangulation for n-view scenarios will only be briefly covered. The three types of triangulation algorithms covered in this section are linear-eigen, \(L_2\), and \(L_\infty\) triangulation.

3.6.1 Linear-Eigen Triangulation

Linear triangulation is one of the most common triangulation techniques due to its great performance in terms of speed. The linear triangulation problem is commonly solved using an eigen method. [14] Given a complete projection matrix \(P\) the projection can be described as follows.

\[
u = w \begin{bmatrix} u_x & u_y \end{bmatrix}^T \\
u = PX
\]

This can be rewritten as two equations where \(P_1\) denotes the first column of the matrix \(P\).

\[
u_x P_3 X = P_1 X \\
u_y P_3 X = P_2 X
\]

In a case with at least two views this becomes a system of at least four equations which can be written in the standard \(Ax = 0\) form. By using the singular vector corresponding to the smallest singular value using singular value decomposition a solution which minimizes \(\|Ax\|\) can be found. [15]

3.6.2 \(L_2\) Triangulation

\(L_2\) triangulation has been an active area of research and the subset of stereo \(L_2\) triangulation has been shown to have a polynomial solution which also is invariant under transformation. [14] To be invariant under transformation in essence means that for any transformation \(H\) applied to the real 3D position \(X\) before applying the projection matrices, the resulting triangulated position is \(HX'\), where \(X'\) is the triangulated position without any transformation.
The algorithm for $L_2$ triangulation by Hartley and Sturm utilizes rigid transformations in order to establish an equation for the $L_2$ error that is only dependent on a single variable $t$ and can thus be solved. The rigid transformation consists of one translation and one rotation for each camera. The translation $L$ places the 2D correspondence coordinate $u$ at position $(0, 0, 1)^T$ and a rotation around the z-axis, $R$, is performed in order to place the epipole at the x-axis $(1, 0, f)^T$. By executing the operation for both cameras a different fundamental matrix $F'$ is established. $F'$ has the following form.

$$
T_1 = R_1 L_1 \\
T_2 = R_2 L_2 \\
F' = T_2^{-T} F T_1^{-1} \\
F' = \begin{bmatrix}
    f & f' & -f' & c \\
    -f & b & a & b \\
    -f & d & c & d
\end{bmatrix}
$$

It is possible to represent all epipolar lines in the first camera using only one variable, $t$. In this case the y-coordinate at which the epipolar line crosses the y-axis is used. The epipolar line in the first camera corresponds with a single epipolar line in the second camera and this produces a pair of epipolar lines. For a given $t$, the closest points in terms of $L_2$ error on each line is the orthogonal projection of the transformed correspondence coordinate on the line. Since it is possible to solve the problem for a specified $t$ in closed form it is possible to specify an equation for the $L_2$ error expressed by the single variable $t$.

$$
s(t) = \frac{t^2}{1 + (tf)^2} + \frac{(ct + d)^2}{(at + b)^2 + f^2 (ct + d)^2}
$$

To find the minimum of this equation the function is evaluated at the roots of it’s derivative, expressed below together with by the function $r(t)$ which is derived from $s'(t)$ when set to zero.

$$
s'(t) = \frac{2 t^2}{(1 + (tf)^2)^2} + \frac{2 (ad - bc) (at + b) (ct + d)}{((at + b)^2 + f^2 (ct + d)^2)^2}$$

$$
r(t) = t \left( (b + at)^2 + g^2 (d + ct)^2 \right)^2 - \left( f^2 t^2 + 1 \right)^2 (ad - bc) (b + a t) (d + c t) = 0
$$

See appendix A for the expanded $r(t)$ function

**Singularity problems**

To recover the global minimum in all cases one should take into account a possibility of a minimum at $t = \infty$ and thus $s(t)$ should be evaluated at $t \to \infty$. This
possibility corresponds to a minimum at the epipole. This scenario gives rise to a singularity problem. There is another method, optimal correction, which has been claimed to perform better at these singularity points. [16] Optimal correction was not considered in this project due to the fact that the stereo cameras did never view each other.

$L_2$ triangulation for more than two cameras

The triangulation algorithm proposed by Hartley and Sturm cannot be extended for more than two cameras. However there are other algorithms which can perform globally optimal $L_2$ error triangulation. [17] [18] These algorithms are based on branch and bound and use convex optimization techniques in order to find good lower bound functions. A simpler more fast branch and bound algorithm has also been presented due to the typically slow performance of convex optimization techniques. [19]

3.6.3 $L_\infty$ Triangulation

$L_\infty$ triangulation was first solved for the stereo problem in closed form similar to the stereo $L_2$ triangulation. [20] The $L_\infty$ algorithm is analogous to the $L_2$ algorithm ans since it was not used in the project, it is left out.

$L_\infty$ triangulation for more than two cameras is, as many $L_\infty$ problems in computer vision, easier to solve than its $L_2$ counterpart. [21] What makes $L_\infty$ triangulation easier to solve is the fact that the $L_\infty$ triangulation problem is quasi-convex. This stems from the fact that triangulation from one camera is a quasi-convex problem and the max of several quasi-convex functions is also quasi-convex. [22] This circumstance makes any attempt to solve the triangulation problem numerically more likely to find the global minimum since it does only contain one minimum. However in order to ensure that a global minimum is found, Second Order Cone Programming (SOCP) is used instead of local iterative methods. [23]

3.7 Structure from Motion (SfM)

Structure from Motion (SfM) is concerned with the problem of recovering the 3D structure based on the relative or absolute motions of one or several cameras. This project focused on stereo calibration but all algorithms and theories presented in this part are applicable to more than two view geometry, either directly or through a small extension. This section covers four different methods, the SVD-method, local iterative methods, bundle adjustment, global iterative methods, and object space error minimization. The minimization methods used in this project were the SVD-method and local iterative methods. However potential improvements from using the other methods are discussed throughout this report.
3.7.1 The SVD-Method

The most known algorithm in SfM, a sub-field of Computer Vision, is the 8-point-algorithm presented by Longuet-Higgins in 1981. [1] The algorithm estimates the essential matrix by using point correspondences in two images. Since its inception the method has been forked into many different variants but for simplicity SVD-methods is henceforth used as an umbrella term for these variants.

The basic standard implementation of the method is called the normalized 8-point-algorithm. This method can be used to calculate both the fundamental and the essential matrix. This section will cover the process to calculate the fundamental matrix. The process to obtain the essential matrix is analogous and conversion between the fundamental matrix and the essential matrix can be made as described in section 3.3.

The fundamental matrix is calculated by using point correspondences \( u \) and \( v \). For noise free data a point \( u \) form a line in camera 2 which passes through \( v \), leads to the following equality.

\[
v_i^T F u_i = 0
\]

This equality can be expressed as a homogeneous system \( Af = 0 \). This system can be solved by using Singular Value Decomposition (SVD) up to scale if \( A \) contains 8 or more point correspondences. The best approximation of \( f \) is the vector corresponding to the smallest singular value.

\[
\begin{align*}
  u &= \begin{bmatrix} x & y & 1 \end{bmatrix}^T \\
  u' &= \begin{bmatrix} x' & y' & 1 \end{bmatrix}^T \\
  A &= \begin{bmatrix}
    f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \\
    x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & x_1 & y_1 & 1 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & x_n & y_n & 1
  \end{bmatrix}
\end{align*}
\]

To improve this result the normalized 8-point-algorithm normalizes the coordinates before calculating the fundamental matrix and which leads to big improvement in result stability. [10] The normalizing transformation consists of scaling and translating the coordinates to satisfy an RMS distance of \( \sqrt{2} \). The transformation is, after calculation of the fundamental matrix \( F' \) relating the normalized coordinates,
applied to $F'$ in order to get the real fundamental matrix.

$$
T_u = \lambda_u \begin{bmatrix} 0 & 0 & x_u \\ 0 & 0 & y_u \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
T_v = \lambda_v \begin{bmatrix} 0 & 0 & x_v \\ 0 & 0 & y_v \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
u' = T_u u
$$

$$
v' = T_v v
$$

$$F = T_v^T F' T_u
$$

There is another restriction on $F$ which states that the fundamental matrix should be of rank 2. This equality is usually ensured by computing a singular value decomposition on $F$, setting its third singular value to 0 and then recomposing the fundamental matrix. When calculating the essential matrix there is another restriction. The essential matrix has one degree of freedom less than the fundamental matrix and the first and second singular value should be equal. This is usually resolved by setting the first and second singular value to the average of the two values.

**Further improvements**

Improvements of SVD-methods can be categorized into a few different categories. Some improvements are made through using a random sample of the correspondences in order to generate a better estimation of the fundamental matrix by using pseudo random exclusion technique. Examples of such methods are Ransac and LMedS. [24] [25] Other improvements are made by improving the number of correspondences needed to determine the fundamental matrix. One of the most common is a 7-point method which works by searching a linear function for the condition $\det(F) = 0$. [10] There are also methods working with as few as 5 point correspondences. [26] Much work has also been done regarding the condition $\det(F) = 0$. Instead of simply setting an element to zero the more statistically correct method would be to use the point on the surface $\det(F) = 0$ closest to the best approximation of $F$. [27]

**3.7.2 Local Iterative Methods**

Local iterative methods is a common way to solve the SfM problem. Three common local iterative methods are gradient descent, Newton’s method and the Levenberg-Marquardt algorithm. These algorithms were presented in section 3.4.1, 3.4.2 and 3.4.3 respectively.
In a calibrated stereo SfM problem the solution space consists of 6 variables, three for rotation and three for translation. Hessians, Jacobians or gradients are then calculated using finite differences as a basis in order to find a new better solution. The main problem with using local iterative methods is that they are susceptible to local minima. The methods are greedy and will only continue as long as they can find a better solution next to the current solution. This problem can either be mitigated by using a better initialization which has a higher probability of the local minimum being the global minimum or to use heuristic methods in order to find several local minima.

### 3.7.3 Bundle Adjustment

Bundle Adjustment is a common algorithm which operates similar to a local iterative method but instead of only including the camera and relative motion parameters also includes all 3D-points in the solution space of the problem. The algorithm was initially used in the field of photogrammetry but is applicable and used to solve computer vision problems as well. Since all 3D-points are included in the solution space any single solution usually consists of at least 100 variables and often many times more. This is a problem when calculating the Hessian since the Hessian would have at least 10000 elements while the benefit being that no triangulations would have had to be done in order to find the corresponding 3D positions and the relative camera motion.

Implementations of bundle adjustment however, uses the sparseness of the Hessian in order to calculate far fewer elements. The Hessian will have an appearance of only being non zero at its diagonal with the exception of elements representing the camera parameters. Since there usually are far more 3D-points than cameras this optimization saves a substantial amount of computing time. The characteristic appearance of the Hessian matrix is displayed in figure 3.1.

Bundle adjustment is a mature method and there are packages available which implement sparse bundle adjustment. Bundle adjustment has also reached speeds which makes it possible to use in real time applications under certain scenarios. There has been much research on bundle adjustment algorithms but inherently it has the same issue as local iterative methods, it is susceptible to local minima due to bad initialization or a high number of local minima in the solution space.

### 3.7.4 $L_2$ Minimization

$L_2$ minimization minimizes the $L_2$ (least square) cost. Minimizing $L_2$ cost is deemed representable of the problem since it corresponds with a posterior maximum likelihood when the noise on the 2D correspondence coordinates is assumed to be Gaussian. This equality is shown below by using the negative logarithm of the
Figure 3.1: Representation of the Hessian matrix used in bundle adjustment. Black areas represent elements with non-zero values and white areas represents elements with a value of zero. X represents a 3D feature and a P represents a projection matrix.

\[
P(u_i | \hat{u}_i) = C \exp \left( -\frac{\|u_i - \hat{u}_i\|^2}{2\sigma^2} \right)
\]

\[
\prod_i P(u_i | \hat{u}_i) = C' \prod_i \exp \left( -\frac{\|u_i - \hat{u}_i\|^2}{2\sigma^2} \right)
\]

\[
-\log \left( \prod_i P(u_i | \hat{u}_i) \right) = C'' \sum_i \|u_i - \hat{u}_i\|^2
\]

This is the reason why minimizing $L_2$ cost is considered as a statistically viable alternative. However this relies on an assumption of Gaussian noise and that maximum likelihood is a good measurement. [12] Another problem with $L_2$ minimization is that there are no globally optimal algorithms due the non-convexity of the problem. This infers that any method used risks being trapped in local minima. There have been research on algorithms which verifies whether the current solution is the global $L_2$ minimum however with no guarantee of optimality. [28]

### 3.7.5 $L_\infty$ Minimization

$L_\infty$ minimization is, as many $L_\infty$ problems in computer vision, much easier to solve than its $L_2$ counterpart. The SfM problem can be solved under $L_\infty$ when the relative rotation is known. This problem, similar to the triangulation problem, can be solved since it is quasi convex. [22] [21] Solving the rotation can be done using a branch and bound approach. [13] However even when using a relaxed bounding function the method does suffer from a very long execution time. Since $L_\infty$ is simply minimizing the sum of the maximum error for each correspondence set the method is usually only considered as an initialization technique.
3.7.6 Object Space Error Minimization

The object space error minimization calculates cost based on virtual rays from the cameras and the distance from these rays to the corresponding 3D feature. [29] One of the main benefits of this method is that it, similar to $L_{\infty}$ minimization, does not require any initialization. The problem formulation used by this method enables it to solve translation and 3D feature position in closed form and thus each iteration does only improve the rotation. The method has been suggested to rival bundle adjustment methods in robustness, speed and accuracy. [30]
Chapter 4

Method

This chapter presents the methods and datasets which were used to evaluate the new cost functions, which minimization techniques were used over what testing sets and the cost functions which were used in the project. The methods which were developed are called $L_2$ symmetry and variance symmetry. These methods were compared with $L_2$ with outlier rejection ($L_2$-WOR). The following sections will explain the details of the evaluation and the reason for the choices which were made. Section 4.1 and 4.2 explains the technique used for minimization and the error models used for the testing datasets. Section 4.4 covers the general objectives of the cost functions and the last section (4.5) of this chapter explains all the cost functions in detail as well as how the cost functions were used in the minimization process.

4.1 Minimization

Minimization is the process of finding the minimum of a cost function. As has been explained earlier, the scope of this project was to look for good cost functions using local iterative methods. These types of methods were chosen since they can use any type of cost function however it should be noted that they only find a local minimum and thus may not find the global minimum due to non-convexity which is common problem with most cost functions.

4.1.1 Minimization Algorithm

The minimization algorithm used for this project was Newton’s method and gradient descent which were covered in section 3.4.2 and 3.4.1 respectively. The Hessian and gradient was generated by finite differences. The choice to not use higher performance methods for instance Levenberg-Marquardt or Guass-Newton was due to that those methods are only applicable on the $L_2$ cost function. A local iterative method can converge slowly and thus the Wolfe conditions, using standard constants $c_1 = 0.9$ and $c_2 = 0.0001$, was implemented. Local iterative methods was chosen
over bundle adjustment since triangulation stereo triangulation can be solved relatively fast using closed form. Branch and bound was not considered since creating a bounding function is difficult if not impossible for most cost functions.

The basic operation of the minimization algorithm during each step is to first use newton’s method and if the method fails then use gradient descent instead. If both methods fail the search was considered done. The line search during newton’s method started by using the step length indicated by the method and then iteratively used the halve of the previous step length until a satisfying step length was found or the step length was smaller than the step length used to calculate the finite differences. Line search using gradient descent used a similar strategy but started at a step length of 1 since gradient descent does not provide a suggested step length for an unknown solution value.

4.1.2 Initialization

Since local iterative methods used on non-convex cost functions does not guarantee a global optima, a good initial solution will increase the likelihood of finding a good final solution. In this project the two initialization techniques primarily used were the following.

- The SVD-method
- \(L_2\) with outlier rejection(\(L_2\)-WOR)

The SVD-method was covered in section 3.7.1 and is a standard initialization algorithm when using local iterative methods. \(L_2\)-WOR is one of the cost functions and is covered in section 4.5.2. It should be noted that any cost function can be used as initialization for another cost function and the choice of \(L_2\)-WOR was due to \(L_2\) being a widely used cost function and usually solution space generated by it the function is not heavily non-convex.

4.1.3 Triangulation Algorithm

The triangulation algorithm is important since it is used both when calculating the cost function and when calculating each element of the Hessian and the gradient. The triangulation algorithm which was used during this project was the \(L_2\) stereo triangulation by Hartley and Sturm which was covered in section 3.6.2. The reason that this algorithm was chosen was due to its frequent usage and that it finds the global minimum of the \(L_2\) cost function which corresponds well with the cost functions this project proposed. The major drawbacks of this algorithm is its singularity near the epipole but this was not an issue for this project since none of the cameras in the real scenarios were visible to one another. Another drawback is the time consumption of the algorithm which is about 3-7 times higher than some of the more simple triangulation algorithms. [14].
The implementation of the algorithm requires solving a sixth degree equation. This was solved by finding the eigenvalues to the companion matrix of the expanded equation specified in appendix A. The eigenvalues were found using the eig function in Matlab which utilizes QR decomposition. Once the globally optimal point coordinates were found the algorithm used linear eigen triangulation to calculate the 3D-coordinate.

4.2 Synthetic Datasets

The datasets which were used in the project were of two types, synthetic and real. The synthetic datasets, which are covered in this section, were made to resemble the characteristics of the real datasets. The intrinsic camera parameters were specified similar to the systems used in the real scenarios and were considered known to the algorithm. The intrinsic camera parameters which were used are displayed in table 4.1.

<table>
<thead>
<tr>
<th>Focal Length ((f))</th>
<th>Skew ((\gamma))</th>
<th>Camera Centre (X(c_x))</th>
<th>Camera Centre (Y(c_y))</th>
</tr>
</thead>
<tbody>
<tr>
<td>700,0</td>
<td>0,0</td>
<td>500,0</td>
<td>500,0</td>
</tr>
</tbody>
</table>

The synthetic dataset were made up of 100 3D-points which were placed in a cuboid made up by a mesh of 5 x-coordinates, 5 y-coordinates and 4 z-coordinates. The cuboid was placed in order to fit in both cameras while still occupying a large area of both camera images. The position remained constant in all datasets as well as the relative motion of the cameras. These two variables were kept constant in order to isolate the noise as the only difference between the sets. The rotation and camera centre coordinates are presented in table 4.2.

<table>
<thead>
<tr>
<th>Yaw (\text{deg})</th>
<th>Pitch (\text{deg})</th>
<th>Roll (\text{deg})</th>
<th>Camera X</th>
<th>Camera Y</th>
<th>Camera Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-70,0^\circ)</td>
<td>(-5,0^\circ)</td>
<td>(-3,0^\circ)</td>
<td>(0,8354)</td>
<td>(0,0442)</td>
<td>(0,5478)</td>
</tr>
</tbody>
</table>

The noise which was generated were of four different types, each corresponding to a different group of datasets. These noise models shared the following noise parameters, presented in table 4.3.

<table>
<thead>
<tr>
<th>Inlier Sigma</th>
<th>Outlier Sigma</th>
<th>Outlier Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,5</td>
<td>5,0</td>
<td>10%</td>
</tr>
</tbody>
</table>
In total there were four different noise models and each noise model consisted of 100 generated datasets. The noise models used were Gaussian, rayleigh, uniform, and independent 2D uniform. The noise was applied to the 2D coordinates in order to analyze how well the cost functions handled noise. Gaussian noise was chosen due to it being a common noise model within the field were as the other noise models were chosen in order to ensure that the cost function was not optimized only for a single noise type. Each noise model are described below in its corresponding section. The figures in all the following sections, regarding noise models, were normalized for visibility by scaling their maximum value to 1.0.

4.2.1 Gaussian Error Distribution

The Gaussian error distribution is a standard distribution which is used in many different applications. This distribution gives the noise distance a Gaussian distribution. The probability of generating a point in terms of distance from its original position is displayed as both an image and a graph in figure 4.1.

\[ f(x) = C \exp\left(-\frac{x^2}{2\sigma^2}\right) \]

![Figure 4.1: Gaussian error model. The left figure displays probability as a function of 2D position. The right figure displays the probability as a function of distance.](image)

4.2.2 Rayleigh Error Distribution

The rayleigh error distribution is a special form related to the Gaussian distribution where the noise consists of two independent orthogonal Gaussian distributions. As is visible in figure 4.2 this gives a more even 2D distribution and changes the location of the peak probability in terms of distance. The probability of generating a point in terms of distance from its original position is displayed as both an image and a graph in figure 4.2.

![Figure 4.2: Rayleigh error model.](image)
4.2.3 Uniform Error Distribution

The uniform error distribution is a less commonly used error distribution in SfM. In this distribution the noise distance is uniformly distributed in the interval \([0, \sigma]\). The probability of generating a point in terms of distance from its original position is displayed as both an image and a graph in figure 4.3.

4.2.4 Independent Uniform Error Distribution

As an extension to the uniform distribution, the independent uniform error distribution consists of two orthogonal uniform noise distributions, one in x-direction and one in y-direction. This produces an even probability inside a square and a probability peak in terms of distance which is not at a distance of zero. The probability of generating a point in terms of distance from its original position is displayed as both an image and a graph in figure 4.4.
Figure 4.4: Independent 2D uniform error model. The left figure displays probability as a function of 2D position. The right figure displays the probability as a function of distance.

4.2.5 Outliers

Outliers in the datasets were modelled by using the independent uniform error distribution but using the parameter outlier sigma to control the error distribution. The outlier sigma was set relatively low since this was aligned with the real data. Each data point in each set had a random chance of becoming an outlier, controlled by the outlier percentage, thus the amount of outliers in each dataset was not constant. Both the outlier percentage and the outlier sigma were displayed in table 4.3.

4.3 Real Datasets

The real datasets are comprised of two different underlying datasets with different characteristics. They are both from stereo camera configurations with the same cameras but in different locations. The first dataset consists of 9720 2D point correspondences and was captured in an indoor setting, with poor light conditions. The second dataset consists of 10000 2D point correspondences and was captured in an outdoor setting at daytime. The point correspondences were captured by using a simple SfM model and projectile tracking. There were inherent differences between the indoor and the outdoor setting. The main difference were that the camera images were of lower quality in the indoor setting due to a darker environment and that the projectiles were only visible about 30 meters in Z-space. The camera intrinsics of both cameras were given at the start of the project and are presented in table 4.4.

<table>
<thead>
<tr>
<th>Camera</th>
<th>( f_x )</th>
<th>( f_y )</th>
<th>( \gamma )</th>
<th>( c_x )</th>
<th>( c_y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>699.93</td>
<td>700.16</td>
<td>0.0</td>
<td>504.65</td>
<td>513.91</td>
</tr>
<tr>
<td>Right</td>
<td>702.69</td>
<td>702.92</td>
<td>0.0</td>
<td>500.62</td>
<td>520.53</td>
</tr>
</tbody>
</table>
The points were gathered from the projectile correspondences from the old system and was constrained to a camera windows of 1040 by 1040 pixels. The distributions of the correspondence points are displayed in the figures below where the intensity in the picture represents the relative amount of point correspondences in that area. Figure 4.5 displays the distribution of the indoor set and figure 4.6 displays the distribution of the outdoor set.

Figure 4.5: The correspondence point distribution in the camera images. The left image is from the left camera and the right image is from the right camera. The source of the distribution is the indoor set.

Figure 4.6: The correspondence point distribution in the camera images. The left image is from the left camera and the right image is from the right camera. The source of the distribution is the outdoor set.

As can be seen from the picture, the outdoor set does not only have a smaller z-space but does also have a higher concentration of correspondence points to a smaller area. In general it can also be seen that the correspondence points were not found in the full size of the camera image but are instead constricted to a smaller part of the camera image.

4.4 Cost Function Objectives

This section will cover the different objectives which is important when creating a cost function, the objectives will also provide a framework for determining the performance of the cost functions. Each cost function may not individually accomplish all objectives but it is important that the final algorithm does not perform
poorly in any of the objectives. The objectives discussed in this section are outlier insusceptibility, inlier error minimization, and pseudo convexity. These objectives are presented in their corresponding sections, 4.4.1, 4.4.2, and 4.4.3.

4.4.1 Outlier Insusceptibility

Outlier insusceptibility is important for the algorithm since any outlier modelled as an inlier will influence the optimization negatively. Outliers might produce bad behaviour in a cost function due to the fact that they may contribute heavily to the cost and thus a non-optimal motion estimation may render a lower total cost due to a lower cost contribution from the outliers. In order for a cost function to be insusceptible to outliers it must not add too high penalties to the cost due to big errors, or remove the outlier prior to applying motion estimation with the cost function.

4.4.2 Error Representation

If outliers can be discarded then the algorithm must produce a cost which represents the error of the inliers well. The errors of inlier reprojection can be calculated by different types of functions. However it is important that the best solution of the cost function, using the noisy data, also provides a good solution when evaluated against the zero noise data. Thus a good algorithm should render a cost space where minimization using noisy data is similar to minimization using zero noise data.

4.4.3 Pseudo Global Convergence with Good Initialization

Most of the cost functions used in SfM are non-convex on the complete search space. However an important property of the cost function is that it converges well when applied in optimization. In most cases this is dependent on the initialization point in the search space. The convergence property is similar to pseudo convexity which was discussed in section 3.6.3. A search space fulfilling this property implies that a gradient descent algorithm should find the global minimum. [31] Complete global convergence is difficult to prove, although it can in special cases and only possible using special cost functions. However in order to satisfy pseudo global convergence the local minimum found should be in an area close to the global minimum. The cost functions and methods used in this project does not provide global convergence but instead relies on a good initialization point and pseudo global convergence.

4.5 Cost Functions

This section covers the cost functions which were evaluated in this project. In addition to the functions presented in this section, SVD and standard $L_2$ were also evaluated. SVD represents the solution generated by using the SVD method and
was covered in section 3.7.1. $L_2$ is a standard cost function and was covered in section 3.7.4.

### 4.5.1 $L_2$ with Outlier Tolerance ($L_2$-WOT)

$L_2$ with outlier tolerance is similar to the standard $L_2$ cost function but it also includes outlier insusceptibility. Standard $L_2$ was as shown in section 3.7.4 equal to minimizing the negative log posterior likelihood of a Gaussian distribution. Previous articles have indicated the possibility of increasing tolerance for outliers by adding a floor to the function. [2] For $L_2$-WOT this was accomplished by specifying a probability for outliers and a region of interest. The function also requires a sigma value however this does not need to correlate well to the sigma of the error distribution model. The formula for the cost function is presented below.

$$f(x) = \sum_i \left( -\log \left( \exp \left( \frac{\|u_i - \hat{u}_i\|^2}{2\sigma^2} \right) + \frac{\text{error probability}}{\text{region of interest}} \right) \right)$$

In the tests conducted the parameters which were used are presented in table 4.5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Error prob</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>Region of interest</td>
<td>10,0</td>
<td>10,0</td>
</tr>
</tbody>
</table>

$L_2$ with outlier tolerance can sometimes have weak convergence properties especially when the data contains many outliers or outliers with large errors. To mitigate this problem the sigma can initially be set high and then running the cost function several times while always lowering the sigma. This method however has the disadvantage of increasing the computation time of the motion estimation.

### 4.5.2 $L_2$ with Outlier Rejection ($L_2$-WOR)

$L_2$ with outlier rejection is a simple extension on the $L_2$ with outlier tolerance cost function. This function first uses $L_2$ with outlier tolerance using local iterative methods. When the function has terminated the point correspondences with a reprojection error above a certain error threshold are rejected. Thus a reduced set of point correspondences remain. These point correspondences are then evaluated using a standard $L_2$ cost function. For the purposes of this test a threshold of 2,0 was used.

### 4.5.3 $L_2$ Difference

The $L_2$ difference function uses the same outlier rejection as $L_2$ with outlier rejection. On the reduced set it uses a cost function which is comprised of an $L_2$ cost function and an added $L_2$ symmetry term. The symmetry term is determined by
using the difference of the $L_2$ distance projected on the image's two different axes. The difference calculated is the difference of the sum of the $L_2$ distances from one side and the sum of the $L_2$ distances from the other side. This process is done for both axes which results in one $L_2$ term and two $L_2$ difference terms per camera. The cost function also uses a multiplier, $\beta$, which specifies the influence of the symmetry term. Additionally, any point further away than 1 pixel in error was assigned an error of 1 pixel. The formula is presented below.

\[
\text{diff}_x(x) = \left| \sum_i^{\forall (u_{ix} - \hat{u}_{ix}) > 0} \|u_{ix} - \hat{u}_{ix}\|^2 - \sum_i^{\forall (u_{ix} - \hat{u}_{ix}) < 0} \|u_{ix} - \hat{u}_{ix}\|^2 \right|
\]

\[
\text{diff}_y(x) = \left| \sum_i^{\forall (u_{iy} - \hat{u}_{iy}) > 0} \|u_{iy} - \hat{u}_{iy}\|^2 - \sum_i^{\forall (u_{iy} - \hat{u}_{iy}) < 0} \|u_{iy} - \hat{u}_{iy}\|^2 \right|
\]

\[
f(x) = \sum_i \left( \|u_i - \hat{u}_i\|^2 \right) + \beta \left( \text{diff}_x(x) + \text{diff}_y(x) \right)
\]

### 4.5.4 Variance Difference

The variance difference function is similar to the $L_2$ difference function. It starts by using the same outlier rejection as $L_2$ with outlier rejection. On the reduced set it uses a cost function which is comprised of a total variance term from each axis and an added variance symmetry term. The symmetry term is comprised of two symmetry terms, one per axis. The symmetry term is the difference of the variance of error vectors projected on one axis from one side of the axis compared to the same operation from the other side of the axis. This process is done for both axes which results in one variance term and two variance difference terms per camera. The cost function also uses a multiplier, $\beta$, which specifies the influence of the symmetry term. Similarly to the variance difference function any point with an error above 1 pixel is assigned an error of 1 pixel. The formula is presented below.

\[
\text{var}_{x>0}(x) = \frac{1}{n_{x>0}} \sum_i^{\forall (u_{ix} - \hat{u}_{ix}) > 0} \|u_{ix} - \hat{u}_{ix}\|^2
\]

\[
\text{var}_{x<0}(x) = \frac{1}{n_{x<0}} \sum_i^{\forall (u_{ix} - \hat{u}_{ix}) < 0} \|u_{ix} - \hat{u}_{ix}\|^2
\]

\[
\text{var}_{y>0}(x) = \frac{1}{n_{y>0}} \sum_i^{\forall (u_{iy} - \hat{u}_{iy}) > 0} \|u_{iy} - \hat{u}_{iy}\|^2
\]

\[
\text{var}_{y<0}(x) = \frac{1}{n_{y<0}} \sum_i^{\forall (u_{iy} - \hat{u}_{iy}) < 0} \|u_{iy} - \hat{u}_{iy}\|^2
\]

\[
f(x) = (\text{var}_{x>0}(x) + \text{var}_{x<0}(x) + \text{var}_{y>0}(x) + \text{var}_{y<0}(x)) + \\
+ \beta \left( |\text{var}_{x>0} - \text{var}_{x<0}| + |\text{var}_{y>0} - \text{var}_{y<0}| \right)
\]
4.5.5 \( \beta \) Values of the Symmetry Functions

As the symmetry functions include \( \beta \) values they may generate an infinite amount of variations. Due to time constraints of the projects, only two \( \beta \) values were chosen for deeper analysis, 0.5 and 1.0. These \( \beta \) values had shown good performance during early symmetry function performance tests which was the reason for choosing the particular \( \beta \) values 0.5 and 1.0.

4.6 Measurements

The measurements which were used to evaluate the cost functions were based on the \( L_1 \) error norm. The error was additionally calculated using two different sources, noisy reprojection error and real projection error.

- Noisy reprojection error
- Real projection error

The noisy reprojection error is calculated by using the noisy correspondence coordinates to triangulate a 3D-point and then evaluating the distance between the reprojection of the 3D-point and the original noisy correspondence coordinates. The noisy reprojection error is available to the system during motion estimation since it does not rely upon noiseless 3D-points and point correspondences.

The real projection error evaluates the distance between the correspondence coordinates before any noise was applied to them and the reprojection of the true 3D-point. This real projection error is useful since the error evaluated is only influenced by the error in the motion estimation and not by the noise in the correspondence coordinates. It should be noted that in the real projection error, the complete cost will be allocated to the second camera since the first camera is always assumed to have no translation and no rotation. The real projection error is typically not available during motion estimation and was only used to analyse the accuracy of the motion estimation in the results section.
Chapter 5

Results

The results which are presented in this paper are based on both synthetic and real data. More weight is placed on the synthetic data since ground truth is available and methods can be compared using the real projection error measurement while only the noisy reprojection measurement can be calculated for the real data.

5.1 Synthetic Data

The synthetic data was, as mentioned before, divided into four different datasets based on the noise model used to displace the 2D correspondence coordinates. The noise models used were Gaussian, rayleigh, uniform and independent uniform noise, all of which were covered in more detail in section 4.2. The synthetic data section is divided in two major parts. The first part covers the performance of the basic functions and the second part covers the performance of the difference functions, which are the contributions of this paper.

5.1.1 Basic Functions - Noisy Reprojection Error Measurement

The tests in this sections display the performance of the basic algorithms. The graphs below display the mean percentage of points inside a certain error distance using the noisy reprojection error. The noisy reprojection error is important since it visualizes how well the function improves the result based on the data that is accessible to the function. There are four graphs for each of the four different datasets. Figure 5.1 displays the results from the Gaussian set, figure 5.2 displays the results from the rayleigh set, figure 5.3 displays the results from the uniform set and figure 5.4 displays the results from the independent uniform set.
Figure 5.1: Mean percentage of points within the specified error distance using Gaussian testing set. The error type is the noisy reprojection error.

Figure 5.2: Mean percentage of points within the specified error distance using rayleigh testing set. The error type is the noisy reprojection error.
Figure 5.3: Mean percentage of points within the specified error distance using uniform testing set. The error type is the noisy reprojection error.

Figure 5.4: Mean percentage of points within the specified error distance using independently uniform testing set. The error type is the noisy reprojection error.
Summary

As can be seen the improvement is hard to notice except that every cost function performs better than the SVD-method. $L_2$-WOR shows slightly better performance than $L_2$-WOT and $L_2$ but only slightly so. There is a clear drop off after about 90% which corresponds to the outlier percentage of 10%. This suggests that $L_2$ or $L_2$-WOT can be used to reject outliers with small errors. $L_2$-WOR shows a similar performance since it is based on the outlier rejection of $L_2$-WOT.

5.1.2 Basic Functions - Real Projection Error Measurement

Since the synthetic data does have the ground truth available, the real projection error is used for the synthetic data. Below, the graphs visualize the mean percentage of points within a certain error distance for a certain data set using the ground truth. Figure 5.5 displays the results from the Gaussian set, figure 5.6 displays the results from the rayleigh set, figure 5.7 displays the results from the uniform set and figure 5.8 displays the results from the independent uniform set.

![Figure 5.5: Mean percentage of points within the specified error distance using Gaussian testing set. The error type is the real projection error.](image-url)
Figure 5.6: Mean percentage of points within the specified error distance using Rayleigh testing set. The error type is the real projection error.

Figure 5.7: Mean percentage of points within the specified error distance using uniform testing set. The error type is the real projection error.
Summary

The data above shows that the differences between the cost function algorithms are far greater when measuring the real projection error than when measuring the noisy reprojection error. A better real projection error can be interpreted as the cost function being less sensitive to the noise and instead finding a solution which is closer to the true motion. In these graphs it is evident that $L_2$-WOT is better than standard $L_2$ but the improvement does seem only minor. However the datasets used does have a low outlier percentage and low outlier sigma which poses only small problems to the standard $L_2$ cost function. A separate test on the performance of $L_2$-WOT in a high outlier percentage and sigma environment is presented in section 5.1.3.

From this data it can be seen that $L_2$-WOR outperforms the other basic algorithms in regards to how close the calculated motion reprojects the points in comparison to the ground truth. Additionally $L_2$-WOR does also outperform the other cost function algorithms in regards to worst case scenario which is included in appendix B, figures B.1,B.2,B.3, and B.4. $L_2$-WOR is used as a base comparison measurement in this paper and the proposed symmetry cost functions are evaluated using the same outlier rejection as $L_2$-WOR. When visualizing other algorithms the relative performance between these algorithms and $L_2$-WOR will at many times be shown instead of their absolute performance in order to increase visibility of differences.
5.1.3 Performance of $L_2$-WOT

Performance of $L_2$ with outlier tolerance is barely visible due to the low noise of the primary datasets. A more noisy dataset was thus created to visualize the effect of $L_2$ with outlier tolerance especially while using the decreasing sigma method. The dataset was of Gaussian noise type, featured 30% outliers and the outlier noise was of the independent uniform type with a sigma of 25. The other dataset properties were held constant. Figure 5.9 displays the mean percentage of points within a certain noisy reprojection error for this dataset using the methods SVD $L_2$ and $L_2$-WOT. $L_2$-WOR was not included since it relies upon $L_2$-WOT for outlier rejection.

![Figure 5.9: Percentage of points inside the specified error distance using the real projection error. The dataset used was a high outlier dataset and the error was of Gaussian type.](image)

This figure clearly display an increase in performance by using $L_2$-WOT. There is a clear drop off around 70% corresponding to the 30% outlier percentage which would suggest that the function does separate inliers and outliers well. However to further display the properties of $L_2$-WOT and to make the advantages of using it more clear the real projection error must also be analysed. Figure 5.10 displays the mean percentage of points within a certain real projection error for this dataset using the methods SVD $L_2$ and $L_2$-WOT.
As can be seen in the figure, $L_2$-WOT clearly outperforms the other alternatives when using the real projection error. This would indicate that $L_2$-WOT can with greater accuracy optimize the true error distance of a point correspondence than the other cost functions when there is a strong presence of outliers. That does indicate that $L_2$-WOT calculates a better motion estimation than the other algorithms displayed. From this result it should be clear that $L_2$-WOT is also a better candidate than the other algorithms to decide whether a point correspondence is an inlier or an outlier. Accordingly $L_2$-WOT was used as a basis for outlier removal for the algorithms presented later in this report. In specific the algorithms using $L_2$-WOT for outlier removal were $L_2$-WOR, L2-diff and var-diff. There are several different methods for choosing an appropriate threshold for outlier removal, however in this project a fixed threshold of 2 pixels of total noisy reprojection error was used.

### 5.1.4 Difference Functions with SVD Initialization

This section displays the results reached by using difference cost functions with SVD initialization. Difference functions were covered in section 4.5.3 and 4.5.4. The beta values tested were 0.5 and 1.0 for each of the difference functions. The figures below display the percentage points of correspondence points inside a certain real projection error distance relative to the same measurement for $L_2$-WOR. Figure 5.11 displays the results from the Gaussian set, figure 5.12 displays the results from the rayleigh set, figure 5.13 displays the results from the uniform set and figure 5.14 displays the results from the independent uniform set.
Figure 5.11: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the Gaussian testing set. The error type is the real projection error and SVD was used as initialization.

Figure 5.12: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the rayleigh testing set. The error type is the real projection error and SVD was used as initialization.
Figure 5.13: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the uniform testing set. The error type is the real projection error and SVD was used as initialization.

Figure 5.14: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the independently uniform testing set. The error type is the real projection error and SVD was used as initialization.
Additional Data

The figures above visualizes the mean improvement relative to $L_2$-WOR. The worst performance figures (i.e. the worst percentage of points inside a certain distance for a single solution) for the same test is presented for reference in appendix B, figures B.5, B.6, B.7, and B.8. No figures are presented on the noisy reprojection error however the following tables displays the main characteristics of both the noisy reprojection error and the real projection error. Each table, 5.1, 5.2, 5.3, and 5.4, represent a specific testing dataset.

Table 5.1: Measured errors for the Gaussian testing set using SVD initialization

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.690</td>
<td>1.451</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.690</td>
<td>1.464</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.692</td>
<td>1.460</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.701</td>
<td>1.462</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.721</td>
<td>1.453</td>
</tr>
</tbody>
</table>

Table 5.2: Measured errors for the rayleigh testing set using SVD initialization

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.810</td>
<td>1.292</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.812</td>
<td>1.303</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.813</td>
<td>1.301</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.819</td>
<td>1.307</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.825</td>
<td>1.299</td>
</tr>
</tbody>
</table>

Table 5.3: Measured errors for the uniform testing set using SVD initialization

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.558</td>
<td>1.474</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.561</td>
<td>1.484</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.569</td>
<td>1.477</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.590</td>
<td>1.481</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.618</td>
<td>1.474</td>
</tr>
</tbody>
</table>
The data displayed in the tables clearly suggest that the noisy reprojection error does not represent the real projection error. Especially when looking at $L_2$ difference with a beta of 0.5 it should be apparent to the reader that the real projection error can be far lower in terms of mean, variance and max while still exhibiting a similar noisy reprojection error.

Summary

From the different types of results presented it is clear that the $L_2$ difference cost function with SVD initialization consistently outperforms $L_2$-WOR at most of the error distances and that $L_2$ difference with a 0.5 multiplier provides better results than the same cost function with a multiplier of 1. While the noisy reprojection error is similar between the different methods, the real projection error shows clear differences between the cost functions. Results were shown which indicates that the $L_2$ difference cost function can achieve greater results than $L_2 - WOR$ using a simple SVD initialization, thus indicating that the difference method can provide competitive results without providing an advanced method of initialization. The performance of the variance difference were however, as was shown, not consistent across all noise models when using SVD as initialization.

5.1.5 Difference Functions with $L_2$-WOR Initialization

In order to further expand upon the performance of the symmetry algorithms it was important to research how they performed using $L_2$-WOR instead of SVD as the initialization point. Figure 5.15 displays the results from the Gaussian set, figure 5.16 displays results from the rayleigh set, figure 5.17 displays results from the uniform set and figure 5.18 displays results from the independent uniform set. The performance displayed is relative to the performance of $L_2$-WOR.

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.629</td>
<td>1.367</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.630</td>
<td>1.377</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.632</td>
<td>1.375</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.639</td>
<td>1.382</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.661</td>
<td>1.380</td>
</tr>
</tbody>
</table>
Figure 5.15: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the Gaussian testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.

Figure 5.16: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the Rayleigh testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.
Figure 5.17: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the uniform testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.

Figure 5.18: Percentage points difference relative to $L_2$-WOR where the measured percentage is the mean percentage of points within the specified error distance for the independently uniform testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.
Additional Data

The figures above clearly visualizes an improvement relative to $L_2$-WOR. The worst performance figures (i.e. the worst percentage of points inside a certain distance for a single solution) for the same test is presented for reference in appendix B, figures B.9, B.9, B.9, and B.9. No figures are presented on the noisy reprojection error however the following tables display the main characteristics of both the noisy reprojection error and the real projection error. Each table, 5.5, 5.6, 5.7, and 5.8, represent a specific training dataset.

**Table 5.5: Measured errors for the Gaussian testing set using $L_2$-WOR as initialization**

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.690</td>
<td>1.451</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.690</td>
<td>1.462</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.693</td>
<td>1.456</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.693</td>
<td>1.454</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.695</td>
<td>1.453</td>
</tr>
</tbody>
</table>

**Table 5.6: Measured errors for the rayleigh testing set using $L_2$-WOR as initialization**

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.810</td>
<td>1.292</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.812</td>
<td>1.302</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.813</td>
<td>1.298</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.817</td>
<td>1.295</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.821</td>
<td>1.292</td>
</tr>
</tbody>
</table>

**Table 5.7: Measured errors for the uniform testing set using $L_2$-WOR as initialization**

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.558</td>
<td>1.474</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>0.561</td>
<td>1.484</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>0.568</td>
<td>1.473</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>0.559</td>
<td>1.484</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>0.560</td>
<td>1.478</td>
</tr>
</tbody>
</table>
Table 5.8: Measured errors for the independently uniform testing set using \(L_2\)-WOR as initialization

<table>
<thead>
<tr>
<th></th>
<th>Noisy reprojection error</th>
<th>Real projection error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>var</td>
</tr>
<tr>
<td>(L_2)-WOR</td>
<td>0.629</td>
<td>1.367</td>
</tr>
<tr>
<td>(L_2)-diff (\beta):0.5</td>
<td>0.631</td>
<td>1.376</td>
</tr>
<tr>
<td>(L_2)-diff (\beta):1.0</td>
<td>0.633</td>
<td>1.370</td>
</tr>
<tr>
<td>Var-diff (\beta):0.5</td>
<td>0.632</td>
<td>1.370</td>
</tr>
<tr>
<td>Var-diff (\beta):1.0</td>
<td>0.637</td>
<td>1.368</td>
</tr>
</tbody>
</table>

The data displayed in the tables clearly suggests, as the SVD initialized test, that the noisy reprojection error does not represent the real projection error. The major difference between the two initializations is that when using \(L_2\)-WOR all of the difference functions are close to \(L_2\)-WOR in terms of mean, variance and max noisy reprojection error but all also improves on \(L_2\)-WOR in terms of mean, variance of and maximum real projection error.

**Summary**

From this data it is clear that the \(L_2\) difference cost functions consistently outperforms \(L_2\)-WOR at most of the error distances and that \(L_2\) difference with a 0.5 multiplier performs better than the same cost function with a multiplier of 1. This provides evidence that the \(L_2\) difference cost functions can achieve greater results than \(L_2\)-WOR using \(L_2\)-WOR as initialization. Further this indicates that the \(L_2\) difference cost functions are more insensitive to noise than \(L_2\)-WOR. The performance of the variance difference algorithm was far more consistent when using \(L_2\)-WOR which suggests that the solution space of the variance difference cost function does have larger subset of the solution space where local minima frequently occur. However it might be useful when used for local improvements. This property, the convergence property, is analysed in more detail in section 5.1.6. As a final insight, by comparing the data in this section with the previous, the data does suggest that the \(L_2\) difference function with a beta of 0.5 produces the result with the lowest real projection error of the cost functions tested.

5.1.6 Convergence

During the project a simple convergence test was also conducted. The test measured the distance between the two solutions produced by the functions when using either SVD or \(L_2\)-WOR as the initialization point. This test will thus show the relative convergence in absolute numbers between the different symmetry cost functions. The mean difference between the different initialization for each algorithm is displayed in table 5.9. Table 5.10 displays the max difference when using different
initializations and table 5.11 displays the mean and max difference between the SVD and the \(L_2\)-WOR initialization points.

**Table 5.9: Mean solution difference in terms of real projection error between using SVD initialization and using \(L_2\)-WOR initialization**

<table>
<thead>
<tr>
<th></th>
<th>Gaussian</th>
<th>Rayleigh</th>
<th>Uniform</th>
<th>Independent uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_2)-diff (\beta):0.5</td>
<td>4.032e-04</td>
<td>5.132e-04</td>
<td>1.715e-04</td>
<td>3.999e-04</td>
</tr>
<tr>
<td>Var-diff (\beta):0.5</td>
<td>1.244e-03</td>
<td>9.159e-04</td>
<td>1.545e-03</td>
<td>1.061e-03</td>
</tr>
<tr>
<td>Var-diff (\beta):1.0</td>
<td>1.152e-03</td>
<td>1.088e-03</td>
<td>1.172e-03</td>
<td>1.073e-03</td>
</tr>
</tbody>
</table>

**Table 5.10: Max solution difference in terms of real projection error between using SVD initialization and using \(L_2\)-WOR initialization**

<table>
<thead>
<tr>
<th></th>
<th>Gaussian</th>
<th>Rayleigh</th>
<th>Uniform</th>
<th>Independent uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_2)-diff (\beta):0.5</td>
<td>3.112e-03</td>
<td>2.513e-03</td>
<td>1.575e-03</td>
<td>2.605e-03</td>
</tr>
<tr>
<td>(L_2)-diff (\beta):1.0</td>
<td>3.241e-03</td>
<td>4.287e-03</td>
<td>3.509e-03</td>
<td>2.219e-03</td>
</tr>
<tr>
<td>Var-diff (\beta):0.5</td>
<td>5.666e-03</td>
<td>5.531e-03</td>
<td>1.314e-02</td>
<td>8.043e-03</td>
</tr>
<tr>
<td>Var-diff (\beta):1.0</td>
<td>5.358e-03</td>
<td>4.029e-03</td>
<td>6.343e-03</td>
<td>6.960e-03</td>
</tr>
</tbody>
</table>

**Table 5.11: Mean and maximum solution difference in terms of real projection error between using SVD and the \(L_2\)-WOR solution**

<table>
<thead>
<tr>
<th></th>
<th>Gaussian</th>
<th>Rayleigh</th>
<th>Uniform</th>
<th>Independent uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3.927e-03</td>
<td>4.159e-03</td>
<td>3.999e-03</td>
<td>3.647e-03</td>
</tr>
<tr>
<td>Max</td>
<td>8.923e-03</td>
<td>1.077e-02</td>
<td>1.065e-02</td>
<td>1.123e-02</td>
</tr>
</tbody>
</table>

From this data it is clear that all difference functions either do encounter problems with local minima or creates a difficult search space which the numerical methods have problems solving. However it is also apparent that all methods seems to converge on an area smaller than the area between the SVD and \(L_2\)-WOR solution. This property is especially clear when looking at the maximum values. Another insight from the data is that the \(L_2\) difference cost functions display a better convergence than the variance difference cost functions.

### 5.1.7 Error Representation

How well the function represents the underlying error is difficult to assess over the complete search space since there are many variables which would have to be measured over the complete search space. Instead this project measured the different
final solutions in order to assess whether the solution representing the smaller error were better in terms of real projection error. The following tables, 5.12 and 5.13, display differences when using the minimum value of the two different solutions. The first solution was generated by the cost function using SVD as initialization. The second solution is generated similarly but instead used $L_2$-WOR as initialization. Table 5.13 displays the percentage of datasets where a better solution according to the cost function corresponded with a better solution in terms of the real projection error. Table 5.12 displays the mean change in real projection error when switching from what the function regarded as the worse solution to what the function regarded as the better solution.

Table 5.12: Mean improvement in terms of real projection error by choosing the best solution of a solution generated by SVD initialization and a solution generated by $L_2$-WOR initialization. Each row represents a different cost function

<table>
<thead>
<tr>
<th>Cost Function</th>
<th>Gaussian</th>
<th>Rayleigh</th>
<th>Uniform</th>
<th>Independent uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-diff $\beta$:0,5</td>
<td>0,034</td>
<td>0,047</td>
<td>-0,017</td>
<td>-0,008</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1,0</td>
<td>-0,024</td>
<td>0,041</td>
<td>-0,048</td>
<td>0,036</td>
</tr>
<tr>
<td>Var-diff $\beta$:0,5</td>
<td>0,129</td>
<td>0,011</td>
<td>0,254</td>
<td>0,094</td>
</tr>
<tr>
<td>Var-diff $\beta$:1,0</td>
<td>0,055</td>
<td>0,125</td>
<td>0,145</td>
<td>0,131</td>
</tr>
</tbody>
</table>

Table 5.13: Percentage of improved testing sets in terms of mean real projection error by choosing the best solution of a solution generated by SVD initialization and a solution generated by $L_2$-WOR initialization. Each row represents a different cost function

<table>
<thead>
<tr>
<th>Cost Function</th>
<th>Gaussian</th>
<th>Rayleigh</th>
<th>Uniform</th>
<th>Independent uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-diff $\beta$:0,5</td>
<td>54,13%</td>
<td>59,54%</td>
<td>44,33%</td>
<td>45,82%</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1,0</td>
<td>51,91%</td>
<td>57,70%</td>
<td>50,23%</td>
<td>60,41%</td>
</tr>
<tr>
<td>Var-diff $\beta$:0,5</td>
<td>55,16%</td>
<td>55,55%</td>
<td>56,88%</td>
<td>53,58%</td>
</tr>
<tr>
<td>Var-diff $\beta$:1,0</td>
<td>55,24%</td>
<td>62,04%</td>
<td>60,67%</td>
<td>60,73%</td>
</tr>
</tbody>
</table>

A mean value above zero and a percentage above 50 does indicate that is generally is beneficial to choose the solution corresponding to a lower cost, as calculated by the cost function. Interesting to note is that the variance difference functions have better results but it should be taken into account that their convergence is lower. Another insight from the data is that $L_2$ difference with a beta of 0,5 does show good result for the testing sets based on the Gaussian distribution while the opposite is true when evaluating the testing sets based on the uniform distribution. Overall this would indicate that for a dataset with unknown error properties there is at most a small benefit of additional searching within the convergence area for $L_2$ difference with a beta of 0,5. For the other cost functions it seems to be more beneficial to conduct a more thorough search. Much of these findings are most likely be explained by the better convergence property of $L_2$ difference with a beta of 0,5.
5.1.8 Iterations

Speed was not measured during the tests since no emphasis was placed on optimization and thus any absolute time measures would not be indicative of an optimized performance. However the test did record the total number of iterations which was needed for each dataset. The following table (5.14) presents the mean number of iterations required for each cost function. As is visible from the table the variance difference cost functions did use substantially fewer iterations which do align well with their lower convergence levels.

Table 5.14: Mean number of iterations when using the specified cost function.

<table>
<thead>
<tr>
<th></th>
<th>Gaussian</th>
<th>Rayleigh</th>
<th>Uniform</th>
<th>Independent uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-WOR</td>
<td>27.96</td>
<td>28.49</td>
<td>27.83</td>
<td>28.15</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5</td>
<td>30.89</td>
<td>31.71</td>
<td>27.90</td>
<td>30.08</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0</td>
<td>31.97</td>
<td>32.95</td>
<td>31.43</td>
<td>32.42</td>
</tr>
<tr>
<td>Var-diff $\beta$:0.5</td>
<td>20.78</td>
<td>23.32</td>
<td>19.49</td>
<td>22.08</td>
</tr>
<tr>
<td>Var-diff $\beta$:1.0</td>
<td>22.36</td>
<td>24.58</td>
<td>18.08</td>
<td>20.81</td>
</tr>
</tbody>
</table>

5.2 Real Data

The real data, introduced in section 4.3, had two different sets, indoor and outdoor. Since there were no ground truth measured for the real data, the only error which could be analysed was the noisy reprojection error. This section presents results based on the noisy reprojection error but also focuses on the differences between different methods. However further analysis can be performed on the real data by evaluating the spread of the solutions from several training sets and the difference between solutions on the same training set. This section focuses only on the $L_2$ difference functions and $L_2$-WOR as they proved to have the better performance when the synthetic results were analysed. The real data consisted, as mentioned, of two different sets. The indoor set consisted of 97 subsets each containing 100 correspondences and the outdoor set consisted of 100 subsets each containing 100 correspondences.

5.2.1 Noisy Reprojection Error

The following figures displays the relative improvements over $L_2$-WOR in terms of the noisy reprojection error. However the errors in the figures were not calculated upon the set of points used to generate the each respective result but over the whole set of point correspondences. The choice to use the full set of correspondences was made in order to test the performance of the cost function with a set which was to some degree independent from the training set in terms of errors. Thus the results predicts the real performance to some degree instead of the minimization of the noisy reprojection error in the training set. Figure 5.19 represents the indoor set
and figure 5.20 represents the outdoor set. As mentioned earlier these test only include the difference functions as they had displayed the better performance in the analysis of the synthetic data. The figures includes both types of initialization, SVD and $L_2$-WOR.

![Graph 5.19](image1.png)

Figure 5.19: Mean percentage of correspondence points within the specified noisy reprojection error distance for all solutions of the real indoor dataset.

![Graph 5.20](image2.png)

Figure 5.20: Mean percentage of correspondence points within the specified noisy reprojection error distance for all solutions of the real outdoor dataset.
Since $L_2$-WOR by definition tries to minimize the $L_2$ error and the set of points after outlier rejection all had small errors the characteristics displayed above were expected. The difference in terms of percentage is small, less than 1% at all error distances. In order to further visualize the performance on the testing set, the figures below displays the minimum percentage of points at a certain distance for all generated solutions. Figure 5.21 displays the indoor set and figure 5.22 displays the outdoor set. The figures includes both types of initialization, SVD and $L_2$-WOR.

![Graph](image)

*Figure 5.21: Minimum percentage of correspondence points within the specified noisy reprojection error distance for all solutions of the real indoor dataset.*

In contrast to the figures displaying the mean noisy reprojection error, these figures do display a bigger difference between the cost functions. The figures display that for the outdoor set, the $L_2$-diff $\beta:0,5$ with SVD initialization improved the accuracy of the result while the same function with $\beta:1,0$ had the opposite effect. These figures display that there can be a benefit from using symmetry functions, however they may also generate worse results depending on how the symmetry term is weighted. When analysing the four figures displayed for the real data, one should note that the error measured was the noisy reprojection error. Thus any real projection error benefits for a cost function, which were analysed in the synthetic data results section (5.1), may not translate into noisy reprojection error benefits for the real data. Thus the real data figures should not be the only source relied upon, when evaluating the accuracy of the cost functions.
Noisy Reprojection Error Summary

In this section it was shown that the functions displayed an expected behaviour when analysing the real data. The source used to generate the data was the solutions generated by the cost functions using a small subset of the full dataset and these solutions were later tested on the full dataset. The tables below displays the mean, standard deviation and maximum of the noisy reprojection error for the different algorithms. The error is based on the average error for the correspondence points of the full set. The tables also display the mean percentage of correspondence points within a combined error distance of 0.5, 1.0 and 2.0 respectively. Table 5.15 displays the results for the indoor set and table 5.16 displays the results for the outdoor set.

Table 5.15: Displays the noisy reprojection error for the full indoor set. Also displays the percentage of correspondence points with a combined error distance of certain threshold values.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>sd</th>
<th>max</th>
<th>pts&lt;0.5</th>
<th>pts&lt;1.0</th>
<th>pts&lt;2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_2 )-WOR</td>
<td>0.656</td>
<td>0.012</td>
<td>0.715</td>
<td>87.51%</td>
<td>96.56%</td>
<td>97.45%</td>
</tr>
<tr>
<td>( L_2 )-diff ( \beta ):1.0 (SVD)</td>
<td>0.657</td>
<td>0.013</td>
<td>0.723</td>
<td>87.32%</td>
<td>96.52%</td>
<td>97.44%</td>
</tr>
<tr>
<td>( L_2 )-diff ( \beta ):0.5 (SVD)</td>
<td>0.657</td>
<td>0.012</td>
<td>0.716</td>
<td>87.40%</td>
<td>96.55%</td>
<td>97.45%</td>
</tr>
<tr>
<td>( L_2 )-diff ( \beta ):1.0 (( L_2 )-WOR)</td>
<td>0.658</td>
<td>0.013</td>
<td>0.717</td>
<td>87.23%</td>
<td>96.53%</td>
<td>97.45%</td>
</tr>
<tr>
<td>( L_2 )-diff ( \beta ):0.5 (( L_2 )-WOR)</td>
<td>0.657</td>
<td>0.012</td>
<td>0.716</td>
<td>87.30%</td>
<td>96.54%</td>
<td>97.45%</td>
</tr>
</tbody>
</table>
Table 5.16: Displays the noisy reprojection error for the full outdoor set. Also displays the percentage of correspondence points with a combined error distance of certain threshold values.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>sd</th>
<th>max</th>
<th>pts&lt;0.5</th>
<th>pts&lt;1.0</th>
<th>pts&lt;2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-WOR</td>
<td>1.005</td>
<td>0.022</td>
<td>1.206</td>
<td>76.76%</td>
<td>94.28%</td>
<td>96.59%</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0 (SVD)</td>
<td>1.009</td>
<td>0.025</td>
<td>1.177</td>
<td>76.18%</td>
<td>94.19%</td>
<td>96.59%</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5 (SVD)</td>
<td>1.006</td>
<td>0.019</td>
<td>1.163</td>
<td>76.54%</td>
<td>94.25%</td>
<td>96.59%</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0 ($L_2$)</td>
<td>1.007</td>
<td>0.026</td>
<td>1.237</td>
<td>76.45%</td>
<td>94.22%</td>
<td>96.57%</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5 ($L_2$)</td>
<td>1.006</td>
<td>0.022</td>
<td>1.200</td>
<td>76.54%</td>
<td>94.26%</td>
<td>96.58%</td>
</tr>
</tbody>
</table>

As displayed by the tables and the graphs the difference in terms of noisy reprojection error is very small between the cost functions. What is interesting to note is the relatively worse performance of the outdoor set even though its environment implied higher quality camera images. This difference is due to that the outdoor has just been set up when the data was collected and thus the manual fine tuning of the system had not been done to the same extent as it had been in the indoor system. This difference meant that the correspondence points collected for the outdoor set were worse than those for the indoor set.

### 5.2.2 Solution Spread

The solution spread is a measurement which displays how much the solutions from the same cost function differed from one another between different training sets. The generated solution from each set is typically represented by a rotation matrix and a translation vector. The translation vector spread is calculated by using the distance between translation vectors. Displaying a difference between matrices becomes less intuitive to understand so instead the choice was made to use another notation for the rotation, a normalized rotation vector and an angle. This rotation notation was covered in section 3.1.3. The rotation spread is calculated as the difference between two angles and the distance between two rotation vectors. The spread of the two rotation components are calculated and displayed separately. The following tables display the standard deviation and maximum distance from the mean using the rotation angle $\omega^s$, the normalized rotation vector $r_v^s$ and the translation vector $t^s$. Table 5.17 displays the solution spread for the indoor set and table 5.18 displays the solution spread for the outdoor set.
Table 5.17: Solution spread in terms of angle($\omega^s$), rotation vector($r^s_v$) and translation($t^s$) for the indoor set. $\omega^s$ is specified in degrees. Symmetry function fields with better values than the corresponding $L_2$-WOR field has been marked with a darker background.

<table>
<thead>
<tr>
<th></th>
<th>sd $\omega^s$</th>
<th>max $\omega^s$</th>
<th>sd $r^s_v$</th>
<th>max $r^s_v$</th>
<th>sd $t^s$</th>
<th>max $t^s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>0.98600</td>
<td>3.03152</td>
<td>0.00976</td>
<td>0.02945</td>
<td>0.01639</td>
<td>0.05016</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.13857</td>
<td>0.36198</td>
<td>0.00043</td>
<td>0.00104</td>
<td>0.00163</td>
<td>0.00468</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0 (SVD)</td>
<td>0.14100</td>
<td>0.34411</td>
<td>0.00042</td>
<td>0.00096</td>
<td>0.00166</td>
<td>0.00556</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5 (SVD)</td>
<td>0.13353</td>
<td>0.33364</td>
<td>0.00042</td>
<td>0.00103</td>
<td>0.00157</td>
<td>0.00494</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0 ($L_2$)</td>
<td>0.14310</td>
<td>0.36684</td>
<td>0.00045</td>
<td>0.00118</td>
<td>0.00169</td>
<td>0.00475</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5 ($L_2$)</td>
<td>0.14070</td>
<td>0.36419</td>
<td>0.00043</td>
<td>0.00104</td>
<td>0.00166</td>
<td>0.00482</td>
</tr>
</tbody>
</table>

Table 5.18: Solution spread in terms of angle($\omega^s$), rotation vector($r^s_v$) and translation($t^s$) for the outdoor set. $\omega^s$ is specified in degrees. Symmetry function fields with better values than the corresponding $L_2$-WOR field has been marked with a darker background.

<table>
<thead>
<tr>
<th></th>
<th>sd $\omega^s$</th>
<th>max $\omega^s$</th>
<th>sd $r^s_v$</th>
<th>max $r^s_v$</th>
<th>sd $t^s$</th>
<th>max $t^s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>1.16642</td>
<td>3.36895</td>
<td>0.02937</td>
<td>0.10279</td>
<td>0.05432</td>
<td>0.17499</td>
</tr>
<tr>
<td>$L_2$-WOR</td>
<td>0.20846</td>
<td>0.39819</td>
<td>0.00105</td>
<td>0.00632</td>
<td>0.00154</td>
<td>0.00374</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0 (SVD)</td>
<td>0.19820</td>
<td>0.45522</td>
<td>0.00104</td>
<td>0.00552</td>
<td>0.00176</td>
<td>0.00692</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5 (SVD)</td>
<td>0.19243</td>
<td>0.45163</td>
<td>0.00098</td>
<td>0.00550</td>
<td>0.00158</td>
<td>0.00388</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:1.0 ($L_2$)</td>
<td>0.22047</td>
<td>0.42948</td>
<td>0.00109</td>
<td>0.00695</td>
<td>0.00158</td>
<td>0.00373</td>
</tr>
<tr>
<td>$L_2$-diff $\beta$:0.5 ($L_2$)</td>
<td>0.20634</td>
<td>0.41349</td>
<td>0.00103</td>
<td>0.00618</td>
<td>0.00154</td>
<td>0.00353</td>
</tr>
</tbody>
</table>

The results from the solution spread display a consistency between the solutions generated for different training sets and for the different cost functions. There is a slight tendency in the data where high $\beta$ values increases the solution spread. From analysing the difference functions with a $\beta$ of 0.5 there also seem to be slight benefit to choosing SVD as the initialization point. As can also be seen from the data using any of the cost functions compared using the SVD-method translates into a great improvement of the solution spread.

5.2.3 Solution Difference

The solution difference is similar to the solution spread but instead of using the distance within a cost function this measurement uses the distance between solutions from different cost functions using the same training set. The results was, as in the solution spread measurement, divided into a rotation angle, a normalized rotation vector and a translation vector. The following tables display the mean, standard deviation and maximum difference between the different algorithms using the rotation angle $\omega^d$, the normalized rotation vector $r^d_v$ and the translation vector $t^d$. Due to space constrictions of the table the names of the cost functions were shortened. The details of the cost function has been put as superscript and subscript (e.g.
Diff $L_2$ translates into $L_2$ difference with a $\beta$ of 0.5 and initialized by the $L_2$-WOR method). Table 5.19 displays the solution difference for the indoor set and table 5.19 displays the solution difference for the outdoor set.

**Table 5.19: Solution difference between the two specified cost functions in terms of angle($\omega^*$), rotation vector($r_v^*$) and translation($t_s^*$) for the indoor set. $\omega^*$ is specified in degrees. Symmetry function difference fields where the value is less than one of the differences with $L_2$-WOR and using the same $\beta$ has been marked with a darker background.**

<table>
<thead>
<tr>
<th>Function</th>
<th>mean $\omega^d$</th>
<th>max $\omega^d$</th>
<th>mean $r_v^d$</th>
<th>max $r_v^d$</th>
<th>mean $t_s^d$</th>
<th>max $t_s^d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-WOR - diff$^{L_2}$</td>
<td>0.02443</td>
<td>0.07702</td>
<td>0.00013</td>
<td>0.00067</td>
<td>0.00032</td>
<td>0.00270</td>
</tr>
<tr>
<td>$L_2$-WOR - SVD</td>
<td>0.01069</td>
<td>0.07482</td>
<td>0.00009</td>
<td>0.00061</td>
<td>0.00015</td>
<td>0.00168</td>
</tr>
<tr>
<td>diff$^{SVD}$ - diff$^{L_2}$</td>
<td>0.02022</td>
<td>0.25447</td>
<td>0.00008</td>
<td>0.00069</td>
<td>0.00025</td>
<td>0.0032</td>
</tr>
<tr>
<td>$L_2$-WOR - SVD</td>
<td>0.03985</td>
<td>0.21279</td>
<td>0.00016</td>
<td>0.00068</td>
<td>0.00048</td>
<td>0.00272</td>
</tr>
<tr>
<td>$L_2$-WOR - SVD</td>
<td>0.01257</td>
<td>0.06975</td>
<td>0.00009</td>
<td>0.00073</td>
<td>0.00016</td>
<td>0.00160</td>
</tr>
<tr>
<td>diff$^{SVD}$ - diff$^{L_2}$</td>
<td>0.03658</td>
<td>0.24655</td>
<td>0.00013</td>
<td>0.00084</td>
<td>0.00044</td>
<td>0.00337</td>
</tr>
</tbody>
</table>

**Table 5.20: Solution difference between the two specified cost functions in terms of angle($\omega^*$), rotation vector($r_v^*$) and translation($t_s^*$) for the outdoor set. $\omega^*$ is specified in degrees. Symmetry function difference fields where the value is less than one of the differences with $L_2$-WOR and using the same $\beta$ has been marked with a darker background.**

<table>
<thead>
<tr>
<th>Function</th>
<th>mean $\omega^d$</th>
<th>max $\omega^d$</th>
<th>mean $r_v^d$</th>
<th>max $r_v^d$</th>
<th>mean $t_s^d$</th>
<th>max $t_s^d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2$-WOR - diff$^{L_2}$</td>
<td>0.03304</td>
<td>0.19456</td>
<td>0.00025</td>
<td>0.00107</td>
<td>0.00040</td>
<td>0.00235</td>
</tr>
<tr>
<td>$L_2$-WOR - SVD</td>
<td>0.01751</td>
<td>0.17077</td>
<td>0.00014</td>
<td>0.00099</td>
<td>0.00020</td>
<td>0.00224</td>
</tr>
<tr>
<td>diff$^{SVD}$ - diff$^{L_2}$</td>
<td>0.02373</td>
<td>0.13248</td>
<td>0.00016</td>
<td>0.00080</td>
<td>0.00028</td>
<td>0.00174</td>
</tr>
<tr>
<td>$L_2$-WOR - SVD</td>
<td>0.04745</td>
<td>0.28587</td>
<td>0.00030</td>
<td>0.00218</td>
<td>0.00060</td>
<td>0.00595</td>
</tr>
<tr>
<td>$L_2$-WOR - SVD</td>
<td>0.01707</td>
<td>0.12272</td>
<td>0.00013</td>
<td>0.00084</td>
<td>0.00017</td>
<td>0.00171</td>
</tr>
<tr>
<td>diff$^{SVD}$ - diff$^{L_2}$</td>
<td>0.04821</td>
<td>0.37882</td>
<td>0.00026</td>
<td>0.00220</td>
<td>0.00055</td>
<td>0.00597</td>
</tr>
</tbody>
</table>

As was expected the resulting solutions from difference functions when using $L_2$-WOR as initialization point were closer to the solutions generated by using only $L_2$-WOR than the difference functions using SVD as initialization point. An interesting discovery from the table above is that the $\beta$ value did only affect the result difference when using SVD as initialization point and not when using $L_2$-WOR as initialization point. Another interesting finding is that the result difference was typically less than the typical result spread for the cost functions. This indicates that, in the case of these datasets, the accuracy of the solutions generated were more related with the accuracy of the data than the performance of the cost functions.
5.2.4 Comparison with the company’s old method

The old method used by the company, which was also used to generate the real data points, was based on manual input and distance measurements in order to calibrate the system and triangulate point correspondences. This section will give a brief overview on how the company’s system worked. The section will also compare the results of the old system with the results of $L_2$ difference and $L_2$-WOR in order to display how an adoption of $L_2$ difference and $L_2$-WOR could improve the accuracy of the system. This section will only cover the performance of the indoor set.

In the old system a user had to use software to point out a specific set of correspondences and measure the distance to those points. In total there were three different points which were to be registered in both the left and the right camera and the distance from the left camera to those points did also have to be registered. Each of the three points had a different purpose and different restrictions. The names of the points are listed below.

- Horizon point
- Distant point
- Close point

The horizon point were to be a point as far away as possible and at the same height as the height at which the cameras were mounted. Ideally the horizon 3d point is at an infinite distance in order for the cameras to have a common point which can be used when calibrating the system. The distant point and the close point are simply two points which were separated in Z distance, were the distant point was to be further away and the close point was to be closer to the cameras.

The calibration process made some assumptions about the general camera configuration. The characteristics of the cameras had to, as was explained previously, be known to the system. The locations of the cameras was also limited to $z = 0$ plane in the world coordinates. The rotation of these cameras in relation to this plane was restricted to only have a big rotation in terms of yaw, whereas the rotation in terms of pitch and roll were to be small. The system had support for cameras mounted at different height but this height had to be known to the system. The cameras were separated horizontally on the $z = 0$ plane but the horizontal distance also had to be known to the system.

The system was first corrected in terms of roll. This process was done by manual estimation based on the images from the cameras and corrections afterwards when evaluating the errors. After that point the system had an automated process to estimate the pitch and yaw between the camera and the $z = 0$ plane. The pitch was estimated by calculating which pitch set the horizon point y-coordinate to 0 in the left camera and to the height difference in the right camera. Finally the yaw
was estimated by calculating which yaw set the horizon point x-coordinate to 0 in the left camera and the right camera. To summarize the rotation correction were applied as follows.

1. Roll correction
2. Pitch correction
3. Yaw correction

The z-coordinate of the close and distant point could be easily calculated from the left camera using the measured distance. In order to triangulate future point correspondences the system established a linear relationship between the world x-coordinate on the camera planes (adjusted for translation) and the z-coordinate of the point in the world.

In order to compare the new system with the old system, rotation matrices were calculated for the right and the left camera based upon the yaw pitch and roll which the system produced. The rotation (yaw, pitch and roll) and camera centre position ($c_x$, $c_y$ and $c_z$) are presented below in table 5.21.

Table 5.21: Rotation and translation values generated using the company’s old method for the outdoor dataset. Rotation values are displayed in degrees.

<table>
<thead>
<tr>
<th></th>
<th>yaw</th>
<th>pitch</th>
<th>roll</th>
<th>$c_x$</th>
<th>$c_y$</th>
<th>$c_z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Camera</td>
<td>-33,541</td>
<td>-0,900</td>
<td>0,000</td>
<td>0,000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>Right Camera</td>
<td>38,042</td>
<td>-2,100</td>
<td>1,400</td>
<td>26,800</td>
<td>-0,173</td>
<td>0,000</td>
</tr>
</tbody>
</table>

The solution from the old method was then converted to a left camera oriented world and a projection matrix to represent the relative motion of the right camera. The mathematical conversion is presented below.

$$R^{left} = (R^{left}_{yaw} R^{left}_{pitch} R^{left}_{roll})^T$$

$$R^{right} = (R^{right}_{yaw} R^{right}_{pitch} R^{right}_{roll})^T$$

$$R = R^{right}(R^{left})^T$$

$$T = -R^{right}[c_x c_y c_z]^T$$

When the rotation matrices had been calculated, $L_2$ triangulation (covered in section 3.6.2) was performed for the first real dataset. The errors from the reprojection were later compared to the errors from the $L_2$-WOR and $L_2$ difference cost functions. In order to compare the accuracy of the old method, the solution was compared both in terms of how much the solution differed from the mean solution.
of the other cost functions and how the noisy reprojection error compared with the noisy reprojection error of the other methods. Table 5.22 displays the difference between the solution from the old method and the mean solution from the other cost functions. Table 5.23 displays the noisy reprojection error and the mean percentage of points within some fixed error distances.

Table 5.22: Solution difference of the old method compared to the mean solutions generated by the cost functions. The solutions were generated from the real indoor dataset.

<table>
<thead>
<tr>
<th></th>
<th>( \omega^d )</th>
<th>( r^d_w )</th>
<th>( t^d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_2^{WOR} )</td>
<td>0.08724</td>
<td>0.05882</td>
<td>0.04668</td>
</tr>
<tr>
<td>( \text{diff}_{0.5}^{SVD} )</td>
<td>0.08490</td>
<td>0.05882</td>
<td>0.04674</td>
</tr>
<tr>
<td>( \text{diff}_{0.5}^{L_2} )</td>
<td>0.08897</td>
<td>0.05883</td>
<td>0.04667</td>
</tr>
</tbody>
</table>

Table 5.23: Noisy reprojection error of the solutions generated by the old method and the cost functions. The solutions were generated from the real indoor dataset.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>sd</th>
<th>max</th>
<th>pts&lt;0.5</th>
<th>pts&lt;1.0</th>
<th>pts&lt;2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Method</td>
<td>6.94787</td>
<td>-</td>
<td>6.94787</td>
<td>5.67%</td>
<td>11.91%</td>
<td>23.76%</td>
</tr>
<tr>
<td>( L_2^{WOR} )</td>
<td>0.65564</td>
<td>0.01165</td>
<td>0.71543</td>
<td>87.51%</td>
<td>96.56%</td>
<td>97.45%</td>
</tr>
<tr>
<td>( \text{diff}_{0.5}^{SVD} )</td>
<td>0.65664</td>
<td>0.01160</td>
<td>0.71642</td>
<td>87.40%</td>
<td>96.55%</td>
<td>97.45%</td>
</tr>
<tr>
<td>( \text{diff}_{0.5}^{L_2} )</td>
<td>0.65729</td>
<td>0.01212</td>
<td>0.71605</td>
<td>87.30%</td>
<td>96.54%</td>
<td>97.45%</td>
</tr>
</tbody>
</table>

As can be seen in the data there is a difference between using the old method and using the cost functions, especially in terms of the noisy reprojection error. It should however be noted that the cost functions are constructed to minimize an error closely resembling the noisy reprojection error and thus the real projection error might provide slightly different results. It should however be clear from these number that changing from the old method to cost functions would be beneficial for the system in terms of accuracy.

## 5.3 Symmetry Tests

During the project, results regarding the symmetry were also extracted. The symmetry error is the same representation of the symmetry error used when calculating the error for the \( L_2 \) difference functions which was covered in section 4.5.3. The only difference between the \( L_2 \) difference error and the symmetry error is the removal of the \( L_2 \) term. The symmetry error is displayed as three different components, the symmetry error for the x component, the symmetry error for the y component and the combined symmetry error for the x and y component. Table 5.24 displays the symmetry error of the indoor set and table 5.25 displays the symmetry error for the outdoor set.
Table 5.24: The mean and max symmetry error for the indoor set, where x represents the symmetry error for the x component and y represent the symmetry error for the y component.

<table>
<thead>
<tr>
<th></th>
<th>mean x</th>
<th>max x</th>
<th>mean y</th>
<th>max y</th>
<th>mean x+y</th>
<th>max x+y</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>0.07252</td>
<td>0.38413</td>
<td>1.04179</td>
<td>2.00000</td>
<td>1.11431</td>
<td>2.15988</td>
</tr>
<tr>
<td>$L^2_{WOR}$</td>
<td>0.00025</td>
<td>0.00403</td>
<td>0.00456</td>
<td>0.02073</td>
<td>0.00481</td>
<td>0.02116</td>
</tr>
<tr>
<td>diff$^SVD_{1,0}$</td>
<td>0.00029</td>
<td>0.00271</td>
<td>0.00040</td>
<td>0.00414</td>
<td>0.00069</td>
<td>0.00658</td>
</tr>
<tr>
<td>diff$^SVD_{0.5}$</td>
<td>0.00028</td>
<td>0.00303</td>
<td>0.00040</td>
<td>0.00423</td>
<td>0.00069</td>
<td>0.00626</td>
</tr>
<tr>
<td>diff$L^2_{1.0}$</td>
<td>0.00022</td>
<td>0.00284</td>
<td>0.00034</td>
<td>0.00428</td>
<td>0.00056</td>
<td>0.00433</td>
</tr>
<tr>
<td>diff$L^2_{0.5}$</td>
<td>0.00024</td>
<td>0.00335</td>
<td>0.00040</td>
<td>0.00399</td>
<td>0.00064</td>
<td>0.00402</td>
</tr>
</tbody>
</table>

Table 5.25: The mean and max symmetry error for the outdoor set, where x represents the symmetry error for the x component and y represent the symmetry error for the y component.

<table>
<thead>
<tr>
<th></th>
<th>mean x</th>
<th>max x</th>
<th>mean y</th>
<th>max y</th>
<th>mean x+y</th>
<th>max x+y</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>0.45569</td>
<td>1.70643</td>
<td>1.69912</td>
<td>2.00000</td>
<td>2.15481</td>
<td>3.70643</td>
</tr>
<tr>
<td>$L^2_{WOR}$</td>
<td>0.00016</td>
<td>0.00067</td>
<td>0.00645</td>
<td>0.02367</td>
<td>0.00661</td>
<td>0.02405</td>
</tr>
<tr>
<td>diff$^SVD_{1,0}$</td>
<td>0.00017</td>
<td>0.00118</td>
<td>0.00047</td>
<td>0.00558</td>
<td>0.00065</td>
<td>0.00626</td>
</tr>
<tr>
<td>diff$^SVD_{0.5}$</td>
<td>0.00015</td>
<td>0.00061</td>
<td>0.00054</td>
<td>0.00593</td>
<td>0.00069</td>
<td>0.00631</td>
</tr>
<tr>
<td>diff$L^2_{1.0}$</td>
<td>0.00015</td>
<td>0.00054</td>
<td>0.00036</td>
<td>0.00240</td>
<td>0.00051</td>
<td>0.00276</td>
</tr>
<tr>
<td>diff$L^2_{0.5}$</td>
<td>0.00015</td>
<td>0.00058</td>
<td>0.00053</td>
<td>0.00373</td>
<td>0.00068</td>
<td>0.00397</td>
</tr>
</tbody>
</table>

From the data it was directly obvious that the symmetry error was greater in the y component than the x component. There can be many reasons for this problem and some of the potential sources of this issue is covered in the discussion. As for the performance of the different functions, the results were expected. The symmetry functions does improve the symmetry error from SVD and $L_2$-WOR. What is interesting however is the small difference in symmetry error between the different symmetry cost functions, even though the $\beta : 1,0$ places double the emphasis on reducing the symmetry error than its $\beta : 0,5$ counterpart. Additionally the indoor set showed a smaller improvement in terms of symmetry error than the outdoor set suggesting a closer correlation between the $L_2$ solutions and the symmetry function solutions in the indoor set.
Chapter 6

Discussion

The performance of the proposed cost functions were evaluated based on the real datasets and the synthetic datasets. The synthetic datasets were constructed to simulate the environment of the projectile tracking system used by the company. The analysis of the results displayed the properties of the cost functions but there were also many limitations to the datasets and the test overall which will be discussed in this chapter.

6.1 Limitations of the Test

The test was limited due to time constraints. A more extensive test could have included more random test variables. In this test camera intrinsics, 3D-coordinates and relative motion was kept constant. The decision to keep those variables constant was made in order to isolate the uncertainties to the random noise applied to the correspondence coordinates. A more extensive test could thus increase randomization to the other variables and analyse how the change of those variables would have changed the result.

The noise variables used in this test were kept constant with the exception of the noise type. The choice of sigma, outlier percentage and similar parameters was chosen to mimic the environment found in the real projectile tracking system. A more extensive test could have analysed how increases in any of the noise parameters would have affected the performance of the cost functions.

The cost functions tested were also limited. The beta value for the difference functions were of only two distinct values, 0.5 and 1.0. A more extensive test of the two difference function types, could have included the evaluation of more beta values.
6.2 Limitations of the Scope

The scope of the evaluation was limited to local iterative methods. Many other methods, in particular more modern SVD-methods, bundle adjustment methods or object space error methods were not evaluated. Additionally there are more cost functions which were not evaluated, in particular $L_1$. The major limitations of using local iterative methods is that the global minimum cannot be guaranteed. Any guarantee of global minimum would require branch and bound methods. However good bounding functions are difficult if not impossible to create depending on the cost function.

Another aspect which was not taken into account was the point distribution. In many structure from motion methods it is common to put importance on a spread of correspondence points in the image. In this project the spread was assumed to be good and the spread of correspondence points were not taken into account when calculating the error or when selecting which correspondence points to use. The only aspect related to the correspondence point spread which was taken into account was that the points used for each training set in the real data were spread evenly in the series of points in the full set, thus reducing any potential dependencies between the point correspondences.

As a final remark, the scope was also limited by what initialization techniques were used. The evaluation performed did only use simple SVD initializations and initializations based on other cost functions. A more thorough evaluation would also include initializations from newer SVD-methods, $L_\infty$ minimization and object space error minimization. This is especially important due to the solution spaces of the cost functions tested contained several local minima in close proximity.

6.3 Extension Possibilities

The project was based around the projectile tracking system at the company and thus focused on a stereo camera system with a limited rotation and translation solution space. However the methods which were presented can be extended to cover a wider scope of problems, with multiple cameras and unbounded rotation. However such changes would require some adjustments.

An increase of the rotation space would make the methods presented more general. The rotation space can be increased, but would necessitate a change of the rotation notation in order to not include singularities. Alternative rotation notations which may be used are a rotation vector and a rotation angle as used in $L_\infty$ minimization [13] or quaternions.
An increase of cameras is the another highly relevant extension of the methods presented. The only problem with increasing the number of cameras in the methods presented is the triangulation method used. The triangulation method used is only applicable for stereo triangulation. There are two solutions to this problem. Either use a bundle adjustment method and thus eliminate the need for a triangulation method or use another triangulation method such as linear, $L_2$ or $L_\infty$ for multiple cameras.

In the analysis of the tests, the spread of the data in the camera images was always assumed to be good. When looking at individual sets however, there were some training sets which had data with most of the correspondence points in a smaller area of the image. These set typically generated solutions in which the noisy reprojection error’s direction depended heavily on the location in the image and the dependency was typically linear. In order to extend the method’s reliability, the use of a weighting matrix could be incorporated in order to put higher emphasis on points whose location on the camera plane has a lower density of correspondence points. However one should take into consideration that a correspondence point in an area which is sparse of other correspondence points may be indicative of an outlier.

6.4 Accuracy Using Real Data

In the real data section (5.2) several metrics were presented to analyse the accuracy of the different cost functions. As real data only used the noisy reprojection error, the difference between the functions in terms of that error can only give an indication of how accurate the function is. The case of using symmetry functions does also get more complicated when using real data. The accuracy of the result from the symmetry functions are dependent on the errors being symmetrical. To better determine the relative accuracy of the cost functions on real data is to analyse the solutions which were generated. This analysis was done in the section 5.2.2 and 5.2.3 and it showed a small benefit of using $L_2$ difference with a $\beta$ of 0.5 which is well aligned with the synthetic results analysing the real projection error. However it is important to note that this test can only validate the use of this method on this camera configuration. In other configurations the error may not be symmetrical and another cost function may be better, especially one must be careful to not include dependencies between the correspondence points when using the symmetry cost functions as this might reduce their accuracy.
Chapter 7

Conclusions

The tests conducted displayed the viability of using $L_2$ difference functions in place of the $L_2$ cost function when estimating camera motion. In particular the $L_2$ difference function with a beta of 0.5 provided the best results with a closer estimation of the relative motion than any other function over all the different error models. The variance difference function performed well but had a lower level of convergence and thus needed far better initialization than the $L_2$ difference functions. The existence of several local minima was a common problem for the difference functions in general and they were only able to pinpoint the solution to an area. However the $L_2$ difference function had a small enough area to outperform $L_2$ with outlier rejection which is commonly used in SfM.

The big drawback of using the difference cost functions is the speed of the relative motion estimation. Standard least square functions does have many possible optimizations which are not feasible for general cost functions which is why $L_2$ is commonly used in SfM. This does imply that the use of these cost functions for real time applications is limited if not impossible with current hardware. However for a camera system in which the relative motion is constant, the cost functions can be used to estimate motion once or infrequently as was the case with the company’s projectile tracking system.

The real data provided further evidence of good performance of the cost function, $L_2$ difference with a beta of 0.5, and the cost function were thus found viable for use in real scenarios. The function also proved to be competitive or better than its $L_2$ counterpart when looking at the solution spread for the real data. However the use of the functions requires independent point correspondences and symmetrical errors, as detailed in the discussion.

All the different results presented show that symmetry functions in general and $L_2$ difference with a beta of 0.5 in particular can be used instead of $L_2$-WOR and the user can expect a slight increase in accuracy of the system. The only drawback
of using the symmetry functions is that there are no optimized algorithms and an increase in the amount of correspondence points would increase this drawback. Thus a good use case for symmetry functions in the current stage, is when only a small set of point correspondences are used. In this project the main goal was to create an algorithm which could create a good solution from a small set of correspondence points and thus the symmetry cost functions are viable both in terms of accuracy and performance.

The system was implemented using C++ and the Eigen library for future use in the company’s projectile tracking system. The system only used a single thread but can be improved in terms of performance by using multiple thread since each element of the Hessian and the gradient can be calculated independently. This would increase the performance since each element requires stereo triangulation and the stereo triangulation is the major source of time consumption in the algorithm.

As was seen in the section about the company’s old method of relative motion estimation, the company has much to gain from switching to either the proposed method or the standard $L_2$ method. The change of method would increase the accuracy of the systems results while at the same time reduce its dependency on human input. It should be noted that the initial correspondence points do still have to be generated. These points can be generated using two different methods.

- Using automatic feature point detection (e.g. SIFT, SURF or BRISK)
- Using the old method to generate the initial correspondence set

These options have different advantages. Using feature point detection would completely automate the process but the images are usually limited in terms of distinct features and many features are similar to one another. The second option would still require human input, however the reliance of such input would be limited to generating the first correspondence set, thus allowing for a higher error tolerance.

The new method would also allow for continuous improvement of the relative motion estimation by building a bigger and more reliable set of correspondence points as the system is used. Additionally such an iterative approach could allow for automatic adjustment of the relative motion estimation if the relative motion of the system was slightly changed (e.g. due to a hit from a projectile or human interference).

7.1 Future Research

This project did only give initial insights on the accuracy and performance of difference functions and many of the limitations and possible extensions of the project was mentioned in the discussion. The possibilities for future research are listed below.
• Increasing the number of random elements in order ensure accuracy over a wider set of problems.
• A more thorough investigation of the beta value and how it affects accuracy.
• Experiment on using the cost functions in a bundle adjustment method.
• The use of different initialization points, e.g. $L_\infty$ or object space error minimization.
• Testing an algorithm based on $L_2$ and the symmetry of the variance

Symmetry functions are a new way of representing the error in structure from motion. Previously $L_2$ errors have been used since the process allows for accelerated algorithms. However as hardware performance continues to improve, the field of Computer Vision should look to adding more details to the cost functions. More relevant details can make the systems more robust and accurate by constraining more features of the observed error.
Bibliography


Appendix A

Formulas

The derivative of the $L_2$ cost function used in Hartley and Sturm’s triangulation method separated into powers of $t$.

\[ r(t) = \]
\[ t^6 \left( abc^2 f^4 - a^2 cd f^4 \right) + \]
\[ t^5 \left( a^4 + 2a^2 c^2 f'^2 - a^2 d^2 f^4 + b^2 c^2 f^4 + c^4 f'^4 \right) + \]
\[ t^4 \left( 4a^3 b - 2a^2 cd f^2 + 4a^2 cd f'^2 + 2abc^2 f^2 + 4abc^2 f'^2 - abd^2 f^4 + b^2 d f^4 + 4c^3 d f'^4 \right) + \]
\[ t^3 \left( 6a^2 b^2 - 2a^2 d^2 f^2 + 2a^2 d^2 f'^2 + 8abc d f'^2 + 2b^2 c^2 f^2 + 2b^2 c^2 f'^2 + 6c^2 d f'^4 \right) + \]
\[ t^2 \left( -a^2 cd + 4ab^3 + abc^2 - 2abd^2 f^2 + 4abd^2 f'^2 + 2b^2 cd f^2 + 4b^2 cd f'^2 + 4cd^3 f'^4 \right) + \]
\[ t^1 \left( -a^2 d^2 + b^4 + b^2 c^2 + 2b^2 d^2 f'^2 + d^4 f'^4 \right) + \]
\[ t^0 \left( -ab d^2 + b^2 cd \right) \]
Appendix B

Additional Figures

Figure B.1: Minimum percentage of points within the specified error distance using Gaussian testing set. The error type is the noisy reprojection error.
Figure B.2: Minimum percentage of points within the specified error distance using Rayleigh testing set. The error type is the noisy reprojection error.

Figure B.3: Minimum percentage of points within the specified error distance using uniform testing set. The error type is the noisy reprojection error.
Figure B.4: Minimum percentage of points within the specified error distance using independently uniform testing set. The error type is the noisy reprojection error.

Figure B.5: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the Gaussian testing set. The error type is the real projection error and SVD was used as initialization.
Figure B.6: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the rayleigh testing set. The error type is the real projection error and SVD was used as initialization.

Figure B.7: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the uniform testing set. The error type is the real projection error and SVD was used as initialization.
Figure B.8: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the independently uniform testing set. The error type is the real projection error and SVD was used as initialization.

Figure B.9: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the Gaussian testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.
Figure B.10: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the rayleigh testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.

Figure B.11: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the uniform testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.
Figure B.12: Percentage points difference relative to $L_2$-WOR where the measured percentage is the minimum percentage of points within the specified error distance for the independently uniform testing set. The error type is the real projection error and $L_2$-WOR was used as initialization.