Generalized Weiszfeld Algorithms for Lq Optimization

Khurrum Aftab

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

October 2014

Declaration

This thesis is an account of research undertaken between July 2010 and May 2014 at the College of Engineering and Computer Science, The Australian National University, Canberra, Australia.

Except where otherwise indicated, this thesis is my own original work. The material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

The contents of this thesis are mainly extracted from the following papers:

- Khurrum Aftab, Richard Hartley, and Jochen Trumpf, "Generalized Weiszfeld algorithm for Lq optimization", submitted to the Pattern Analysis and Machince Intelligence (PAMI).
- Khurrum Aftab, Richard Hartley, and Jochen Trumpf, "Lq closest point to affine subspaces using the Generalized Weiszfeld algorithm", submitted to the International Journal of Computer Vision (IJCV).
- Khurrum Aftab, Richard Hartley, and Jochen Trumpf, "Lq-bundle adjustment", journal paper in preparation.
- Khurrum Aftab and Richard Hartley, "Lq averaging for Symmetric Positive-Definite matrices", International Conference on Digital Image Computing Techniques and Applications (DICTA), 2013.
- Richard Hartley, Khurrum Aftab, and Jochen Trumpf, "L1 rotation averaging using the Weiszfeld algorithm", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2011. (30% of my contribution)

Supervisory panel:

Prof. Richard Hartley	The Australian National University
Dr. Jochen Trumpf	The Australian National University
Dr. Hongdong Li	The Australian National University

Khurrum Aftab
19 May 2014

to my parents, Aftab and Gulshan, for their endless love...

Acknowledgments

No work exists in isolation and this thesis is by no means an exception. This thesis would not have been possible without the contributions and support of many people. It is impossible to list all the people who have influenced and helped me through the years, but you should all know that I am very grateful for your help, support, encouragement and love.

First and foremost, I like to thank my supervisor Prof. Richard Hartley for introducing me to the mysteries of computer vision and for giving me the pleasure and opportunity to work with him. The joy and enthusiasm he has for his research was motivational for me, even during tough times in the Ph.D. pursuit. I am very grateful that he has spent so much time with me discussing different problems ranging from philosophical issues down to minute technical details. I particularly thank him for his dedication and support throughout my PhD candidature, his insightful advice, and his valuable assistance in putting this thesis together. It has been a privilege to have a supervisor who always keeps the door open and is eagerly willing to discuss the work.

I also like to thank Dr. Jochen Trumpf, my advisor, for his guidance, compassion, encouragement, and collaboration. Jochen's boundless creativity has been a constant source of inspiration, and has had a huge impact on how I approach problems. I appreciate all his contributions of time, and ideas to make my Ph.D. experience productive and stimulating.

This work was sponsored by the National ICT Australia (NICTA), which is gratefully acknowledged. Thanks for providing a good research environment.

I would like to take this opportunity to thank all those who have contributed to this thesis, directly or indirectly. First of all, I would like to thank all the members of the Computer Vision laboratory, for providing a friendly atmosphere, a stimulating research environment, for sharing their ideas with me, and for being good friends. In particular, I like to acknowledge Behrooz, Adnan, Ahmed, Usman, Zeeshan, Samunda, Cong, Muhammad, and Sarah. A special thanks to Behrooz for being a dear friend and an awesome officemate. I wish to thank my friends Usman, Khurram, Nabeel, Sajjad, Inam and Bilal for being present in my life, all the time. And last but not least my university house friends Umbu Raya and Tamara for helping me to kill my spare time and for making my stay at the university house memorable and pleasant.

Finally, I would like to thank my family for all their love and encouragement. I want to express my deepest gratitude towards my parents who raised me with love and supported me in all my pursuits. A special thanks to my brothers, Faisal and Qaisar, and my sister for their love and for being a tremendous source of strength for me. I am pretty sure that, without their support, this thesis would not have been possible.

Thank you
Khurrum Aftab

Abstract

In many computer vision applications, a desired model of some type is computed by minimizing a cost function based on several measurements. Typically, one may compute the model that minimizes the L2 cost, that is the sum of squares of measurement errors with respect to the model. The simplicity of L2 methods is well appreciated, but they suffer from a drawback when it comes to robustness against outliers; even a single outlier may change the L2 solution drastically. This thesis is primarily focused on the development of simple and robust Lq optimization methods, for q ranging from 1 to 2 (excluding 2), for problems in the area of computer vision. The proposed Lq optimization methods minimize the sum of the q-th power of errors and give more robust results than least squares methods in the presence of outliers. We particularly consider two classes of problems: Firstly, Lq-closest-point problems, where we seek a point for which the sum of the q-th power of distances from a given set of measurements is the minimum. Secondly, Lq non-linear parameter estimation problems, where parameters of a model are estimated by minimizing the sum of the q-th power of distances to a given set of points.

The proposed Lq-closest-point algorithms are inspired by the Weiszfeld algorithm, a classic algorithm for finding the L1 mean of a set of points in a Euclidean space. The proposed Lq-closest-point algorithms inherit all the properties of the Weiszfeld algorithm, such as provable convergence to the global Lq minimum, analytical updates, simple to understand and easy to code. We specifically propose the following algorithms: First of all, we propose a generalization of the Weiszfeld algorithm to find the Lq mean of a set of points in a Euclidean space. We refer to this as the Lq Weiszfeld algorithm. We then propose a generalization of the Lq Weiszfeld algorithm to find the Lq mean of a set of points on a Riemannian manifold of non-negative sectional curvature and apply it to rotation averaging and Symmetric Positive-Definite matrices averaging problems. In addition to the proof of convergence, we relax the bounds on maximum distance between points on manifold to ensure convergence. Furthermore, we propose a generalization of the Lq Weiszfeld algorithm to find the Lq-closest-point to a set of affine subspaces, possibly of different dimensions, in a Euclidean space and apply it to the triangulation problem.

In addition to the closest-point problems, we propose an algorithm to find an Lq solution of a non-linear parameter estimation problem, specifically the bundle adjustment problem. An advantage of the proposed algorithm is that an efficient least squares optimization method, namely the Levenberg-Marquardt method, is used to find a robust solution to the problem, even an Lq solution.

In all the cases, our experimental results confirm the fact that in the presence of outliers the proposed Lq algorithms give superior results than least squares algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we propose several methods to minimize a more robust cost function than a least squares cost function for different types of target functions used in the computer vision applications. Specifically, we are interested in the minimization of an L_q cost function, for $1 \leq q < 2$, where the sum of the q-th power of errors or distances is minimized. It is known that L_q cost functions, especially for $q = 1$, are more robust to outliers than L_2 or least squares cost functions that minimize the sum of squared errors. Generally, L_q optimization methods use relatively complex optimization strategies compared to least squares methods. Nonetheless, the proposed algorithms make a trade-off between simplicity and robustness by iteratively minimizing a weighted L_2 cost function to find the more robust L_q solution. As a result, the proposed methods are simple to understand, easy to formulate and do not require any new optimization strategy. Thus, the proposed L_q algorithms involve a simple iterative extension of least squares methods. Furthermore, from an implementation point of view an existing least squares implementation can be modified to give a more robust L_q solution.

In the following section we will start by discussing the two classes of problems that are solved in thesis followed by a discussion on challenges in solving L_q cost functions, in section 1.2. In section 1.3 and section 1.4 we will give an overview of the L_q-closest-point problems and L_q parameter estimation problems, respectively, in detail. In section 1.5 we summarize the contributions of this thesis followed by the thesis outline, in section 1.6.

1.1 Distance Minimization Problems

In this thesis, we focus on two basic classes of distance minimization problems, that are L_q-closest-point (or L_q averaging) problems and L_q non-linear parameter estimation problems. Given a set of measurements, such as points, lines, subspaces, or their mixture, the L_q-closest-point problem is to find a point for which the sum of the q-th power of distances to the given measurements is the minimum. On the other hand, the L_q non-linear parameter estimation problem is to find a model that best describes a given set of data points by estimating model parameters via minimizing the sum of the q-th power of distances to all the given points. Clearly, both the problems have been extensively studied in the literature, but our objective is to propose methods that are simple to understand and easy to implement. Below we discuss these problems in detail.

1.1.1 L_q Averaging or L_q-Closest-Point Problems

The L_q averaging or L_q-closest-point algorithm finds the L_q mean or the L_q-closest-point to a set of given measurements such as points, lines, subspaces, etc. Specifically, we solve the problems of finding the L_q mean of a set of points in $\mathbb{R}^N$ and on a Riemannian manifold of non-negative curvature; and the problem of finding the L_q-closest-point to a set of affine subspaces.

Given a set of measurements $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ in some metric space, the L_q mean or L_q-closest-point is defined as a point for which the sum of the q-th power of distances to the given measurements is the minimum. Depending on the type of problem these measurements $\mathcal{S}_i$ can be points, lines, subspaces, or a mixture of these. Here we are only interested in points in $\mathbb{R}^N$, points on a Riemanninan manifold of non-negative curvature and affine subspaces in $\mathbb{R}^N$. The L_q minimization function takes the following form

$$\min_{\mathbf{x}} \sum_{i=1}^{k} d(\mathbf{x}, \mathcal{S}_i)^q \,, \tag{1.1}$$

where $d(\cdot,\cdot)$ is a distance function and $1 \leq q < 2$. We refer to the minimum of this function as the L_q mean or L_q-closest-point.

1.1.2 L_q Non-linear Parameter Estimation Problems

The other class of distance minimization problems that we solve here is non-linear parameter estimation problems where we try to estimate parameters β of a model that best fits a given set of points. Let $\mathbf{x}_i$ be a noisy measurement of a vector $\bar{\mathbf{x}}_i$. Assuming that the measurement $\mathbf{x}_i$ is related by a non-linear function f_i to a parameter vector β. The L_q minimization function takes the following form,

$$\min_{\beta} \sum_{i=1}^{k} d(\mathbf{x}_i, f_i(\beta))^q \,, \tag{1.2}$$

where $d(\cdot,\cdot)$ is a distance function. We refer to the minimum of this function as an L_q minimum.

Applications: The applicability of the proposed algorithms in the area of computer vision is shown by solving the following problems: Firstly, we solve the rotation averaging problem, where several noisy observations of a rotation are averaged to get a better estimates of the rotation; Secondly, averaging of a set of Symmetric Positive-Definite (SPD) matrices is performed. Thirdly, we solve the triangulation problem, where a point in 3D space is obtained by finding a closest-point to the backtracked lines obtained from several noisy projections of the 3D point. Lastly, we solve the bundle adjustment problem, a non-linear parameter estimation problem. Given a set of images of a scene, bundle adjustment simultaneously estimates camera motion and 3D structure of the scene. Note that the applicability of the proposed algorithms is not only limited to the problems in the area of computer vision, rather a wide range of distance/error minimization problems can be solved by using the proposed algorithms.

For $q = 2$, the above mentioned problems have been studied extensively in the literature and their theory is well established. However, we are interested in finding the L_q solution of the problems, for $1 \leq q < 2$. Below we discuss some of the challenges in finding the L_q solution of the problems.

1.2 Challenges

Generally, least squares techniques are preferred whenever a simple and easy to understand solution of a problem is desired because these techniques are well studied and have simple formulation. In addition to being simple, least squares techniques are efficient because generally a closed form solution exists for these problems. However, least squares methods suffer from a major drawback when it comes to the robustness against outliers; even a single outlier in the data set can considerably deviate the solution from its true value. Thus, in order to make a least squares technique work for data with outliers, an outlier removal procedure is required as a pre-processing step. Due to this, least squares techniques are not very useful when there are outliers in data.

Thus, there is a need to explore other cost functions that are more robust to outliers. It is known that L_q cost functions, for $1 \leq q < 2$, are more robust to outliers than least squares functions. But the minimization of L_q cost functions generally require different optimization strategies than least squares optimization techniques. Therefore, it is important to devise methods that fit in the existing literature of least squares techniques and do not require new optimization strategies.

Generally, an iterative method is used to minimize an L_q cost function, for $1 \leq q < 2$. Iterative minimization techniques rely on finding an update in the descent direction to decrease the cost in successive iterations. There are several techniques in the literature to find an update in the descent direction, for example, line search, differentiation, etc. The efficiency of iterative methods primarily depend on how quickly these updates are computed. Therefore, it is important to propose methods that use simple techniques for the computation of updates.

After adding up all the above mentioned complications to the solution strategy, the implementation of an L_q optimization method is no longer an easy task, especially compared to least squares optimization strategies. Thus, a final issue of easy implementation has to be resolved to get new algorithms work for real world problems.

The proposed L_q optimization algorithms address all of the above mentioned issues. These new techniques not only find a robust L_q solution but also trivially fit in the existing literature of the least squares techniques because they minimize weighted least squares cost functions to find an L_q solution. As a result, the proposed algorithms are simple to understand and easy to implement.

1.3 L_q-Closest-Point Problems

In this section we discuss the proposed algorithms to solve the L_q-closest-point problems along with their advantages. The L_q-closest-point algorithms proposed in this thesis are inspired by

the Weiszfeld algorithm [Weiszfeld, 1937] that finds the L_1 mean of a set of points in $\mathbb{R}^N$.

Weiszfeld Algorithm: Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k\}$, the Weiszfeld algorithm finds the L_1 mean by iteratively minimizing a weighted L_2 cost function,

$$\min_{\mathbf{x}} \sum_{i=1}^{k} w_i^t \, \|\mathbf{x} - \mathbf{y}_i\|^2 \,,$$

where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{-1}$ and $\mathbf{x}^t$ is the estimate of the L_1 mean at an iteration t. A major advantage of the Weiszfeld algorithm is that the iterative updates are quick to compute because updates in descent directions are computed analytically.

1.3.1 Solution Strategy: Weiszfeld-style Algorithms

We show that the L_q-closest-point problems can be solved using a Weiszfeld inspired approach, that is, by minimizing a weighted L_2 cost function. We propose a generalization of the Weiszfeld algorithm to find the L_q mean of a set of points in $\mathbb{R}^N$. The algorithm is referred to as the L_q *Weiszfeld algorithm*. The L_q Weiszfeld algorithm finds the L_q mean of a set of points in $\mathbb{R}^N$ by iteratively minimizing a weighted L_2 function,

$$\min_{\mathbf{x}} \sum_{i=1}^{k} w_i^t \, d(\mathbf{x}, \mathbf{y}_i)^2 \,, \tag{1.3}$$

where w_i^t is a weight associated with $d(\mathbf{x}, \mathbf{y}_i)$ at the iteration t. We give a proof that if we choose $w_i^t = d(\mathbf{x}^t, \mathbf{y}_i)^{q-2}$ and iteratively minimize (1.3) then it will result in the minimization of the corresponding L_q cost function, that is $C_q = \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q$.

Furthermore, we prove that the L_q Weiszfeld algorithm can also be used to find the L_q mean of a set of points on a Riemannian manifold of non-negative sectional curvature, and to find the L_q-closest-point to a given set of affine subspaces in $\mathbb{R}^N$. The proposed algorithms inherit all the features of the Weiszfeld algorithm and are provably convergent to the L_q minimum.

Note: The L_q Weiszfeld algorithm is the same as the Iterative Re-weighted Least Squares (IRLS) technique, where a weighted least squares cost function is solved iteratively to find a relatively robust solution compared to the least squares technique. The IRLS techniques have existed in the literature for a long time, but here we show that for a particular choice of weights the IRLS algorithm converges to the desired L_q minimum, for $1 \leq q < 2$. The proposed technique must not be confused with the IRLS technique in compressed sensing [Daubechies et al., 2008; Chartrand and Yin, 2008; Eldar and Mishali, 2009], because in this thesis we solve a more general class of problems rather than the problems with a sparse solution. Here a solution to these problems can possibly be non-sparse. In other words, in our problems of interest it is robustness, and not sparsity, that is the goal.

1.3.2 Advantages of Weiszfeld-style Algorithms

The basic reason for proposing Weiszfeld-style algorithms is to overcome the disadvantages of least squares methods with a little compromise on their advantages. Just like the Wesizfeld algorithm, the proposed algorithms find the L_q solution of a problem by iteratively minimizing a weighted L_2 function. Therefore, the advantages of least squares methods in terms of simple formulation are inherited readily. The proposed algorithms being iterative optimization methods are not as efficient as least squares methods, but explicit computation of updates makes up for the high computational cost. Some of the advantages of the proposed Weiszfeld-style algorithms are summarized below:

- **Provably Convergent:** A major advantage of the proposed algorithms is that they are guaranteed to converge to the L_q minimum, for $1 \leq q < 2$. For each algorithm, we identify the convergence conditions and give a detailed proof of convergence.
- **Analytical Updates:** When proposing a new method it is important to show that the algorithm is efficient and can be applied to real world problems. The proposed algorithms are iterative gradient descent methods. Generally, gradient descent methods use a complex and computationally expensive process, such as line search, to find an update in the descent direction. On the contrary, the proposed algorithms have the advantage of computing updates analytically; hence do not require any complex strategy to find updates. As a result, the proposed algorithms are easy to implement and efficient.
- **Optimization Strategy:** Generally, the minimization of least squares cost functions is preferred because of their simple optimization strategies. On the other hand, the minimization of an L_q cost function requires a new optimization strategy that is sometimes very different than that of least squares optimization costs. However, the proposed algorithms do not require any new optimization strategy because the minimization of an L_q cost function is achieved by minimizing a weighted least squares cost function. Therefore, existing least squares optimization strategies are used to minimize the L_q cost functions.
- **Simple to Understand:** Since the theory of least squares methods is well established, the fact that the proposed algorithms solve weighted L_2 functions makes the proposed algorithms simple to understand.
- **Easy to Implement:** Since the proposed algorithms minimize weighted least squares cost functions and do not require any new optimization strategy, any existing least squares implementation of a problem can be exploited to find the L_q solution. Thus, the implementation of the L_q algorithm requires less effort than other L_q optimization methods in the literature that require specialized toolboxes to solve a problem.

1.4 L_q Non-linear Parameter Estimation Problems

In this section we discuss the L_q non-linear parameter estimation problem and the proposed solution strategy. We are particularly interested in the bundle adjustment problem and its solution using the Levenberg-Marquardt (LM) method. The Levenberg-Marquardt method is a

numerical method for solving non-linear least squares problems. We show that a non-linear least squares method, that is the LM method, can be used to find an L_q solution of the bundle adjustment problem.

Let $\mathbf{x}_i$ be a noisy measurement of a vector $\bar{\mathbf{x}}_i$ and $\mathbf{X}$ be a vector obtained by concatenating all measurements: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, where N is the total number of measurements. The true vector $\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \cdots, \bar{\mathbf{x}}_N)$ is assumed to be related by a non-linear function f to a parameter vector β, that is $\bar{\mathbf{X}} = f(\beta)$. Typically, the LM method minimizes the squared norm of a vector function as

$$\min_{\beta} \ \mathbf{E}^T\mathbf{E} \,, \tag{1.4}$$

where $\mathbf{E}$ is an error vector computed from the values of $\mathbf{X}$ and $f(\beta)$. Depending on the cost function to be minimized, there are several ways of computing the error vector $\mathbf{E}$. For example, when minimization of the least squares cost is desired then the vector $\mathbf{E}$ is taken as the difference of vectors $\mathbf{X}$ and $f(\beta)$, $\mathbf{E} = \mathbf{X} - f(\beta)$. Then from (1.4), the least squares cost function to be minimized is

$$C_2(\mathbf{X}, \beta) = \sum_i \ \|\mathbf{e}_i\|^2 \ = \sum_i \ d(\mathbf{x}_i, f_i(\beta))^2 \,, \tag{1.5}$$

where $d(\mathbf{x}_i, f_i(\beta))$ is the Euclidean distance between a measured value $\mathbf{x}_i$ and a predicted value $f_i(\beta)$; and $\mathbf{e}_i = \mathbf{x}_i - f_i(\beta)$, while it is assumed that the true vector $\bar{\mathbf{x}}_i$ is related by a function f_i to a parameter vector β, that is $\bar{\mathbf{x}}_i = f_i(\beta)$.

1.4.1 Solution Strategies: the Levenberg-Marquardt Method

We show that the Levenberg-Marquardt method can be used to find a robust solution to this problem, especially an L_q solution. The proposed technique is very simple and only requires a modification of the error vector $\mathbf{E}$. Thus, the minimization of a desired cost function is achieved by replacing the error vector $\mathbf{E}$ in (1.4) with a modified error vector $\mathbf{E}'$. This new vector is obtained by modifying each component of $\mathbf{E}$. This modification will result in the minimization of a desired cost function rather than a standard least squares cost function. The resultant minimization function is,

$$\min_{\beta} \ (\mathbf{E}')^T(\mathbf{E}') \,, \tag{1.6}$$

or equivalently,

$$\min_{\beta} \ \sum_i \mathbf{e}'^T_i \mathbf{e}'_i \,. \tag{1.7}$$

For example, if we choose $\mathbf{e}'_i = \mathbf{e}_i / \|\mathbf{e}_i\|^{(2-q)/2}$, then the above equation results in the minimization of an L_q cost function, that is the sum of the q-th power of errors. The resultant cost function can be minimized by using the LM method and does not require any new optimization strategy.

Since the proposed algorithm modifies a least squares technique, namely the LM method,

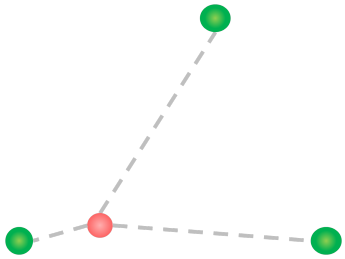


Figure 1.1: *L_q optimization for points in $\mathbb{R}^N$: The above figure shows three fixed (green) points and a starting (red) point from which the sum of the q-th power of distances to fixed (green) points is to be minimized.*

to find a robust solution, the proposed algorithm trivially fits in the existing domain of the standard least squares techniques. Even from an implementation point of view the minimization of a desired L_q function only requires a slight modification of an existing L_2 technique. Thus, the resultant algorithm is easy to understand and code.

1.5 Contributions

We apply the proposed algorithms to several problems in the area of computer vision. To illustrate the techniques, we consider in detail the following problems:

1. L_q optimization for points in $\mathbb{R}^N$.
2. L_q optimization for points on a Riemannian manifold.
   - L_q rotation averaging.
   - L_q averaging for Symmetric Positive-Definite matrices.
3. L_q optimization for subspaces:
   - L_q-closest-point to affine subspaces.
4. L_q non-linear parameter estimation.
   - L_q bundle adjustment.

In the rest of the chapter we will summarize these contributions.

1.5.1 L_q Optimization for Points in $\mathbb{R}^N$

Here we propose a generalization of the Weiszfeld algorithm to find the L_q mean of a set of points in $\mathbb{R}^N$, for $1 \leq q < 2$. The algorithm uses a Weiszfeld inspired approach to find the solution and is therefore referred as the L_q *Weiszfeld algorithm*. Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, we seek a point $\mathbf{x} \in \mathbb{R}^N$, for which the sum of the q-th power of distances to all the given points is the minimum, as shown in fig. 1.1. The minimization function is then

$$\min_{\mathbf{x} \in \mathbb{R}^N} \sum_{i=1}^{k} \|\mathbf{x} - \mathbf{y}_i\|^q ,$$

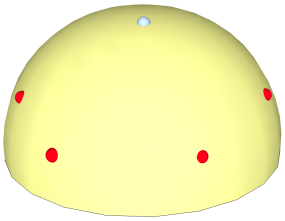


Figure 1.2: *L_q optimization for points on a Riemannian Manifold: The above figure represents a manifold with some given fixed points (red). The L_q Weiszfeld algorithm finds a point (white) for which the sum of the q-th power of distances to all the given points is the minimum.*

where $\|\cdot\|$ is the Euclidean distance and $1 \leq q < 2$.

The L_q Weiszfeld algorithm is simply an extension of the Weiszfeld algorithm to find the L_q mean. We identify the critical argument in the Weiszfeld algorithm, and show that it may be generalized to find the L_q solution, where $1 \leq q < 2$. In addition to the proof of convergence of the L_q Weiszfeld algorithm, we identify the conditions under which a general gradient descent algorithm converges. Our proofs of convergence of the L_q-closest-point algorithms, including the L_q Weiszfeld algorithm, are based on these conditions.

1.5.2 L_q Optimization for Points on a Riemannian Manifold

Just like the L_q optimization for points in $\mathbb{R}^N$, we propose a generalization of the L_q Weiszfeld algorithm to find the L_q mean of a set of points on a Riemannian manifold of non-negative curvature. The Weiszfeld algorithm has already been generalized to find the L_1 mean of a set of points on a Riemannian manifold of non-negative curvature [Fletcher et al., 2009]. We not only show that the L_q Weiszfeld algorithm can be used to find the L_q mean of a set of points but we further relax the bounds on the maximum possible distance between points on manifold.

Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ on a Riemannian manifold of non-negative curvature $\mathcal{M}$, we seek a point $\mathbf{x} \in \mathcal{M}$, for which the sum of the q-th power of geodesic distances to all the given points is the minimum, as shown in fig. 1.2. A *geodesic* is a generalization of the notion of straight line to Riemannian manifolds. The minimization function is then

$$\min_{\mathbf{x} \in \mathcal{M}} \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q \,,$$

where $d(\cdot, \cdot)$ is a geodesic distance between two points on $\mathcal{M}$. Note that the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$ is a special case of the Riemannian version of the algorithm where sectional curvature is null and geodesic distance is the Euclidean distance.

Starting from an initial estimate of the L_q mean, the proposed algorithm updates a current estimate by transferring all of the given points to the tangent space centered around a current estimate. The updated solution is then projected back to the manifold and the process is repeated until convergence. The key idea in the proof of convergence of the algorithm is to show that the projected solution results in a decrease in the L_q cost function.

We apply the L_q algorithm to the problems of finding the L_q mean of a set of rotations and of Symmetric Positive-Definite (SPD) matrices. These two problems are discussed below:

1.5.2.1 L_q Rotation Averaging

We apply the proposed algorithm to the problem of rotation averaging where several estimates of a rotation are averaged to find a better estimate. A rotation matrix is used to perform a rotation in the Euclidean space. The problem of L_q rotation averaging takes two forms: single rotation averaging in which several estimates of a single rotation are averaged to give the best estimate; and multiple rotation averaging, in which relative rotations $\mathtt{R}_{ij}$ are given, and absolute rotations $\mathtt{R}_i$ are computed to satisfy the compatibility constraint $\mathtt{R}_{ij}\mathtt{R}_i = \mathtt{R}_j$. For single-rotation averaging the algorithm provably finds the global L_q optimum. However, for multiple rotation averaging no such proof exists.

The problem of rotation averaging has significant applications to structure and motion and to non-overlapping camera calibration. It has been studied quite extensively in the past, both in computer vision and in other fields. Some interesting applications of averaging include camera network calibration [Tron and Vidal, 2009], sensor network localization [Cucuringu et al., 2012a] and molecule problem in structural biology [Cucuringu et al., 2012b].

1.5.2.2 L_q Averaging for Symmetric Positive-Definite (SPD) Matrices

We use the proposed algorithm to solve the problem of finding the L_q mean of a set of symmetric positive-definite (SPD) matrices. The space of SPD matrices is not a vector space; instead, it is a Riemannian manifold that constitutes a convex half-cone in the vector space of matrices. The structure of the Riemannian manifold of SPD matrices depends on the metric induced. Some of the popular metrics that we discuss are: the Euclidean metric, the Log-Euclidean metric [Arsigny et al., 2007]. and the Affine-Invariant metric [Pennec et al., 2006; Fletcher and Joshi, 2004; Lenglet et al., 2004; Moakher, 2005]. In this case we are more interested in the Log-Euclidean metric because the SPD manifold has a null sectional curvature when endowed with the Log-Euclidean metric and it does not suffer from the drawbacks of other metrics.

In the area of computer vision and medical imaging SPD matrices have a lot of applications. In medical imaging SPD matrices are used to model the anatomical variability of the brain [Fillard et al., 2007] and to encode the principal diffusion directions in Diffusion Tensor Imaging (DTI) [Fletcher and Joshi, 2007; Pennec et al., 2006]. Furthermore, SPD matrices are widely used in computer vision for motion analysis and texture segmentation [Brox et al., 2003]; and to model the appearance of objects for tracking [Porikli et al., 2006; Tyagi et al., 2008]. Clearly, the application of SPD matrices is not only limited to the area of computer vision and medical imaging. Outside the domain of computer vision, in physics SPD matrices can be used as stress-strain tensors [Moakher, 2006; Salençon, 2001] and they can also be used to solve partial differential equations (PDEs) [Borouchaki et al., 1997].

1.5.3 L_q Optimization for Subspaces

Here we address the problem of finding the L_q-closest-point to a set of affine subspaces. The problem of finding the L_q mean of a set of points in $\mathbb{R}^N$ is a special case of the problem of

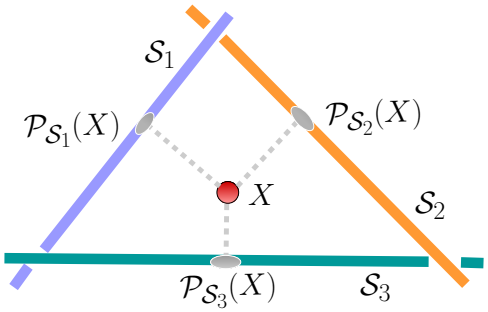


Figure 1.3: *L_q-closest-point to affine subspaces: The above figure shows three subspaces (lines), $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. We seek a point $\mathbf{X}$ for which the sum of the q-th power of orthogonal distances is the minimum, $\sum_i d(\mathbf{X}, \mathcal{S}_i)^q$. Gray points represent the orthogonal projections of a red point $\mathbf{X}$, on subspaces.*

finding the L_q-closest-point to a set of affine subspaces in $\mathbb{R}^N$ where all the subspaces are zero dimensional, that is points. Thus, the proposed algorithm is a generalization of the L_q Weiszfeld algorithm for higher dimensional subspaces in $\mathbb{R}^N$.

1.5.3.1 L_q-closest-point to Subspaces

In most of the computer vision applications information can be represented by affine subspaces, for example, multi-body structure and motion, objects under different illumination settings, etc. We present a method, based on the L_q Weiszfeld algorithm, to find an L_q-closest-point, "intersection" point, to a set of affine subspaces. Once again the minimization of an L_q cost function is achieved by iteratively minimizing a weighted L_2 cost function.

For a given set of affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, possibly of different dimensions, the L_q-closest-point to the given subspaces is the one for which the sum of the q-th power of orthogonal distances is the minimum, as shown in fig. 1.3. The minimization function is

$$\min_{\mathbf{X} \in \mathbb{R}^N} \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q ,$$

where $\|\cdot\|$ is the Euclidean norm and $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ is an orthogonal projection of a point $\mathbf{X}$ on the i-th subspace $\mathcal{S}_i$.

The problem has significant applications in the field of geometrical computer vision. We consider the problem of triangulation in which a $3D$ point in space is determined by finding the point of intersection of lines, where each line is passing through the center of the camera and intersecting the image plane at the corresponding image point. Due to various types of noise in image point measurements these lines are a skewed form of the original lines. As a result, these skewed lines do not intersect at a single point, possibly these lines may not intersect at all. So the problem is reduced to finding the optimal point of intersection of 1-dimensional subspaces, that is lines, in $\mathbb{R}^3$.

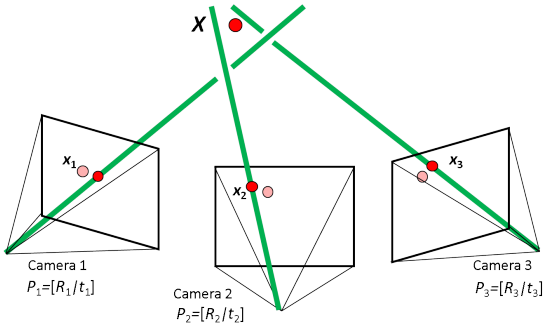


Figure 1.4: *Bundle Adjustment: We seek camera matrices* $\mathtt{P}_i$ *and a 3D point* $\mathbf{X}$ *for which the sum of re-projection errors* $\sum_i d(\mathtt{P}_i\mathbf{X}, \mathbf{x}_i)^q$ *is the minimum.*

1.5.4 L_q Non-linear Parameter Estimation

In parameter estimation problems we want to fit a model to a given set of measurements. We specifically consider the bundle adjustment problem where camera parameters and 3D structure of a scene are recovered simultaneously from multiple images of the scene.

1.5.4.1 L_q Bundle Adjustment

Given multiple images of a scene, bundle adjustment [Hartley and Zisserman, 2004; Triggs et al., 2000a; Engels et al., 2006] simultaneously estimates camera matrices and 3D points in the scene and is therefore a non-linear optimization problem. Several techniques exist in the literature to minimize a non-linear least squares problem, such as the Gauss-Newton method, the Leveberg-Marquardt (LM) method, Trust Region methods, etc. The LM method, a least squares technique, has become a fairly standard optimization technique to solve the bundle adjustment problem. By using the proposed techniques the Levenberg-Marquardt method can be used to solve for a robust solution of the bundle adjustment problem, especially an L_q solution.

Given a set of image point measurements $\mathbf{x}_{ij}$, bundle adjustment solves for camera matrices $\mathtt{P}_i$ and 3D points $\mathbf{X}_j$ such that $\mathbf{x}_{ij} = \mathtt{P}_i\mathbf{X}_j$. In practice, due to the noise in image point measurement the reprojected point $\hat{\mathtt{P}}_i\hat{\mathbf{X}}_j$ is not same as the measured image point $\mathbf{x}_{ij}$, as shown in fig. 1.4. The problem is then to find an optimal estimate of the camera parameter $\hat{\mathtt{P}}_i$ and the 3D point $\hat{\mathbf{X}}_j$ such that the error between the measured image points $\mathbf{x}_{ij}$ and the estimated image points $\hat{\mathtt{P}}_i\hat{\mathbf{X}}_j$ is minimized. The L_q cost function to be minimized takes the following form,

$$\min_{\hat{\mathtt{P}}_i, \hat{\mathbf{X}}_j} \sum_{i,j} d(\hat{\mathtt{P}}_i\hat{\mathbf{X}}_j, \mathbf{x}_{ij})^q \, .$$

When compared to the least squares problem, this problem is considered as a hard problem and is solved using complex optimization strategies.

By using the proposed techniques the Levenberg-Marquardt method can be used to find a solution of the above problem. We propose several strategies to minimize robust cost functions, especially the L_q cost function. The minimization of the L_q cost function is achieved using two different methods. The first method, we call the L_q *method*, minimizes the sum of the q-th

power of distances using a modified difference vector in the LM method. The second method, *Iterative Re-Weighted Least Squares (IRLS)*, finds an L_q solution by iteratively minimizing a weighted least squares cost function, an L_q Weiszfeld inspired approach.

1.6 Thesis Outline

The rest of the thesis contains one background chapter and four contribution chapters. In chapter 2, we start by reviewing the L_1 optimization techniques where we provide a brief overview the Weiszfeld algorithm for points in $\mathbb{R}^N$ and on a Riemannian manifold of non-negative sectional curvature. The main contributions of this thesis are presented in chapters 3, 4, 5 and 6. In chapter 3, we propose a generalization of the Weiszfeld algorithm to find the L_q mean of a set of points in $\mathbb{R}^N$, referred as the L_q Weiszfeld algorithm; and then identify the conditions under which descent algorithms are guaranteed to converge. In chapter 4, we further extend the theory presented in chapter 2 by proposing an algorithm, based on the L_q Weiszfeld algorithm, to find the L_q mean of a set of points on a Riemannian manifold of non-negative sectional curvature. In addition to the proof of convergence of the algorithm, we relax the bounds on the maximum distance between points on manifold. Furthermore, we show that the proposed algorithm can be used to find the L_q mean of a set of rotations and Symmetric Positive-Definite matrices, and our experimental results confirm that the L_q mean is more robust to outliers than the L_2 mean. In chapter 5, we propose an extension of the L_q Weiszfeld algorithm to find the L_q-closest-point to a set of affine subspaces. Chapter 6 presents several methods to find a robust solution, especially an L_q solution, of the bundle adjustment problem. Finally, in chapter 7 we conclude the thesis.

Chapter 2

Background: L1 Optimization for Points

In this chapter we review the existing techniques in the literature to find the L_1 mean of a set of points in $\mathbb{R}^N$ and on a Riemannian manifold of non-negative curvature. Given a set of points, the L_1 mean is a point for which the sum of distances to all the given points is the minimum. The L_1 mean is also known as the Geometric Median, a generalization of the univariate median to higher dimensional spaces. A well known algorithm to find the L_1 mean of a set of points in $\mathbb{R}^N$ is the Weiszfeld algorithm [Weiszfeld, 1937]. The Weiszfeld algorithm is provably convergent iterative method that does not require computation of derivatives or line-search. Therefore it is very easy to understand and code. The iterative update is very quick to compute and in practice, the algorithm is quick to converge.

The Weiszfeld algorithm has been studied extensively in the past and several extensions of the algorithm have also been proposed. However, in this thesis we are only interested in the Weiszfeld algorithm [Weiszfeld, 1937] and its extension for points on a Riemannian manifold [Fletcher et al., 2009]. It is known that the L_1 mean or geometric median of a set of points is more robust than the L_2 mean or arithmetic mean. For instance, if all of the given points lie in a one dimensional space, that is a line, then the L_2 mean is the usual arithmetic mean of the points, whereas the geometric median is the median. Thus, the geometric median is less affected by the presence of distant (outlier) values. Our interest in this problem relates to its robustness to outliers.

In the following section we will discuss several measures of centrality. We dedicate the rest of the chapter for discussion on the Geometric median or L_1 mean, the Weiszfeld algorithm and its extension for points on a Riemannian manifold.

2.1 Measure of Central Tendency

In statistics, a measure of central tendency is a single value that attempts to describe a set of data by identifying the central position within that set of data. It is also known as an average of the data. Some of the most common measures of central tendency are the arithmetic mean, the median and the mode. The arithmetic mean is also known as the L_2 mean. Depending on univariate or multivariate data these measures of central tendency are computed differently. For example, a univariate median is the middle value that separates the higher half from the

lower half of the data; however, a multivariate median can be computed in several ways, a straightforward way is to compute univariate median for each component of the input vectors.

We are only interested in one particular type of multivariate median, that is, the geometric median. Therefore, we focus our discussion on the arithmetic mean and the median. In the rest of the section we discuss the mean and the median.

2.1.1 Mean

There are several types of mean, for example, the arithmetic mean, the geometric mean, the Fréchet mean, etc. Among these measures, the arithmetic mean is commonly used where an L_2 cost function, that is the sum of squared distances, is minimized. The arithmetic mean is also referred as the L_2 mean. The Fréchet mean also minimizes the sum of squared distances but in this case the data lies on a surface, specifically, on a Riemannian manifold. Therefore, here we only consider the arithmetic mean and the Fréchet mean.

Arithmetic Mean: For a given set of measurements $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, the arithmetic mean minimizes the sum of squared distances,

$$\min_{\mathbf{x}} \sum_{i=1}^{k} \|\mathbf{x} - \mathbf{y}_i\|^2 ,$$

where $\|\cdot\|$ is the Euclidean norm. By solving the above equation the arithmetic mean is simply the sum of all measurements $\mathbf{y}_i$ divided by the number of observations in the data set,

$$\bar{\mathbf{x}} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{y}_i ,$$

Regardless of whether the data is univariate or multivariate data, the arithmetic mean is computed by using the same method.

Fréchet Mean: Just like the arithmetic mean, the Fréchet mean also minimizes an L_2 cost function but in this case data points lie on a surface or, more generally, on a Riemannian manifold. Unlike many other means, the Fréchet mean is defined on a space whose elements cannot necessarily be added together or multiplied by scalars. The minimization function is still a sum of squared distance function,

$$\min_{\mathbf{x}} \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^2 ,$$

where $d(\cdot, \cdot)$ is a distance function. It is also known as the Karcher mean, named after Hermann Karcher.

2.1.2 Median

The median is of central importance in robust statistics, as it is the most robust measure of central tendency having a breakdown point of 50% which means that as long as no more than half of the data is outliers, the median will not give an arbitrarily large result. Unlike the mean, the univariate median and multivariate median have different forms. There are several types of multivariate medians but they all share a common property, that is, for one-dimensional data they are the same as the univariate median. In the rest of the section we will discuss the univariate and multivariate medians in detail.

Univariate Median: For a given set of points on a line, that is one-dimensional data, the median is the middle value that separates the higher half from the lower half of the dataset. The median of a finite list of numbers can be found by arranging all the observations from lowest value to highest value and picking the middle one. If there is an even number of observations, then the median is the mean of the two middle values. Note that the median is only defined on the order of data and is independent of any distance metric. Thus, the median of a set of points will not change, unless more than 50% of the points are shifted to some other value.

Multivariate Median: As mentioned before that the extension of the univariate median to multi-dimensional data is not straight forward and there are several measures of median for mutli-dimensional data. When the dimension is two or higher, there are multiple concepts that extend the definition of the univariate median. However, these multivariate medians agree with the univariate median for one-dimensional data. Some of the commonly used multivariate medians are summarized below.

- **Marginal Median:** For a given set of vectors, a marginal median is defined to be the vector whose components are univariate medians.

- **Geometric Median or L_1 Mean:** The geometric median of a given set of points minimizes the sum of distances to all the given points. It is also known as the L_1 mean. The geometric median is unique when given points are non-collinear. This is the same as the median when applied to one-dimensional data, but it is not the same as taking the median of each dimension independently.

There are several other generalizations of the multivariate median such as the affine equivariant Oja median [Oja, 1983; Niinimaa et al., 1990], the Halfspace median [Tukey, 1975], etc. Some common ideas of breakdown, equivariance, symmetry and computational convenience for several definitions of multivariate median are discussed in [Small, Dec. 1990].

Since we are only interested in the geometric median, a detailed discussion of multivariate medians is out of the scope of this thesis. We dedicate the rest of the chapter for discussion on the geometric median and algorithms to find it. In the next section we discuss the geometric median in more detail.

2.2 Geometric Median

The geometric median of a discrete set of sample points in a Euclidean space is the point that minimizes the sum of distances to the sample points. This generalizes the univariate median, and provides a measure of central tendency in higher dimensions. The geometric median is an important estimator of location in statistics. It is also a standard problem in facility location, where it models the problem of locating a facility to minimize the cost of transportation.

The median of a data set is a robust estimate of centrality. It has a different character from the arithmetic mean as is illustrated by different breakdown properties. In the case of the mean it suffices to have a single point at infinity to send the mean to infinity. On the other hand, at least 50% of the data must be moved to infinity to force the median to do the same.

Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ in some metric space, the L_1 mean or geometric median is a point $\bar{\mathbf{x}}$ that minimizes the sum of distances to all given points,

$$\bar{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}} \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i) \,, \tag{2.1}$$

where $d(\cdot, \cdot)$ is a distance function. A well-known and provably convergent algorithm for finding the L_1 mean of a set of points in $\mathbb{R}^N$ is the Weiszfeld algorithm [Weiszfeld, 1937].

2.2.1 History: Fermat-Weber Problem

A special case of the above problem is known as Fermat's problem where the L_1 solution of 3 sample points in a plane is desired. This problem was originally posed by Pierre de Fermat to Evangelista Torricelli, who solved it. Its solution is known as the Fermat point of a triangle with each sample point as a vertex. The more general form of the problem for more than 3 points in $\mathbb{R}^N$ was studied by Alfred Weber [Weber, 1909] and therefore is known as the Fermat-Weber problem.

The Fermat-Weber problem is a well studied problem in the area of operations research [Kulin and Kuenne, 1962; Plastria and Elosmani, 2008; Weiszfeld and Plastria, 2009]. In the area of operations research and computational geometry the Fermat-Weber problem is studied in the context of the facility location problem. The goal is to find an optimal location for the placement of a set of facilities to reduce the distance to each of the fixed demand points. The Fermat-Weber problem is the simplest form of facility location problem where the location of a single facility is desired such that the distance to each of the fixed demand points is the minimum, as shown in fig. 2.1.

A mechanical system is shown in fig. 2.2 to demonstrate the working of the Fermat-Weber problem. In fig. 2.2 each given point $\mathbf{y}_i$ is represented by a pulley. Strings are passed through pulleys and unit weights are attached to one end of the strings. The other ends of all the strings are tied together. This mechanical system will reach an equilibrium state. At this point all the forces (due to unit weights) in all directions will cancel each other and there will be no change in the location of the knot connecting the strings. The final location of the knot is also the L_1 solution in case of points in $\mathbb{R}^2$ (2.1), because at the minimum point all the unit magnitude forces cancel each other. It follows that at this point the gradient of (2.1) is zero, it is shown in (2.4) that the gradient of (2.1) is in fact the sum of unit vectors.

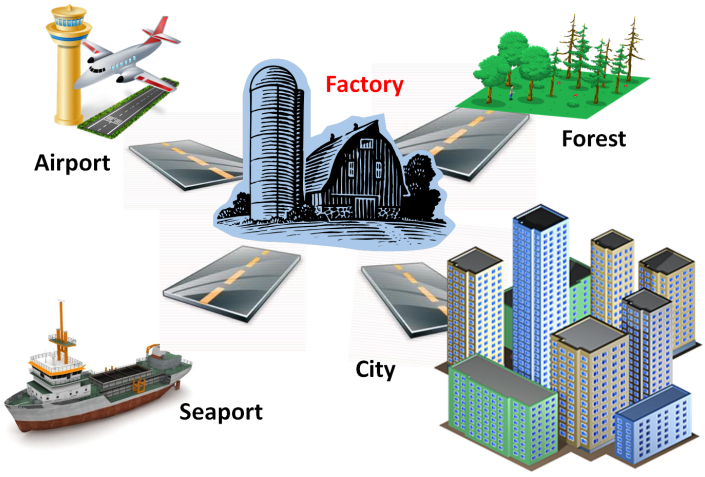


Figure 2.1: *Facility location problem: The problem is to find an optimal location for the placement of a single facility (factory) such that the sum of distances to all resources (forest, city, airport and seaport) is the minimum.*

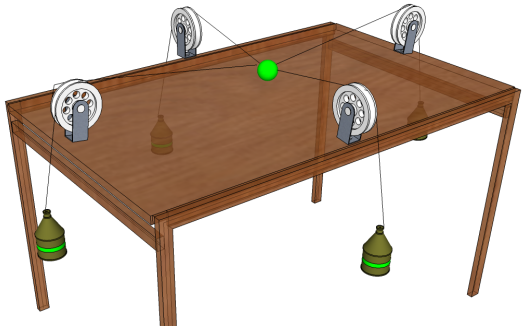


Figure 2.2: *Mechanical setup for the Fermat-Weber problem: Each pulley represents a fixed point. Strings are passed over the pulleys and unit weights are attached to one end of string while other ends are tied together. The system will eventually reach an equilibrium state and that point will be the L_1 minimum.*

2.2.2 Properties of Geometric Median

Here we discuss some properties of the geometric median. For a detailed discussion on multivariate medians and their properties see [Small, Dec. 1990].

- For the 1-dimensional case, the geometric median coincides with the median. This is because the univariate median also minimizes the sum of distances from the points.
- **Uniqueness:** The geometric median is unique whenever the points are not collinear.
- **Equivariance:** The geometric median is equivariant for Euclidean similarity transformations, including translation and rotation. This means that one would get the same result either by transforming the geometric median, or by applying the same transformation to the sample data and finding the geometric median of the transformed data.
- **Breakdown point:** The geometric median has a breakdown point of 0.5. That is, up to half of the sample data may be arbitrarily corrupted, and the median of the samples will still provide a robust estimator for the location of the uncorrupted data.

In the next section we discuss the Weiszfeld algorithm to find the L_1 mean or geometric median of a set of points in $\mathbb{R}^N$.

2.3 Weiszfeld Algorithm for Points in $\mathbb{R}^N$

The Weiszfeld algorithm solves for the L_1 minimum of a set of points in $\mathbb{R}^N$. Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ where $\mathbf{y}_i \in \mathbb{R}^N$, their L_1 mean or geometric median is the point $\mathbf{x}$ that minimizes the cost function

$$C_1(\mathbf{x}) = \sum_{i=1}^{k} \|\mathbf{x} - \mathbf{y}_i\| \,, \tag{2.2}$$

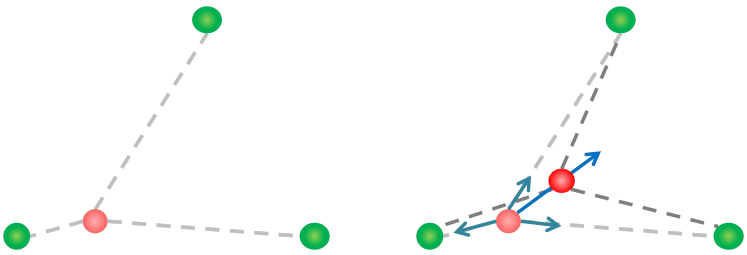
where $\|\cdot\|$ is the Euclidean norm.

The Weiszfeld algorithm updates a current estimate $\mathbf{x}^t$ to

$$\mathbf{x}^{t+1} = \frac{\sum_{i=1}^{k} w_i^t \, \mathbf{y}_i}{\sum_{i=1}^{k} w_i^t} \,, \tag{2.3}$$

where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{-1}$. If all of the given points are non-collinear then the cost function C_1 has a unique minimum, and the sequence of iterates $\mathbf{x}^t$ will converge to the minimum of the cost (2.2), except if it gets stuck at one of the points $\mathbf{y}_i$, as explained next.

Continuity of the Update: If one of the iterations $\mathbf{x}^t$ approaches one of the points $\mathbf{y}_i$, then the weight w_i^t becomes very large, and in the limit, the update step, as given by (2.3) becomes undefined. However, this is a *removable* singularity. Suppose that one of the points, say $\mathbf{y}_1$ is the one closest to an iteration $\mathbf{x}^t$, then one may replace all the weights by $\tilde{w}_i^t = w_i^t / w_1^t$ and $w_1' = 1$ without altering the update. With these weights, the update step is well-defined, and



(a) Iteration 1 (b) Iteration 2

Figure 2.3: *Weiszfeld Algorithm (Gradient Descent Form): (a) shows three fixed points (green) and a starting point (red) from which the sum of distances to fixed points (green) is to be minimized. (b) shows an updated point (red) after one iteration of the Weiszfeld algorithm in the descent direction.*

continuous, even when $\mathbf{x}^t$ is equal to one of the points $\mathbf{y}_i$. Although this renormalization of weights removes the apparent singularity in the definition of the update step, we shall continue to use the formula $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{-1}$, just so as to avoid complicating the exposition.

Although this trick removes the singularity at the points $\mathbf{y}_i$, the update so defined results in $\mathbf{x}^{t+1} = \mathbf{x}^t = \mathbf{y}_i$ whenever $\mathbf{x}_t$ equals $\mathbf{y}_i$ exactly. Thus, the sequence of iterations gets stuck at $\mathbf{y}_i$. In this case, it cannot be concluded (and is not generally true) that $\mathbf{y}_i$ is the minimum of the cost function.

Getting Stuck: This possibility of "getting stuck" at a value $\mathbf{x}^t = \mathbf{y}_i$ is perhaps the main theoretical flaw of the Weiszfeld algorithm. From a practical point of view, however, it is not a significant issue. This eventuality is rarely if ever encountered in practice. If it is, there are ways to handle it.

A simple strategy if $\mathbf{x}^t$ coincides with one of the $\mathbf{y}_i$ is to displace the iterate $\mathbf{x}^t$ slightly and continue. It may be shown that successive iterates will "escape" from some point $\mathbf{y}_i$, not the minimum, by approximately doubling the distance at each iteration. A second possibility is to start the iteration at some point with cost smaller than the cost of any of the points $\mathbf{y}_i$, in which case it is not possible that an iteration will return to approach one of the points.

2.3.1 Different Interpretations of the Weiszfeld Algorithm

The Weiszfeld algorithm can be viewed as a gradient descent algorithm, Iterative Re-Weighted Least Squares (IRLS) algorithm and Weighted mean algorithm. Below we discuss several interpretations of the Weiszfeld algorithm.

2.3.1.1 Gradient Descent Form

The Gradient of the cost function (2.2) is

$$\nabla C_1 = -\sum_{i=1}^{k} \frac{\mathbf{y}_i - \mathbf{x}}{\|\mathbf{y}_i - \mathbf{x}\|} \,. \tag{2.4}$$

Given a current estimate of the L_1 minimum $\mathbf{x}^t$ at iteration t, the next estimate of the minimum in the descent direction is computed as

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \lambda \nabla C_1 \,, \tag{2.5}$$

where λ is the update step size, fig. 2.3. The value of λ in case of the Weiszfeld algorithm is

$$\lambda = \frac{1}{\sum_{i=1}^{k} w_i^t} \,,$$

where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{-1}$. By substituting the value of λ and ∇C_1 in (2.5), an updated estimate $\mathbf{x}^{t+1}$ is computed as

$$\mathbf{x}^{t+1} = \frac{\sum_{i=1}^{k} w_i^t \, \mathbf{y}_i}{\sum_{i=1}^{k} w_i^t} \,.$$

An advantage of the Weiszfeld algorithm is that the descent direction and step size is computed in closed form. Therefore each iteration of the Weiszfeld is fast as compared to other gradient descent algorithms that compute the step size using complex strategies such as line search, etc.

2.3.1.2 Iterative Re-weighted Least Squares (IRLS) Form

Note that (2.3) updates a current estimate $\mathbf{x}^t$ by computing a weighted mean of points $\mathbf{y}_i$. An alternative interpretation of the Weiszfeld algorithm is that it can be viewed as an Iterative Re-weighted Least Squares method. At iteration t, the weighted least squares cost function is

$$\mathbf{x}^{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{k} w_i^t \, \|\mathbf{x} - \mathbf{y}_i\|^2 \,, \tag{2.6}$$

where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{-1}$. By taking the derivative of the weighted Least Squares function and equating to zero we get an update function same as (2.3). At each step $\mathbf{x}^{t+1}$ is the exact minimum of the weighted problem. Thus, the Weiszfeld algorithm solves a special type of the IRLS cost function to achieve the L_1 solution.

From the above discussion it is obvious that the L_1 solution can be obtained by minimizing either an L_1 cost or a weighted L_2 function as in (2.6). This makes the Weiszfeld algorithm very simple to understand and easy to implement. For a details convergence proof and conditions of convergence see [Weiszfeld, 1937].

We propose a generalization of the Weiszfeld algorithm, the L_q Weiszfeld algorithm, to find the L_q mean of a set of points in $\mathbb{R}^N$, for $1 \leq q < 2$. The L_q mean of a set of points minimizes the sum of the q-th power of distances. The L_q Weiszfeld algorithm inherits all the properties of the Weiszfeld algorithm.

2.3.2 Generalizations of the Weiszfeld algorithm

Several generalizations of the Weiszfeld algorithm have also been proposed. Some of the generalizations are summarized below:

L_1 Generalizations of the Weiszfeld Algorithm: The original paper by Weiszfeld [Weiszfeld, 1937] gave an algorithm for the problem of finding the point in $\mathbb{R}^N$ that minimizes the sum of distances to a given set of points. Since then it has been generalized to L_1-closest-point problems in Banach spaces [Eckhardt, 1980], rotation space $\mathrm{SO}(3)$ [Hartley et al., 2011, 2013] and general Riemannian manifolds [Fletcher et al., 2009; Yang, 2010]. The characteristic of the Weiszfeld algorithm and its generalizations is that they are provably convergent iterative L_1 optimization algorithms that do not require computation of derivatives or line-search, and in therefore, the algorithms are quick to converge in practice.

L_q Generalization of the Weiszfeld Algorithm: The Weiszfeld algorithm has been studied extensively in other fields of research and is even generalized to solve different form of the problem. A Weiszfeld [Weiszfeld, 1937] inspired solution strategy to solve for the minimum of L_q norm of the problem has also been proposed in [Brimberg and Love, 1993; Brimberg and Chen, 1998; Brimberg, 2003; Morris and Verdini, 1979]. Note that in [Brimberg and Love, 1993] L_q norm is minimized, that is

$$\min_{\mathbf{x}} \sum_{i=1}^{k} w_i \|\mathbf{x} - \mathbf{y}_i\|_q.$$

This generalization must not be confused with the type of problem we solve in this thesis. That is fundamentally different from the type of problems we solve, where the sum of the q-th power of distances is minimized, that is

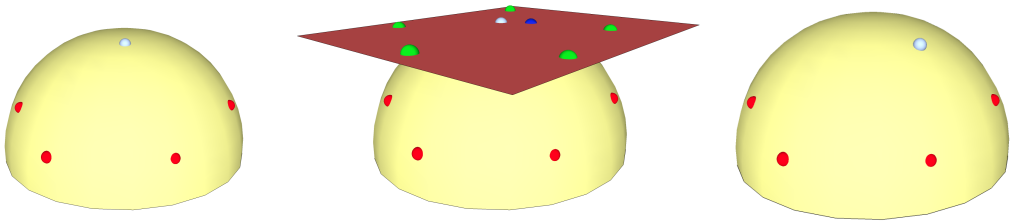
$$\min_{\mathbf{x}} \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q.$$

For points in $\mathbb{R}^N$ the distance function is $d(\mathbf{x}, \mathbf{y}_i) = \|\mathbf{x} - \mathbf{y}_i\|_2$.

2.4 Weiszfeld Algorithm on a Riemannian Manifold

We now discuss an extension of the Weiszfeld algorithm to find the L_1 minimum of a set of points on a Riemannian manifold of non-negative sectional curvature [Fletcher et al., 2009]. Given a set of points, $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ on a Riemannian manifold $\mathcal{M}$, of non-negative sectional curvature the L_1 mean or geodesic median is a point $\mathbf{x} \in \mathcal{M}$ for which the sum of geodesic distances

$$C_1(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i) = \sum_{i=1}^{k} \| \log_{\mathbf{x}}(\mathbf{y}_i) \| \,, \tag{2.7}$$



(a) Initial Setup (b) Update in Tangent Space (c) Projected Solution

Figure 2.4: *Weiszfeld Algorithm on Manifold: (a) represents a manifold with some given fixed points (red) and a starting point (white). (b), the Weiszfeld algorithm is applied to the transformed points (green) in the tangent space (red plane) and an updated point (blue) is computed in a descent direction. (c), the updated point in the tangent space is mapped back to the manifold. This procedure is repeated until convergence.*

is minimized, where $d(\cdot,\cdot)$ is the geodesic distance between two points on the manifold $\mathcal{M}$ and $\log_{\mathbf{x}}(\mathbf{y})$ is the *logarithm map* that takes a point $\mathbf{y} \in \mathcal{M}$ to the tangent space $T_{\mathbf{x}}\mathcal{M}$ of $\mathcal{M}$ centered at $\mathbf{x} \in \mathcal{M}$.

Given some points on a Riemannian manifold and an initial estimate of their L_1 mean. An updated solution is computed by transferring all the given points to the tangent space of the manifold centered at the current estimate; see fig. 2.4. The point is then transferred back to the manifold. This process is repeated until convergence. This technique is convergent [Fletcher et al., 2009] on manifolds of non-negative sectional curvature, provided all the points lie in a suitably small convex set, as will be described in later chapters.

The gradient of the cost function (2.7) is

$$\nabla C_1(\mathbf{x}) = -\sum_{i=1}^{k} \frac{\log_{\mathbf{x}}(\mathbf{y}_i)}{\|\log_{\mathbf{x}} \mathbf{y}_i\|} .$$

A current solution $\mathbf{x}^t$ is updated in the descent direction as

$$\mathbf{x}^{t+1} = \exp_{\mathbf{x}^t}(-\lambda \, \nabla C_1(\mathbf{x}^t)) , \tag{2.8}$$

where λ is the step size for the gradient descent algorithm and $\exp_{\mathbf{x}^t}(\mathbf{v})$ is the *exponential map* that maps a vector $\mathbf{v} \in T_{\mathbf{x}^t}\mathcal{M}$ to a point on $\mathcal{M}$.

In case of the Weiszfeld algorithm the step size is

$$\lambda = \frac{1}{\sum_{i=1}^{k} 1/d(\mathbf{x}^t, \mathbf{y}_i)}.$$

By substituting the value of λ in (2.8) we get

$$\mathbf{x}^{t+1} = \exp_{\mathbf{x}^t}\left(\frac{\sum_{i=1}^{k} \log_{\mathbf{x}^t}(\mathbf{y}_i) \;/\; \|\log_{\mathbf{x}^t}(\mathbf{y}_i)\|}{\sum_{i=1}^{k} 1 \;/\; \|\log_{\mathbf{x}^t}(\mathbf{y}_i)\|}\right) . \tag{2.9}$$

It may be written more simply as

$$\mathbf{x}^{t+1} = \exp_{\mathbf{x}^t}\left(\frac{\sum_{i=1}^{k} w_i^t \log_{\mathbf{x}^t}(\mathbf{y}_i)}{\sum_{i=1}^{k} w_i^t}\right). \tag{2.10}$$

where $w_i^t = d(\mathbf{x}^t, y_i)^{-1} = \|\log_{\mathbf{x}^t}(\mathbf{y}_i)\|^{-1}$. This update equation is seen to come from finding the weighted average in the tangent space at $\mathbf{x}^t$ of the points $\log_{\mathbf{x}^t}(\mathbf{y}_i)$, followed by mapping back to the manifold by the exponential map.

A proof and conditions for the convergence of the L_1 Weiszfeld algorithm on manifolds of non-negative sectional curvature can be found in [Fletcher et al., 2009; Yang, 2010]. However, the L_q Weiszfeld algorithm proposed in this thesis solves for the L_q solution $1 \leq q < 2$ of the problem. In addition, the proof given in this thesis is more complete than the proofs in [Fletcher et al., 2009] and [Yang, 2010], even for the L_1 case. In particular, we improve the bounds on maximum distance between points on manifold for which the algorithm will converge, compared to those given in [Fletcher et al., 2009], and we fill in some detail concerning the convergence point of the algorithm (for example, what happens when the algorithm converges to some $\mathbf{y}_i$.) In [Yang, 2010], a line-search is used at each step, so the algorithm is not a true Weiszfeld-style algorithm.

2.5 Summary

In summary, the L_1 mean or geometric median is a more robust measure of central tendency than the L_2 mean or arithmetic mean. A classic algorithm to find the L_1 mean or geometric mean of a set of points in $\mathbb{R}^N$ is the Weiszfeld algorithm. An extension of the Weiszfeld algorithm shows that a Weiszfeld inspired approach can be used to find the L_1 mean of a set of points on a Riemannian manifold of non-negative curvature. The Weiszfeld algorithm has an advantage of being simple since it minimizes a weighted L_2 cost function. Another advantage of the Weiszfeld algorithm is that updates are computed analytically This makes the algorithm even more attractive.

Chapter 3

Lq Optimization for Points in $\mathbb{R}^N$

In this chapter we propose a generalization of the Weiszfeld algorithm to find the L_q mean, for $1 \leq q < 2$, of a set of points in $\mathbb{R}^N$, we refer to it as the L_q *Weiszfeld Algorithm*. The L_q mean of a set of points, as described before, is the point for which the sum of the q-th power of distances to all the given points is the minimum. The L_q Weiszfeld algorithm finds the L_q mean by using a gradient descent approach, where updates are computed analytically. This eliminates the need of using complex strategies, such as line search, to find an update in the descent direction.

In gradient descent algorithms, an update step in the descent direction ensures that the cost is non-increasing at every iteration; but only this property is not enough to show that an algorithm is convergent. Therefore, in addition to proposing the L_q Weiszfeld algorithm, in this chapter, we identify the conditions under which descent algorithms are guaranteed to converge. Thus, any algorithm that satisfies these conditions converges is convergent. Our proof of the L_q Weiszfeld algorithm is also based on these conditions. Moreover, this enables us to propose several other generalizations of the Weiszfeld algorithm, such as a generalization to find the L_q mean of a set of points on a Riemannian manifold, and a generalization to find the L_q-closest-point to a set of affine subspaces in $\mathbb{R}^N$.

In the following section we propose the L_q Weiszfeld algorithm and list some properties of the L_q cost function for points in $\mathbb{R}^N$. Before actually proving the convergence of the L_q Weiszfeld algorithm we identify the conditions under which a gradient descent algorithm converges, in section 3.2. Based on these conditions we give a proof of convergence of the L_q Weiszfeld algorithm, in section 3.3. Our experimental results on synthetic data, in section 3.5, confirm the fact that the L_q mean, for $1 \leq q < 2$, is more robust to outliers than the L_2 mean.

3.1 L_q Weiszfeld Algorithm

In this section we propose the L_q Weiszfeld algorithm and list some properties of the L_q cost function. In the end, we state the convergence theorem for the L_q Weiszfeld algorithm; however, a detailed proof of the theorem is provided in section 3.3. Given a set of points

$\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, $k > 2$, in $\mathbb{R}^N$, the L_q cost function C_q is

$$C_q(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q = \sum_{i=1}^{k} \|\mathbf{x} - \mathbf{y}_i\|^q \,, \tag{3.1}$$

where $1 \leq q < 2$, $d(\cdot, \cdot)$ is the Euclidean distance and $\|\cdot\|$ is the Euclidean norm. For $q = 1$ the above problem is the same as the one solved by Weiszfeld [Weiszfeld, 1937], as described in chapter 2.

The primary purpose of this chapter is to prove the convergence of the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$. Although the case L_1 has received much attention in previous work, there are some advantages to considering the L_q case. First this gives a range of choices between the L_1 cost, favoured for its robustness to outliers, and the L_2 cost which is more theoretically justified statistically, assuming Gaussian noise. In addition, considering the L_q cost function avoids the difficulty that the L_1 cost is not differentiable. For $q > 1$ the cost function is differentiable everywhere. A final consideration is that the L_1 mean of a set of points may (with non-zero probability) coincide with one of the points themselves. This requires special care, since the minimum is then a point where the cost-function is non-differentiable. On the other hand, the L_q mean of a set of points will not generically coincide with any of the points, and besides, the cost function is differentiable everywhere.

3.1.1 Algorithm

Starting from an initial estimate $\mathbf{x}^0$, the L_q Weiszfeld algorithm generates a sequence of estimates $\mathbf{x}^t$ found by solving a weighted least-squares problem and the process is repeated until convergence. The L_q Weiszfeld algorithm differs from the L_1 algorithm mainly in the choice of weights applied at each step of iteration. In this case, the update equation of the Weiszfeld algorithm is with weights given by $w_i^t = d(\mathbf{x}^t, \mathbf{y}_i)^{q-2} = \|\mathbf{x}^t - \mathbf{y}_i\|^{q-2}$. With this update rule we shall a give proof of convergence of the L_q Weiszfeld algorithm in $\mathbb{R}^N$.

A current estimate $\mathbf{x}^t$ of the L_q minimum is updated to a new estimate

$$\begin{aligned} \mathbf{x}^{t+1} = W(\mathbf{x}^t) &= \frac{\sum_{i=1}^{k} w_i^t \mathbf{y}_i}{\sum_{i=1}^{k} w_i^t} \qquad \text{if } \mathbf{x}^t \notin \{\mathbf{y}_i\} \\ &= \mathbf{y}_j \qquad\qquad\qquad\quad \text{if } \mathbf{x}^t = \mathbf{y}_j \end{aligned} \,, \tag{3.2}$$

where

$$w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{q-2} \,.$$

Starting from a point $\mathbf{x}^0$, a sequence of points $(\mathbf{x}^t)$ is obtained using W as $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$. In section 3.3 we will show that the sequence of points $(\mathbf{x}^t)$ converges to the L_q minimum or it will stop at some point $\mathbf{x}^t = \mathbf{y}_i$.

Remarks: It is easy to see that W finds the exact minimizer of a weighted L_2 cost function

$C_2^{\mathbf{w}}$, defined as

$$C_2^{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{k} w_i^t \left\|\mathbf{x} - \mathbf{y}_i\right\|^2 ,$$

unless $\mathbf{x}^t$ is equal to some $\mathbf{y}_i$, where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{q-2}$. However, at this stage, it is not clear that the L_q cost function is a non-increasing function the under update function W. Since we are only interested in a decrease in the L_q cost function, in section 3.3.1 we will show that a decrease in the weighted L_2 cost function also results in a decrease in the L_q cost function (3.1). Therefore, the value of C_q decreases after every iteration of the L_q Weiszfeld algorithm. For details see section 3.3.1.

3.1.2 L_2 Mean for Points in $\mathbb{R}^N$

The L_2 version of the problem has a very simple form and can be computed in closed form. The L_2 mean of a set of points in $\mathbb{R}^N$ is the same as the arithmetic mean and is computed by taking the sum of points and dividing it by the total of points. Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ in $\mathbb{R}^N$, the L_2 cost function is defined as,

$$C_2(\mathbf{x}) = \sum_{i=1}^{k} \left\|\mathbf{x} - \mathbf{y}_i\right\|^2 ,$$

where $\|\cdot\|$ is the Euclidean norm.

The gradient of the above equation is,

$$\nabla C_2(\mathbf{x}) = \sum_{i=1}^{k} (\mathbf{x} - \mathbf{y}_i) .$$

By equating the gradient equal to zero, we get

$$\mathbf{x} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{y}_i .$$

The L_2 mean is not as robust to outliers as the L_q mean, even a single value of high magnitude can move the L_2 mean to an arbitrarily large value.

3.1.3 Properties of the L_q Cost Function for Points in $\mathbb{R}^N$

Here we explore and give proof of some of the properties of the L_q cost function for points in $\mathbb{R}^N$. Later on, these properties will be used to prove the convergence of the L_q Weiszfeld algorithm to the L_q mean of a given set of points in $\mathbb{R}^N$. The Gradient of the cost function (3.1) is

$$\nabla C_q = -\sum_{i=1}^{k} \|\mathbf{x} - \mathbf{y}_i\|^{q-2} (\mathbf{y}_i - \mathbf{x}) .$$

Just like the above equation of gradient, the rest of the thesis ignores a constant factor of q. Note that the L_q function is differentiable for $q > 1$, and for $q = 1$ it is non-differentiable when L_1 minimum is one of the given points $\mathbf{y}_i$.

Proposition 3.1. *The cost function C_q is strictly convex, except in the degenerate case when $q = 1$ and all points are collinear; in this case C_q is only convex.*

The proof of the above proposition is trivial and hence is omitted.

The minimum of C_q may be characterized in terms of a sub-gradient function defined as follows. If $\mathbf{x} = \mathbf{y}_i$ for some i, then $\nabla\widehat{C}_q(\mathbf{x})$ is the gradient of the cost function (3.1), omitting the term involving $\mathbf{y}_i$. Otherwise $\nabla\widehat{C}_q(\mathbf{x}) = \nabla C_q(\mathbf{x})$. Since $C_q(\mathbf{x})$ is differentiable when $q > 1$, it follows that $\nabla\widehat{C}_q(\mathbf{x}) = \nabla C_q(\mathbf{x})$ everywhere in this case.

Lemma 3.2. *A point $\mathbf{x}^* \in \mathbb{R}^N$ is a minimum of the cost function (3.1) if and only if it satisfies one of the following conditions:*

1. $\nabla C_q(\mathbf{x})$ *vanishes at* $\mathbf{x}^*$,
2. $q = 1$, $\mathbf{x}^* = \mathbf{y}_j$ *and* $\|\nabla\widehat{C}_q(\mathbf{x}^*)\| \leq 1$.

The proof of the lemma is simple. For $q > 1$ the cost function is differentiable, thus the gradient of the cost function vanishes at the minimum. However, for the case $q = 1$ see [Weiszfeld, 1937].

3.1.4 Convergence Theorem

We are now able to state a convergence theorem for the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$. Theorem 3.3 shows that the sequence of points $(\mathbf{x}^t)$ obtained using (3.2) converges to the minimum of (3.1) provided none of the intermediate iterates $\mathbf{x}^t$ equals any of the $\mathbf{y}_i$.

Theorem 3.3. *Given a set of non-collinear points $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ in $\mathbb{R}^N$, starting from a random point $\mathbf{x}^0 \in \mathbb{R}^N$, the sequence of points $(\mathbf{x}^t)$ obtained using (3.2) will either converge to the L_q minimum, or it will stop at some point $\mathbf{x}^t = \mathbf{y}_i$.*

Proof of this theorem is deferred until section 3.3.

The L_q Weiszfeld algorithm for points in $\mathbb{R}^N$ uses a gradient descent approach to find the L_q mean. In the case of gradient descent algorithm it is ensured that the cost function is non-increasing at every iteration. Clearly, this condition is not sufficient to ensure the convergence of a descent algorithm. Therefore, other conditions are also required to ensure the convergence of the algorithm. In the following section we identify the conditions under which descent algorithms are convergent. Later in the chapter we show that the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$ satisfies all these conditions and therefore converges to the L_q mean.

3.2 Convergence of Descent Algorithms

In this section we identify the conditions under which the convergence of a descent algorithm is guaranteed. Let W be an update function for which the value of a cost function C decreases at each iteration, except at fixed points where its value remains constant. Let $\mathbf{x}^0$ be a starting point of a descent algorithm, and $\mathbf{x}^{k+1} = W^k(\mathbf{x}^0)$ for $k = 1, 2, \ldots$. Of course simply because the value of the cost function decreases, that does not guarantee convergence of $(\mathbf{x}^k)$ even to a local minimum; other conditions are needed to ensure this. The Wolf conditions concerning sufficient step length [Nocedal and Wright, 1999] are one way to ensure convergence. A simpler but very general condition also guarantees convergence. Here we identify the conditions under which a descent algorithm converges.

Theorem 3.4 follows from the well known Global Convergence Theorem [Luenberger, 2003, section 6.6] applied to the special case of a single valued, continuous algorithm map.

Theorem 3.4. *Let D be a compact topological space, $C : D \to \mathbb{R}$ a continuous function defined on D. Let $W : D \to D$ be a continuous function with the property $C(W(\mathbf{x})) \leq C(\mathbf{x})$ for every $\mathbf{x} \in D$.*

Let $\mathbf{x}^0 \in D$, and $\mathbf{x}^{k+1} = W^k(\mathbf{x}^0)$ for $k = 1, 2, \ldots$. Then the sequence $(\mathbf{x}^k)$ converges to $S = \{\mathbf{x} \mid C(W(\mathbf{x})) = C(\mathbf{x})\}$, in the sense that if $\mathcal{O}$ is an open set containing S, then there exists an N such that $\mathbf{x}^k \in \mathcal{O}$ for all $k > N$.

Proof. Choose a starting point $\mathbf{x}^0 \in D$, and denote $\mathbf{x}^k = W^k(\mathbf{x})$ for $k > 0$. This theorem states that the sequence of iterates $\mathbf{x}^k$ converges to S, assuming only that the update rule $\mathbf{x}^k \mapsto \mathbf{x}^{k+1} = W(\mathbf{x}^k)$ is continuous on D and strictly decreasing, except on S.

Since D is compact, there exists a subsequence of $\mathbf{x}^k$ that is convergent. Let such a subsequence be $\mathbf{x}^{k_j}$; $j = 1, \ldots, \infty$ and let $\lim_j \mathbf{x}^{k_j} = \mathbf{x}$, which is a point in D. Then, $\lim_j C(\mathbf{x}^{k_j}) = C(\mathbf{x})$, since C is continuous, and $\lim_j C(W(\mathbf{x}^{k_j})) = C(W(\mathbf{x}))$, since $C \circ W$ is continuous.

Now, $C(\mathbf{x})$ is bounded below, for $\mathbf{x} \in D$, since D is compact. So, $C(\mathbf{x}^k)$ is a bounded non-increasing sequence in $\mathbb{R}$, and hence has a limit. Any subsequence of the $C(\mathbf{x}^k)$ must also have the same limit. In particular, $C(\mathbf{x}^{k_j})$ and $C(W(\mathbf{x}^{k_j}))$ are both subsequences of $C(\mathbf{x}^k)$ so

$$\begin{aligned} C(W(\mathbf{x})) &= \lim_j C(W(\mathbf{x}^{k_j})) = \lim_k C(\mathbf{x}^k) \\ &= \lim_j C(\mathbf{x}^{k_j}) = C(\mathbf{x}) \,. \end{aligned}$$

By the property of the function W, it follows that $\mathbf{x} \in S$. Since this argument holds for any convergent subsequence of $\mathbf{x}^k$, it show that any convergent subsequence converges to a point in S.

Now consider an open set $\mathcal{O}$ containing S. The theorem is proved by showing that at most a finite number of $\mathbf{x}^k$ lie outside of $\mathcal{O}$. Suppose the contrary, and hence that there is a subsequence $\mathbf{x}^{k_j}$ lying in $\bar{\mathcal{O}} = D - \mathcal{O}$. Since $\bar{\mathcal{O}}$ is a closed subset of a compact set, D, it is itself compact. Therefore, the sequence $\mathbf{x}^{k_j}$ must itself contain a convergent subsequence, and this subsequence converges to a point in $\bar{\mathcal{O}}$, and hence not in S. This is a contradiction, and the proof is complete. □

The above theorem shows that the sequence $(\mathbf{x}^k)$ converges to the set S but we are interested in the conditions under which the sequence $\mathbf{x}^k$ is convergent to a point. We show that under the conditions of Theorem 3.5 the sequence $(\mathbf{x}^k)$ is convergent. The condition in Theorem 3.5 is strictly weaker than the usually stated corollaries to the Global Convergence Theorem.

Theorem 3.5. *If in addition to Theorem 3.4, D is a metric space, S is a finite or countable set and $W(\mathbf{x}) = \mathbf{x}$ for all $\mathbf{x} \in S$, then the sequence $(\mathbf{x}^k)$ is convergent to a point in S.*

Proof. If $\mathbf{x}^k$ has an accumulation point $\mathbf{y}^*$, then $C(\mathbf{y}^*) = \lim_{k\to\infty} C(\mathbf{x}^k)$, so all accumulation points have the same cost. Furthermore $C(\mathbf{y}^*) = C(W(\mathbf{y}^*))$, so $\mathbf{y}^* \in S$. By the hypothesis of the theorem $W(\mathbf{y}^*) = \mathbf{y}^*$.

By assumption, D is compact. Let $\mathbf{y}_0^*$ be an accumulation point of $\mathbf{x}^k$. If $\mathbf{x}^k$ is not convergent, there exists $\epsilon > 0$ such that the sequence $\mathbf{x}^k$ enters and exits an open ball $B = B(\mathbf{y}_0^*, \epsilon)$ infinitely many times. There exists a subsequence $\mathbf{x}^{k_j}$ such that $\mathbf{x}^{k_j}$ lies inside this ball, whereas $\mathbf{x}^{k_j+1}$ lies outside. Again taking a subsequence, if necessary, it may be assumed that $\mathbf{x}^{k_j}$ converges to a point $\mathbf{y}_1^*$ and $\mathbf{x}^{k_j+1}$ converges to a point $\mathbf{y}_2^*$. Clearly, $d(\mathbf{y}_1^*, \mathbf{y}_0^*) \leq \epsilon$ and $d(\mathbf{y}_2^*, \mathbf{y}_0^*) \geq \epsilon$.

From the continuity of W, we have $W(\mathbf{x}^{k_j}) \to W(\mathbf{y}_1^*)$. However, also, $W(\mathbf{x}^{k_j}) = \mathbf{x}^{k_j+1} \to \mathbf{y}_2^*$ so $W(\mathbf{y}_1^*) = \mathbf{y}_2^*$. However, $W(\mathbf{y}_1^*) = \mathbf{y}_1^*$, so $\mathbf{y}_1^* = \mathbf{y}_2^*$, and $d(\mathbf{y}_1^*, \mathbf{y}_0^*) = \epsilon$.

The same thing holds for an open ball with any radius $\xi < \epsilon$; there must exist an accumulation point at any distance $\xi < \epsilon$ from $\mathbf{y}_0^*$, thus $\mathbf{x}^k$ has an uncountable number of accumulation points, and S is uncountable. □

Theorem 3.4 and Theorem 3.5 give simple but widely useful conditions for convergence of a descent algorithm. Later on, we will use these theorems to prove the convergence of the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$.

3.2.1 Example of Uncountably many Fixed Points

It is important to note here that if there are uncountably many fixed points of a function then a descent algorithm may not converge. We demonstrate this claim with the help of an example of a cost function that has uncountably many fixed points under an update function. We take an example of a cost function and an update function that satisfy the conditions of Theorem 3.4 and Theorem 3.5 but the set S is uncountable, that is, there are uncountably many fixed points. Under these conditions, we show that although the cost function is decreasing at every iteration of the algorithm and achieves a minimum value, but the sequence of iterates obtained using the update function do not converge. This shows that we need additional conditions to show that a descent algorithm is convergent, for example, the condition of Theorem 3.4 and Theorem 3.5.

For simplicity we define the update function W in polar coordinates (r, θ). Let $C(\mathbf{x})$ be a cost function defined as

$$C(\mathbf{x}) = \|\mathbf{x}\| = \sqrt{x^2 + y^2} = r\,, \tag{3.3}$$

where $x = r\cos(\theta)$, and $y = r\sin(\theta)$.

	Starting Point	Iteration 1	Iteration 2	Iteration 3	
r	1	$1+0$	$1+0$	$1+0$	$\cdots$
θ	θ	$\theta+\pi 0$	$\theta+\pi 0$	$\theta+\pi 0$	$\cdots$

Table 3.1: *Example of Uncountably many Fixed Points (Starting from a Fixed point): The above table shows the parameter values under the update function defined in (3.4). Note that when the starting point is one of the fixed points, that is a point on a unit circle, then the value of parameters remains the same after repetitive application of the update function W.*

	Starting Point	Iteration 1	Iteration 2		Iteration n
r	2	$1+(1/2)$	$1+(1/3)$	$\cdots$	$1+(1/n)$
θ	θ	$\theta+\pi\,(1/2)$	$\theta+\pi\,(1/2+1/3)$	$\cdots$	$\theta+\pi\ (\sum_{i=1}^{n} 1/i)$

Table 3.2: *Example of Uncountably many Fixed Points (Starting from a random point outside a unit circle): The above table shows the parameter values under the update function defined in (3.4). When starting point from a point outside a unit circle the value of r converges to* 1 *and the value of θ does not converge under the update function W. Note that the rate of change in the value of θ increases with increase in the number of iterations. Thus, under the update function W the cost function C in (3.3) is decreasing at every iteration, except on fixed points, but the algorithm does not converge.*

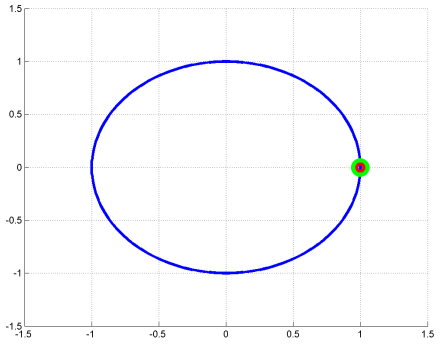
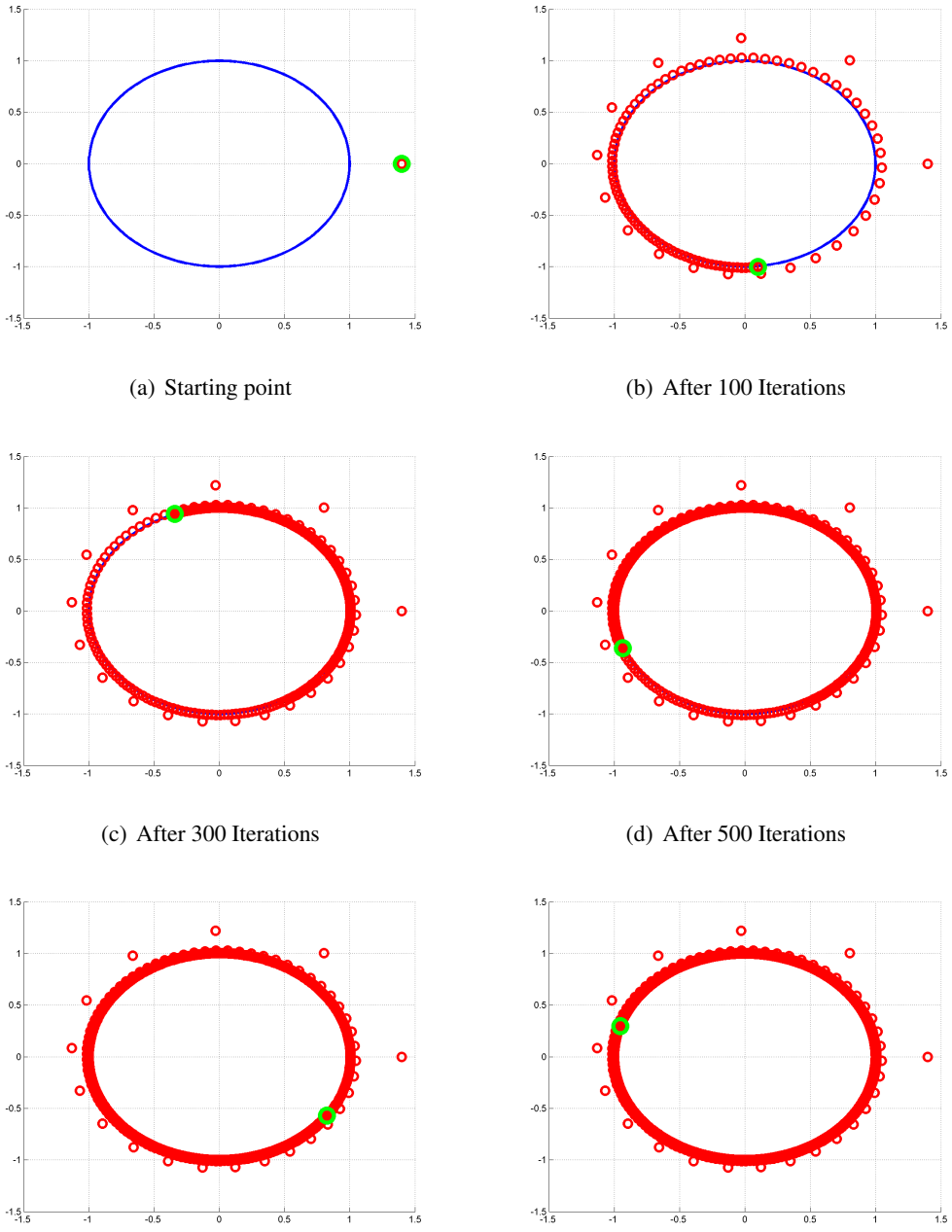


Figure 3.1: *Example of uncountably many fixed points (Starting from a Fixed point): Above figure shows a plot of fixed points (blue circle) of the cost function defined in (3.3) and the update function (3.4). Clearly, if we start from a (red) point on the (blue) circle the update function will map the red point to itself, since it is a fixed point.*



(a) Starting point

(b) After 100 Iterations

(c) After 300 Iterations

(d) After 500 Iterations

(e) After 1000 Iterations

(f) After 3000 Iterations

Figure 3.2: *Example of uncountably many fixed points (Starting from a general point outside unit circle): In each of the above figures the blue circle represents a set of fixed points of the cost function defined in (3.3) under the update function (3.4). Here we show that if we start from a (red) point that does not lie on the blue circle, set of fixed points. Then the sequence of (red) points obtained by using the update function (3.4) is not convergent, as shown in the above plots.*

Let $[r^0\ \theta^0]^T$ be a starting point of the algorithm. An update function W is defined as

$$\begin{bmatrix} r^{t+1} \\ \theta^{t+1} \end{bmatrix} = W\left(\begin{bmatrix} r^t \\ \theta^t \end{bmatrix}\right) = \begin{bmatrix} 1 + f(r^t) \\ \theta^t + \pi f(r^t) \end{bmatrix}, \tag{3.4}$$

where,

$$f(r) = \frac{|r-1|}{1+|r-1|},$$

where r^t and θ^t are estimates at iteration t. Under the update function W the cost function C is a non-increasing function.

Note that the value of C decreases after every iteration, except when starting point is one of the fixed points. The set of fixed points of C under the update function W is a unit circle, that is, $[1\ \ \theta]^T$, for all values of θ. Thus, if we start from a point on the circle of radius of 1, the starting point is also the point of convergence, since it is a fixed point, as shown in fig. 3.1. A Few iterations of the algorithm are shown in Table 3.1, where the starting point is on unit circle and under the update function W the point is not changing its position.

However, if we start from any point outside the unit circle the cost function C decreases under the update function W and the sequence of iterates get closer to the unit circle but the sequence of points do not converge. In this case the change in angle becomes higher as the iterates get closer to the unit circle. The value of radius converges to 1 but the value of angle keeps changing, preventing the sequence of iterates to converge, as shown in fig. 3.2. Several iterations of the algorithm are shown in Table 3.2 where the starting point is $[2\ \ \theta]^T$. It shows that the value of r converges to 1 but the rate of change of θ becomes higher as the iterates get closer to the unit circle. Thus, the sequence of points is not convergent.

This shows that even if the value of a cost function decreases at every iteration of an algorithm and converges to a value, the convergence of the algorithm is not guaranteed. In Theorem 3.5 we have shown that if there are countably many or finite fixed points the sequence is convergent.

3.3 Proof of Convergence of the L_q Weiszfeld Algorithm

In this section we give a proof of convergence of the L_q Weiszfeld algorithm, specifically, a proof of Theorem 3.3. It will be shown later in this section that the L_q cost function is a non-increasing function under the update function W in (3.2). As mentioned before that this condition, alone, is not enough to ensure the convergence of the algorithm. Thus, we need to verify the conditions of Theorem 3.4 and Theorem 3.5 for the L_q cost function C_q and the update function W defined in (3.1) and (3.2), respectively, to prove the convergence of the L_q Weiszfeld algorithm to the L_q mean.

Let $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ be a set of non-collinear points in $\mathbb{R}^N$ and $\mathbf{x}^0$ be an an initial estimate

of the L_q minimum. At iteration t, a current estimate of the L_q minimum is updated as

$$\begin{aligned}\mathbf{x}^{t+1} = W(\mathbf{x}^t) &= \frac{\sum_{i=1}^{k} w_i^t \mathbf{y}_i}{\sum_{i=1}^{k} w_i^t} \qquad \text{if } \mathbf{x}^t \notin \{\mathbf{y}_i\} \\ &= \mathbf{y}_j \qquad\qquad\qquad \text{if } \mathbf{x}^t = \mathbf{y}_j \,,\end{aligned}$$

where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{q-2}$. Theorem 3.3 shows that the sequence of points $(\mathbf{x}^t)$ obtained using the above equation, as defined in (3.2), converges to the minimum of C_q (3.1) provided none of the intermediate iterates $\mathbf{x}^t$ equals any of the $\mathbf{y}_i$.

Now we list some basic steps based on Theorem 3.4 and Theorem 3.5 to prove the convergence of the sequence $(\mathbf{x}^t)$ obtained in Theorem 3.3.

Outline 3.1. *Given an update function W and a strictly convex function C_q, to prove that the sequence $(\mathbf{x}^t)$ obtained using $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$ is convergent to the minimum of C_q we proceed as follows:*

1. *The update function W is extendible by limits to a continuous, defined on a compact domain $\mathcal{D}$ and maps $\mathcal{D}$ to itself.*
2. *The value of C_q is non-increasing at every iteration.*
3. *The set S of Theorem 3.4 is a finite set containing the L_q minimum point and $\{\mathbf{y}_i\}$.*
4. *Since there is a finite number of accumulation points, the sequence $(\mathbf{x}^t)$ is in fact convergent, see Theorem 3.5.*
5. *If $(\mathbf{x}^t)$ converges to one of the given points, $\mathbf{y}_j$, then this is the minimum, except when $\mathbf{x}^t \in \{\mathbf{y}_i\}$ for any of the intermediate iterates.*
6. *Therefore, unless $\mathbf{x}^t$ gets stuck at $\mathbf{y}_i$, it converges to the L_q minimum.*

Proof of Theorem 3.3: The above discussion simplifies the proof of Theorem 3.3 and we only need to verify that the cost function C_q and the update function W satisfy all the conditions of the steps of Outline 3.1. Below we verify each step of Outline 3.1. The proof of Theorem 3.3 is complete once all the steps are verified.

1. The compact domain $\mathcal{D}$ is defined as $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^N \mid C_q(\mathbf{x}) \leq C_q(\mathbf{x}^0)\}$, where $\mathbf{x}^0$ is a random starting point. It is easy to see that the update function W (3.2) is continuous at $\mathbf{x} \neq \mathbf{y}_i$, and is extendible to a continuous function by taking the limit at each $\mathbf{y}_i$. It is shown in section 3.3.1 that W results in a non-increasing cost C_q; the value of $C_q(\mathbf{x}^t)$ will never be greater than $C_q(\mathbf{x}^0)$. Thus, from the definition of $\mathcal{D}$ the update function W maps a point in $\mathcal{D}$ to itself.
2. The cost function C_q is a non-increasing function under the update function W, that is $C_q(W(\mathbf{x})) \leq C_q(\mathbf{x})$ with equality only when $W(\mathbf{x}) = \mathbf{x}$, see lemma 3.7.
3. Under the update function W the value of C_q remains constant in successive iterations either when $\mathbf{x} = \mathbf{y}_i$ where $W(\mathbf{y}_j) = \mathbf{y}_j$, or when $\mathbf{x}$ is a minimum of C_q. Therefore the set S in Theorem 3.4 is a union of stationary points of C_q and $\{\mathbf{y}_i\}$.

4. The cost function C_q being strictly convex has a unique stationary point $\mathbf{x}^*$. The set S is a finite set because its a union of $\{\mathbf{y}_i\}$ and $\mathbf{x}^*$. Thus, from Theorem 3.5 every sequence $(\mathbf{x}^t)$ is convergent to $S = \{\mathbf{x}^*\} \cup \{\mathbf{y}_i\}$.

5. When $\mathbf{x}^t$ converges to a point other than $\{\mathbf{y}_i\}$ then from Theorem 3.5 the sequence of points $\mathbf{x}^t$ converges to $\mathbf{x}^*$, that is the L_q minimum. However, if $\mathbf{x}^t$ converges to $\mathbf{y}_j$ such that $\mathbf{x}^t \notin \{\mathbf{y}_i\}$ for $i \neq j$, then $\mathbf{y}_j$ is also the stationary point of C_q, see lemma 3.8.

Thus, the proof relies on lemma 3.7 and lemma 3.8 which are proven in the subsequent sections. This proves our claim that the sequence of points $(\mathbf{x}^t)$ converges to the L_q minimum, except when some $\mathbf{x}^t \in \{\mathbf{y}_i\}$ for any of the intermediate iterates. □

In the rest of this section we show that under the update function W in (3.2) the value of C_q decreases after every iteration that if the sequence of points $(\mathbf{x}^t)$ is convergent to $\mathbf{y}_j$ then this point satisfies the conditions of lemma 3.2 and is therefore the L_q minimum.

3.3.1 Decreasing L_q Cost for Points in $\mathbb{R}^N$

Here we show that the value of C_q decreases at every iteration, except when $W(\mathbf{x}) = \mathbf{x}$. Let $C_2^{\mathbf{w}}$ be a weighted L_2 function, defined as

$$C_2^{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{k} w_i^t \|\mathbf{x} - \mathbf{y}_i\|^2 ,$$

where $w_i^t = \|\mathbf{x}^t - \mathbf{y}_i\|^{q-2}$, $\mathbf{x}^t$ is an estimate of the L_q minimum at iteration t. Note that the update function (3.2) finds the exact minimizer of $C_2^{\mathbf{w}}$, unless $\mathbf{x}^t$ is equal to some $\mathbf{y}_i$.

In preparation for proving lemma 3.7, the following general result establishes the relation between an L_q cost for $1 \leq q < 2$ and a weighted L_2 cost.

Lemma 3.6. *If a_i and b_i are positive real numbers, $0 < q < n$ and $\sum_{i=1}^k a_i^{q-n} b_i^n \leq \sum_{i=1}^k a_i^q$ then $\sum_{i=1}^k b_i^q \leq \sum_{i=1}^k a_i^q$ with equality only when $a_i = b_i$ for all i.*

Proof. The proof of above lemma depends on the following simple but critical observation, which was implicitly stated (albeit in less generality) in [Weiszfeld, 1937]. Consider the following self-evident statement.

- *Let g be a convex function and $0 < q < n$. If $g(n) \leq g(0)$ then $g(q) \leq g(0)$ with strict inequality if g is strictly convex.*

This statement will now be applied to the function

$$g(n) = \sum_{i=1}^{k} a_i^{q-n} b_i^n ,$$

to prove the above lemma. To show this, one computes the second derivative of g with respect to n. The result is

$$g''(n) = \sum_{i=1}^{k} a_i^{q-n} \, b_i^n \, (\log a_i - \log b_i)^2 .$$

Since all the a_i and b_i are positive, this second derivative is positive unless $\log a_i = \log b_i$ for all i, which proves the lemma. □

Note that the above lemma is true for $0 < n < \infty$ and $q < n$ and is not limited to $n = 2$ and $1 \leq q < 2$. We imposed a restriction to $n = 2$ for the problems in this thesis to take advantage of the simple solutions of the least squares problems, but this result holds for a general value n.

Applying this lemma to (3.2) shows that W results in a decreased cost.

Lemma 3.7. *For the update function W in (3.2), $C_q(W(\mathbf{x})) \leq C_q(\mathbf{x})$, where equality holds only when $W(\mathbf{x}) = \mathbf{x}$.*

Proof. To be able to apply lemma 3.6 to a L_q function C_q, and a weighted L_2 function $C_2^{\mathbf{w}}$, we proceed as follows. Let $a_i = d(\mathbf{x}, \mathbf{y}_i)$, $b_i = d(W(\mathbf{x}), \mathbf{y}_i)$, and $w_i = d(\mathbf{x}, \mathbf{y}_i)^{q-2}$ in lemma 3.6. Since $W(x)$ finds the global minimum of the weighted L_2 cost function $C_2^{\mathbf{w}}$, the following relation holds:

$$\sum_{i=1}^{k} \frac{d(W(\mathbf{x}), \mathbf{y}_i)^2}{d(\mathbf{x}, \mathbf{y}_i)^{2-q}} = C_2^{\mathbf{w}}(W(\mathbf{x})) \leq C_2^{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q \, .$$

Then from lemma 3.6 it follows that

$$C_q(W(\mathbf{x})) = \sum_{i=1}^{k} d(W(\mathbf{x}), \mathbf{y}_i)^q \leq \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q = C_q(\mathbf{x}) \, ,$$

and equality holds only when $W(\mathbf{x}) = \mathbf{x}$. □

3.3.2 L_q Minimum on Given Points:

Now, we show that when $\mathbf{x}^t$ converges to one of the given points $\mathbf{y}_j$ without getting stuck at $\mathbf{y}_i$, then it satisfies the conditions to be the L_q minimum mentioned in lemma 3.2. Note that when any of the intermediate iterates is equal to one of the given points $\mathbf{y}_i$ then the sequence $\mathbf{x}^t$ gets stuck at that point. Therefore, when minimum point is one of the given points and none of the intermediate iterates lands on any of the given points then we show that the accumulation point of the sequence is the L_q minimum and satisfies the minimum point condition stated in lemma 3.2. When a minimum point is not one of the given points $\mathbf{y}_j$ then the L_q function is differentiable at the minimum point, even for $q = 1$. However, when the L_q minimum is one of the points $\mathbf{y}_j$ then the following lemma shows that for $1 < q < 2$ the gradient of the cost function vanishes at this point, while for $q = 1$ the gradient of the L_q function omitting the entry corresponding to x_j has a norm no greater than one.

Lemma 3.8. *For $1 \leq q < 2$, if the limit of the sequence $(\mathbf{x}^t)$ is one of the points $\mathbf{y}_j$, then $\mathbf{y}_j$ is the minimum point of C_q, except when any of the intermediate iterates $\mathbf{x}^t$ is equal to one of the $\mathbf{y}_i$ and the iteration gets stuck.*

Proof. Suppose that the sequence $\mathbf{x}^t$ converges to one of the points $\mathbf{y}_i$, which we take to be $\mathbf{y}_1$ for simplicity. Our goal is to show that for $q > 1$ the gradient $\nabla C_q = \sum_{i=1}^{k} w_i^t \, (\mathbf{x} - \mathbf{y}_i)$

vanishes at $\mathbf{y}_1$ and for $q = 1$ the magnitude of gradient excluding $\mathbf{y}_1$ is no greater than 1, see lemma 3.2.

The update function is defined as

$$\mathbf{x}^{t+1} = \frac{\sum_{i=1}^{k} w_i^t \, \mathbf{y}_i}{\sum_{i=1}^{k} w_i^t} \,,$$

where $w_i^t = \|\mathbf{y}_i - \mathbf{x}^t\|^{q-2}$ and $\mathbf{x}^t$ is an estimate of L_q minimum at iteration t.

By a small rearrangement one sees that

$$(\mathbf{x}^{t+1} - \mathbf{y}_1)\, w_1^t = \sum_{i=2}^{k} w_i^t \, \mathbf{y}_i - \mathbf{x}^{t+1} \sum_{i=2}^{k} w_i^t \,.$$

In the limit as $t \to \infty$ this yields

$$\begin{aligned} \lim_{t\to\infty} (\mathbf{x}^{t+1} - \mathbf{y}_1)\, w_1^t &= \sum_{i=2}^{k} \frac{\mathbf{y}_i}{\|\mathbf{y}_1 - \mathbf{y}_i\|^{2-q}} - \sum_{i=2}^{k} \frac{\mathbf{y}_1}{\|\mathbf{y}_1 - \mathbf{y}_i\|^{2-q}} \\ &= \sum_{i=2}^{k} \frac{\mathbf{y}_i - \mathbf{y}_1}{\|\mathbf{y}_1 - \mathbf{y}_i\|^{2-q}} \,. \end{aligned} \tag{3.5}$$

Note that this is the gradient $\nabla \widehat{C}_q(\mathbf{y}_1)$ where $\widehat{C}_q$ is the L_q cost function without the contribution of the term for $i = 1$. If $q > 1$, then $\widehat{C}_q(\mathbf{y}_1) = C_q(\mathbf{y}_1)$, since the missing term in $C_q(\mathbf{x})$ is $\|\mathbf{x} - \mathbf{y}_1\|^q$ which has zero gradient at $\mathbf{x} = \mathbf{y}_1$.

Now, still under the assumption that $\mathbf{x}^t \to \mathbf{y}_1$, from the left side of this expression one obtains

$$\begin{aligned} \|\nabla \widehat{C}_q(\mathbf{y}_1)\| &= \left\| \lim_{t\to\infty} (\mathbf{x}^{t+1} - \mathbf{y}_1)\, w_1^t \right\| \\ &= \lim_{t\to\infty} \|\mathbf{x}^{t+1} - \mathbf{y}_1\| \, \|\mathbf{x}^t - \mathbf{y}_1\|^{q-2} \\ &= \lim_{t\to\infty} \frac{\|\mathbf{x}^{t+1} - \mathbf{y}_1\|}{\|\mathbf{x}^t - \mathbf{y}_1\|} \, \|\mathbf{x}^t - \mathbf{y}_1\|^{q-1} \,. \end{aligned} \tag{3.6}$$

Now, the cases $q = 1$ and $q > 1$ must be dealt with differently because when $q > 1$ the cost function C_q is differentiable at $\mathbf{y}_1$, whereas when $q = 1$ it is not. We use a simple observation about convergent sequences, stated here without proof.

Lemma 3.9. *Let $(\mathbf{x}^t)$ be a sequence in a metric space converging to $\mathbf{y}$. Then*

$$\lim_{t\to\infty} d(\mathbf{x}^{t+1}, \mathbf{y}) / d(\mathbf{x}^t, \mathbf{y}) \leq 1 \,,$$

if the limit exists. If g^t is a sequence of real numbers such that $g^t \to 0$, then

$$\lim_{t\to\infty} g^t \, d(\mathbf{x}^{t+1}, \mathbf{y}) / d(\mathbf{x}^t, \mathbf{y}) = 0 \,,$$

if the limit exists.

Of course, the given limits may not exist in either case. In applying this lemma to the right hand side of (3.6), however, the limit is known to exist and equal $\|\nabla \widehat{C}_q(\mathbf{y}_1)\|$.

Consider the case $q > 1$. Then in (3.6) the term $\|\mathbf{x}^t - \mathbf{y}_1\|^{q-1}$ converges to zero, since $\mathbf{x}^t \to \mathbf{y}_1$. It follows from lemma 3.9 and (3.6) that

$$\|\nabla C_q(\mathbf{y}_1)\| = \|\nabla \widehat{C}_q(\mathbf{y}_1)\| = 0$$

so $\mathbf{y}_1$ is a stationary point (hence global minimum) of C_q.

In the case $q = 1$, lemma 3.9 and (3.6) yield

$$\|\nabla \widehat{C}_q(\mathbf{y}_1)\| = \lim_{t\to\infty} \frac{\|\mathbf{x}^{t+1} - \mathbf{y}_1\|}{\|\mathbf{x}^t - \mathbf{y}_1\|} \leq 1$$

which is the condition given in lemma 3.2 for $\mathbf{y}_1$ to be a minimum of the cost function. □

This completes the proof of Theorem 3.3 that the L_q Weiszfeld algorithm either converges to the L_q mean or stops at a point $\mathbf{y}_i$.

3.4 Discussion

Generally a randomly selected starting point will result in convergence of the L_q Weiszfeld algorithm to the L_q minimum without getting stuck at $\mathbf{y}_i$. However, if such a condition occurs where $\mathbf{x}^t = \mathbf{y}_j$ then the L_q Weiszfeld algorithm, even the Weiszfeld algorithm, gets stuck at that point. A simple strategy to escape this situation is to move the current solution $\mathbf{x}^t$ in the descent direction and continue with the algorithm. Since the ambient space for these problems is high, this condition is not very likely to occur. However, this situation can be avoided by a careful selection of a starting point $\mathbf{x}^0$, as explained below in Algorithm 3.1.

Algorithm 3.1. *Given a set of points, $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\} \in \mathbb{R}^N$ or $\mathcal{M}$, $k > 1$. Let $d(\mathbf{y}_i, \mathbf{y}_j)$, be the distance between two points, $\mathbf{y}_i$ and $\mathbf{y}_j$.*

1. *Among the $\mathbf{y}_i$ select the one with minimum cost:*

$$\mathbf{x}^* = \operatorname*{argmin}_j C_q(\mathbf{y}_j)\ ,$$

   where $C_q(\mathbf{x}) = \sum_{i=1}^k d(\mathbf{x}, \mathbf{y}_i)^q$.

2. *Compute the gradient of C_q and check $\mathbf{x}^* = \mathbf{y}_j$ for the minimality condition according to lemma 3.2. If it satisfies the condition then $\mathbf{x}^*$ is the required minimum and the algorithm is complete.*

3. *Otherwise, displace $\mathbf{x}^*$ in the downhill gradient direction to obtain $\mathbf{x}^0$. Backtrack if necessary to ensure $C_q(\mathbf{x}^0) < C_q(\mathbf{x}^*)$.*

4. *Repeat, $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$ until convergence, where W is defined in (3.2).*

The initial point $\mathbf{x}^0$ so found has cost less than any of the points $\mathbf{y}_i$, and iterations of the algorithm from $\mathbf{x}^0$, can not again approach any of the $\mathbf{y}_j$. Thus Algorithm 3.1 ensures that the non-differentiability conditions never occur by a careful selection of a starting point for the algorithm. Hence the L_q Weiszfeld algorithm is guaranteed to converge to the minimum of $C_q(\mathbf{x})$.

Theorem 3.10. *The Algorithm 3.1 converges to the desired L_q minimum.*

The proof of above theorem follows trivially from the proof of Theorem 3.3.

3.5 Experiments

In order to show the applicability of the proposed algorithm we perform averaging over a synthetic data of a set of points in $\mathbb{R}^3$. We try to estimate a set of 50 points, where each point is averaged from its 100 noisy samples by using the L_2 averaging and L_q averaging methods. In all of our experiments we assume that the ground truth is known and errors are computed using the ground truth values. These experiments are designed to demonstrate the robustness of the L_q averaging method against outliers. These experiments confirm the fact that the L_q averaging method is more robust to outliers than the L_2 averaging method. In our experiments we show the following results:

1. *Convergence behavior:* This experiment shows the convergence behavior of the L_1 technique. Results have shown that the L_1 averaging method converges close to the actual solution with in the first few iterations of the algorithm. Therefore, even a few iterations of the algorithm are enough to obtain a more robust solution compared to the L_2 mean.

2. *Robustness against different number of Outliers:* This experiment is designed to compare the robustness of the L_q averaging methods for different values of q ranging from 1 to 2. In order to show the robustness against outliers we add different proportion of outliers in the data. Our results have shown that the output of the L_1 technique does not change much with change in the amount of outliers and is therefore more robust to outliers than the rest of the L_q methods, for $1 < q \leq 2$.

3. *Relative Position of Points in $\mathbb{R}^3$:* In this experiment we show the location of the L_1 mean, L_2 mean and ground truth with respect to the given points, in $\mathbb{R}^3$. This experiment is designed to show the actual spread of the data points and location of the results of the averaging methods in 3D space.

In all of the above mentioned scenarios our experimental results confirm that the L_q averaging method gives superior results than the L_2 averaging method in the presence of outliers. However, when there are no outliers in data, a condition that rarely occurs, the results of both the techniques are roughly the same.

Data Points: In our experiments we take a random 3D vector $\hat{\mathbf{x}}$ and generate several samples by adding noise to it. In all cases a Gaussian noise of zero mean and a standard deviation of

$\|\hat{\mathbf{x}}\|$ is added to the point, where $\|\cdot\|$ is the L_2 norm. Thus, a set of random points is generated by using the following equation

$$\mathbf{y} = \hat{\mathbf{x}} + N(0, \|\hat{\mathbf{x}}\|)\,, \tag{3.7}$$

where $N(\mu,\sigma)$ is a normal distribution with mean μ and standard deviation σ. We perform averaging for a set of 50 points, where each point is averaged from its 100 noisy samples. The percentage of outliers vary in different experiments.

Error Measure: We report an RMS error with with respect to the known ground truth. Let $\mathbf{x}_i$ be a known ground truth values and $\mathbf{y}_i$ be estimated values, then the RMS error is computed as,

$$\text{RMS Error} = \sqrt{\textstyle\sum_{i=1}^{k} \|\mathbf{x}_i - \mathbf{y}_i\|/k}\,, \tag{3.8}$$

where $\|\cdot\|$ is the L_2 norm and k is the total number of points.

Starting Point and Number of Iterations: A maximum of 30 iterations is allowed for the L_q averaging method because even after first 10 to 15 iterations of the L_q algorithm, the change in error is very less and algorithm becomes stable. The proposed L_q averaging algorithm being an iterative optimization technique requires a starting point. In our experiments we use a random point as a starting point for the L_q averaging method. However, a slightly better option is to use the L_2 mean as a starting point, that can be computed in closed form.

In the rest of the section we will discuss our experimental results.

3.5.1 Convergence Behavior

Here we show the convergence behavior of the L_1 averaging algorithm. We perform averaging for a set of 50 points, where each point is averaged from its 100 noisy samples. In order to generate noisy samples for each point, we add a Gaussian noise of zero mean and standard deviation of the magnitude of the point according to the criteria defined before in (3.7). In addition to noise in data we modify 30% of the data points to represent outliers. In this experiment we take a random point as a starting point of the L_1 averaging algorithm.

Fig. 3.3 shows the results of the L_2 method and several iterations of the L_1 averaging algorithm in the presence of outliers. In fig. 3.3 the x-axis represents iterations and y-axis represents the RMS error, where error is computed using (3.8). It is evident from the plots that the L_1 mean is closer to the known ground truth value than the L_2 mean.

We allow the L_1 algorithm to execute for 30 iterations but the change in error is very less after 15 iterations, as shown in fig. 3.3. Note that even after the few first iterations of the L_1 algorithm, the value of the estimated L_1 mean is significantly closer to the ground truth than the L_2 mean. Thus, if efficiency is of primary concern then even a few iterations of the L_1 algorithm are better than the L_2 averaging algorithm to get robust results.

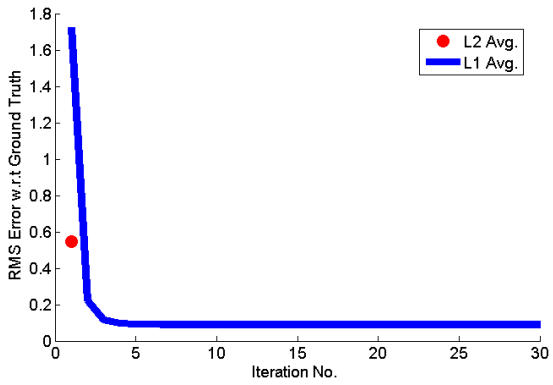


Figure 3.3: *Convergence Behavior: In the above figure the result of the* L_2 *averaging method is represented by a (red) point and the results of the* L_1 *averaging method is represented by the blue line. Errors are computed with respect to the known ground truth and are plotted against iterations. We perform averaging for a set of* 50 *points, where each point is averaged from its* 100 *noisy samples. In order to generate noisy samples for each point, we add a Gaussian noise of zero mean and standard deviation of the magnitude of the point according to the criteria defined in (3.7). We also modify* 30% *of the samples to represent outliers. The above plots confirm the fact that in the presence of outliers the* L_1 *method gives superior results than the* L_2 *method. Note that the* L_1 *algorithm stabilizes in* 10 *to* 15 *iterations.*

3.5.2 Robustness Against Different Number of Outliers

In this experiment we compare the results of the L_q averaging algorithm for several values of q ranging from 1 to 2. We add different number of outliers to the data to observe the robustness of the L_q averaging algorithms against outliers. Same as before, we perform averaging for a set of 50 points, where each point is averaged from its 100 noisy samples. In order to generate noisy samples for each point, we add a Gaussian noise of zero mean and standard deviation of the magnitude of the point according to the criteria defined before in (3.7). However, the number of outliers in the data is variable and its value ranges from 0% to 40%, with an increment of 10%. Finally, an RMS error between the estimated mean value and the known ground truth value is computed using (3.8).

It is obvious from fig. 3.4 that in the presence of outliers the estimated L_1 mean is closer to the ground truth than the rest. Furthermore, the results of the L_1 averaging does not vary much with a change in the percentage of outliers in data. Thus, the L_1 averaging method is more robust to outliers than the other L_q averaging methods, for $1 < q \leq 2$. The only point where the results of the L_2 averaging method is closer to the ground truth than the rest of the methods is when there are no outliers in data, as shown by the left-most set of bars in fig. 3.4. Thus, in the absence of outliers the L_2 algorithm is recommended because it finds the solution in closed form. But in practice, having outlier free data is not likely, therefore the proposed L_1 averaging algorithm is very practical.

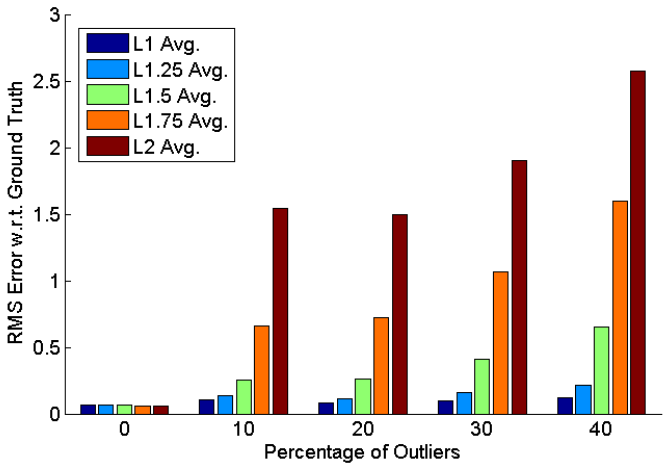


Figure 3.4: *Robustness against different number of outliers: In the above figure the results of the* L_q *averaging algorithms for several values of* q *are compared, specifically, we consider* L_1, $L_{1.25}$, $L_{1.5}$, $L_{1.75}$ *and* L_2 *averaging methods. We perform averaging for a set of* 50 *points, where each point is averaged from its* 100 *noisy samples. In order to generate noisy samples for each point, we add a Gaussian noise of zero mean and standard deviation of the magnitude of the point according to the criteria defined before in (3.7). The number of outliers is varied and is indicated by the* x*-axis of the above figure. Errors are computed with respect to the known ground truth values. The above plot shows that in the presence of outliers the* L_1 *mean is closer to the ground truth than the rest of the* L_q *averaging methods, for* $1 \leq q < 2$*. However, when there are no outliers in data then the results of all of the algorithms are roughly the same. Thus, in the presence of outliers the* L_1 *averaging method is recommended.*

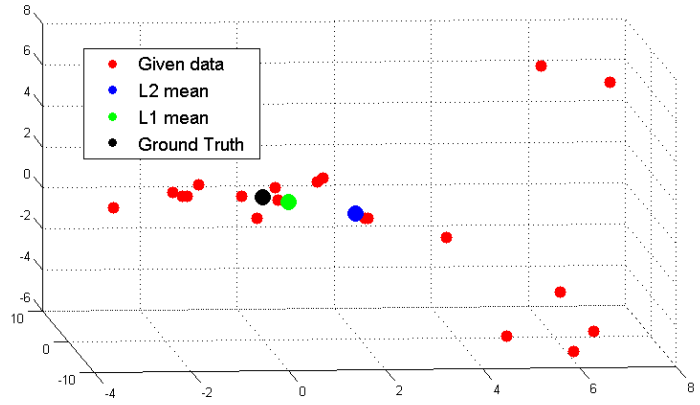


Figure 3.5: *Relative Position of Points in* $\mathbb{R}^3$*: the above figure shows a plot of the* L_1 *mean (green point) and* L_2 *mean (blue point) relative to the ground truth (black point) and sample points (red points). In this case, only a single point is averaged from its* 20 *noisy samples. In order to generate the sample points, we take a random point as the ground truth and add noise to it according to the criteria defined before in (3.7), that is a Gaussian noise of zero mean and standard deviation of the magnitude of the point. In addition to the noise in data we modify* 30% *of the data points to represent outliers. In the above figure red points represent given points, and the ground truth is represented by a black point. The above figure confirms the fact that in the presence of outliers the* L_1 *mean (represented by a green point) is closer to the ground truth than the* L_2 *mean (represented by a blue point).*

3.5.3 Relative Position of Points in $\mathbb{R}^3$

This experiment is designed to show the location of the L_1 mean and L_2 mean relative to the ground truth and sample points in 3D space. In this case, only a single point is averaged from its 20 noisy samples. In order to generate the sample points, we take a random point as the ground truth and add noise to it according to the criteria defined before in (3.7), that is a Gaussian noise of zero mean and standard deviation of the magnitude of the point. In addition to the noise in data we modify 30% of the data points to represent outliers. In this case we only perform averaging over a set of 20 points. It can easily be seen in fig. 3.5 that the L_1 mean (green point) is closer to the ground truth (black point) than the L_2 mean (blue point).

3.6 Summary

In summary, in this chapter we proposed a Weiszfeld-style algorithm to find the L_q mean of a set of points in $\mathbb{R}^N$, for $1 \leq q < 2$. We referred to it as the L_q Weiszfeld algorithm. We gave a proof for the convergence of the L_q Weiszfeld algorithm to the L_q mean and showed that the L_q mean can be found by iteratively minimizing a weighted L_2 function. Ease of implementation makes the proposed algorithm attractive wherever L_q optimization is desired. Furthermore, our experimental results showed that the L_q optimization method, for $1 < q \leq 2$, gives superior results to the L_2 method, in terms of robustness to outliers.

In addition to the proof of convergence of the L_q Weiszfeld algorithm, we identified the

conditions under which a descent algorithm converges. Our proof of the L_q Weiszfeld algorithm is based on these condition. Furthermore, in the later chapters these conditions will also be used to prove the convergence of other extensions of the L_q Weiszfeld algorithm, that is, the L_q Weiszfeld algorithm for points on a positively curved Riemannian manifold, and a L_q Weiszfeld inspired method to find the L_q-closest point to affine subspaces.

Chapter 4

Lq Optimization for Points on a Riemannian Manifold

In this chapter we propose an algorithm, based on the L_q Weiszfeld algorithm, to find the L_q mean, for $1 \leq q < 2$, of a set of points on a Riemannian manifold of non-negative sectional curvature. In addition to the proof of convergence of the proposed algorithm, we show that the bounds on the maximum distance between points on manifold can further be relaxed than the bounds in the existing techniques to find the L_1 mean [Fletcher et al., 2009]. Just like the L_q Weiszfeld algorithm, the proposed generalization also inherits all of the advantages of the Weiszfeld algorithm, such as updates are computed analytically, guaranteed convergence to the L_q mean, etc. The proof of convergence of the proposed algorithm is similar to the proof of the L_q Weiszfeld algorithm in chapter 3. We apply the proposed algorithm to the problem of rotation avearging and averaging of Symmetric Positive-Definite (SPD) matrices. Our experimental results, on both real and synthetic data, confirm the fact that the L_q mean, for $1 \leq q < 2$, is more robust to outliers than the L_2 mean.

4.1 Introduction / Literature Review

This chapter describes a very simple iterative and provably convergent algorithm for L_q optimization for points on a Riemannian manifold of non-negative curvature, and applies it to different problems. In addition to the theoretical proof of convergence of the L_q Weiszfeld algorithm we apply the proposed algorithm to several problems. We consider in detail two major problems, that is, L_q rotation averaging and L_q averaging for SPD matrices. The problem of L_q rotation averaging takes two forms: single rotation averaging in which several estimates of a single rotation are averaged to give the best estimate; and multiple rotation averaging, in which relative rotations $\mathtt{R}_{ij}$ are given, and absolute rotations $\mathtt{R}_i$ are computed to satisfy the compatibility constraint $\mathtt{R}_{ij}\mathtt{R}_i = \mathtt{R}_j$. We apply the L_q Weiszfeld algorithm to both single-rotation averaging (under which the algorithm provably finds the global L_q optimum) and multiple rotation averaging (for which no such proof exists). We also consider the problem of averaging of SPD matrices under different metrics, for example, the Log-Euclidean metric and the Affine-Invariant metrics. The space of SPD matrices is a flat manifold when embedded with the Log-Euclidean metric and has a negative sectional curvature under the Affine-Invariant metric. Therefore, the convergence of the proposed algorithm to the L_q mean is only guaranteed when

embedded with the Log-Euclidean metric. However, our experimental results show that the algorithm converges nicely even in the case of the Affine-Invariant metric.

Rotation Averaging: The problem of rotation averaging has significant applications to structure and motion [Martinec and Pajdla, 2007; Sim and Hartley, 2006; Hartley and Schaffalitzky, 2004; Kahl, 2005; Rother and Carlsson, 2001; Kaucic et al., 2001] and to non-overlapping camera calibration [Dai et al., 2009]. It has been studied quite extensively in the past, both in computer vision and in other fields. The problem has been studied in the context of distributed averaging algorithms Tron et al. [2008, 2012, 2011, 2013]. In a network of cameras the distributed averaging algorithms can be used to estimate the pose of an object seen from different cameras in the network Tron and Vidal [2011]. In the area of information theory the problem of rotations averaging is also known as the synchronization problem [Wang and Singer, 2012]. Significant work in this area includes the work of Govindu [Govindu, 2004, 2001] and Pajdla [Martinec and Pajdla, 2007]. A slightly different Maximum Likelihood (ML) approach to estimate a rotation from its several noisy observations has been proposed in Boumal et al. [2013].

Significant contributions to the single rotation averaging problem have been made by [Moakher, 2002; Manton, 2004; Sarlette and Sepulchre, 2009] and others. Most significant from our point of view is the work reported in [Fletcher et al., 2009; Yang, 2010; Afsari, 2011; Arnaudon et al., 2012] which considers a version of the Weiszfeld algorithm on classes of Riemannian manifolds, proving convergence theorems in a broad context, which relate directly to our algorithm [Hartley et al., 2011]. However, the problem of multiple rotation averaging has been studied outside the vision field in the context of sensor network localization [Cucuringu et al., 2012a; Tron and Vidal, 2009] and molecule problem in structural biology [Cucuringu et al., 2012b].

Averaging of SPD matrices: Averaging of positive definite matrices Lee et al. [2011]; Bini et al. [2010]; Ando et al. [2004]; Bhatia and Holbrook [2006] and positive semi-definite matrices Bonnabel et al. [2013]; Bonnabel and Sepulchre [2009]; Petz and Temesi [2005] has been an active area of research. However, in this case we are interested in averaging of Symmetric Positive-Definite (SPD) matrices Fletcher and Joshi [2007]; Pennec et al. [2006]; bar; Moakher [2006]; Arsigny et al. [2007]; Collard et al. [2012]; Cetingul et al. [2012] In the area of computer vision and medical imaging SPD matrices have many applications. They have been used rigorously in medical imaging to model the anatomical variability of the brain [Fillard et al., 2007]. They can also be used to encode principal diffusion directions in Diffusion Tensor Imaging (DTI) [Fletcher and Joshi, 2007; Pennec et al., 2006]. Furthermore, SPD matrices are widely used in computer vision for motion analysis and texture segmentation [Brox et al., 2003]; and to model the appearance of objects for tracking [Porikli et al., 2006; Tyagi et al., 2008].

Clearly, the application of SPD matrices is not only limited to computer vision and medical imaging. Outside the domain of computer vision, in physics SPD matrices can be used as stress-strain tensors [Moakher, 2006; Salençon, 2001] and they can also be used to solve partial differential equations (PDEs) [Borouchaki et al., 1997].

The space of SPD matrices is not a vector space; instead, the set of SPD matrices lies on a Riemannian manifold that constitutes a convex half-cone in the vector space of matrices.

The structure of the Riemannian manifold of SPD matrices depends on the metric induced. Some of the popular metrics that we discuss in this chapter are: the Euclidean metric, the Log-Euclidean metric [Arsigny et al., 2007]. and the affine invariant metric [Pennec et al., 2006; Fletcher and Joshi, 2004; Lenglet et al., 2004; Moakher, 2005]. Several other metrics and computational frameworks have also been proposed in Collard et al. [2012]; Cetingul et al. [2012] that process shape and orientation independently to find a solution. The problem of finding the L_2 mean of a given set of SPD matrices has been a popular area research since the last decade and even before that. Recently, some methods have been proposed to find the L_2 mean of a set of SPD matrices [Fiori, 2009; Bini and Iannazzo, 2011; Tyagi and Davis, 2008]. In this chapter we show that the proposed algorithm can be used to find the L_q solution of the problem, for $1 \leq q < 2$.

In the next section we state the algorithm along with the properties of the L_q cost function and convergence theorem. We then show that the proposed algorithm can be applied to the L_q averaging problem for rotations and SPD matrices; and present our experimental results for both the problems. Finally, we dedicate a section, at the end of the chapter, for proofs of theorems and lemmas stated in the chapter.

4.2 L_q Weiszfeld Algorithm for Points on a Riemannian Manifold

In this section we propose the L_q Weiszfeld algorithm for points on a Riemannian manifold. Furthermore, we explore and give proofs of properties of the L_q cost function. In the end, we state the convergence theorem. Detailed proofs of the theorems of this section, including the proof of the convergence theorem, are presented in section 4.6. Given a set of points $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}, k > 2$, on a Riemannian manifold $\mathcal{M}$ of non-negative section curvature, the L_q cost function C_q is

$$C_q(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q = \sum_{i=1}^{k} \| \log_{\mathbf{x}}(\mathbf{y}_i) \|^q \,, \tag{4.1}$$

where $d(\cdot, \cdot)$ is the geodesic distance between two points on the manifold $\mathcal{M}$ and $\log_{\mathbf{x}}(\mathbf{y})$ is a Riemannian logarithm map. The gradient of the cost function C_q is

$$\nabla C_q(\mathbf{x}) = -\sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^{q-2} \log_{\mathbf{x}}(\mathbf{y}_i) \,.$$

Just like the above equation of gradient, the rest of the thesis ignores a constant factor of q. In the following section we will state the L_q Weiszfeld algorithm.

4.2.1 Algorithm

The L_q Weiszfeld algorithm differs from the L_1 algorithm [Fletcher et al., 2009] mainly in the choice of weights applied at each step of iteration. In this case, the update equation of

the Weiszfeld algorithm is with weights given by $w_i^t = d(\mathbf{x}^t, \mathbf{y}_i)^{q-2} = \|\log_{\mathbf{x}^t}(\mathbf{y}_i)\|^{q-2}$. Starting from an initial estimate $\mathbf{x}^0$, the algorithm generates a sequence of estimates $\mathbf{x}^t$ found by solving a weighted least-squares problem in the current tangent space of a Riemannian manifold. An updated solution is then projected back on the manifold and the process is repeated until convergence.

A current estimate $\mathbf{x}^t$ of the L_q minimum is updated to a new estimate

$$\begin{aligned} W(\mathbf{x}^t) &= \exp_{\mathbf{x}^t}\left(\frac{\sum_{i=1}^k w_i^t \log_{\mathbf{x}^t}(\mathbf{y}_i)}{\sum_{i=1}^k w_i^t}\right) \quad \text{if } \mathbf{x}^t \notin \mathcal{Y} \\ &= \mathbf{y}_j \quad \text{if } \mathbf{x}^t = \mathbf{y}_j \end{aligned}, \tag{4.2}$$

where

$$w_i^t = d(\mathbf{x}^t, \mathbf{y}_i)^{q-2} .$$

Starting from a point $\mathbf{x}^0 \in \mathcal{M}$, a sequence of points $(\mathbf{x}^t)$ is obtained using W as $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$. In section 4.6.1 we will show that under certain conditions, the sequence of points $(\mathbf{x}^t)$ converges to the L_q minimum or it will stop at some point $\mathbf{x}^t = \mathbf{y}_i$. The conditions required for convergence are that the manifold has non-negative sectional curvature, and that the points $\mathbf{y}_i$ and the initial estimate $\mathbf{x}^0$ lie in a sufficiently restricted region in the manifold.

4.2.2 Convex Sets

The theory of L_q distances in a Riemannian manifold is connected with the concept of a convex set. We discuss this concept before proceeding. It will be used later in section 4.6.1 to prove the convergence of the L_q Weiszfeld algorithm.

A *geodesic* is a generalization of the notion of straight line to Riemannian manifolds. A *geodesic segment* joining points $\mathbf{x}$ and $\mathbf{y}$ in $\mathcal{M}$ is an arc length parametrized continuous curve $\gamma : [a, b] \to \mathcal{M}$ such that $\gamma(a) = \mathbf{x}$ and $\gamma(b) = \mathbf{y}$. A geodesic segment is called *minimizing* if it is the minimum length curve joining its end points. By the Hopf-Rinow theorem [Myers, 1945], any two points $\mathbf{x}$ and $\mathbf{y}$ in a complete Riemannian manifold are joined by a minimizing geodesic segment (though this may not be unique). A convex (or strongly convex) set in a manifold $\mathcal{M}$ is a set that contains a unique geodesic joining two points in the set, and that geodesic is the minimizing geodesic.

A function $f : C \to \mathbb{R}$ defined on a convex set C in $\mathcal{M}$ is *convex* if its restriction to a geodesic in C is a convex function of arc-length. Let $B(\mathbf{x}, r)$ denote an open ball of radius r centered at $\mathbf{x} \in \mathcal{M}$ and $\bar{B}(\mathbf{x}, r)$ be the closure of $B(\mathbf{x}, r)$. A ball satisfying the property of a convex set is a *convex ball*.

The injectivity radius at a point $\mathbf{x}$ of a Riemannian manifold is the supremum of the radii r for which the inverse exponential map at $\mathbf{x}$ is a diffeomorphism on $B(\mathbf{x}, r)$. The injectivity radius r_{inj} of a Riemannian manifold, $\mathcal{M}$, is the infimum of the injectivity radii at all points. Open balls of radius r_{inj} or less in the manifold are therefore diffeomorphic to a Euclidean ball.

Another important quantity is the *convexity radius* of the manifold, which is the largest value r_{conv} such that all open balls $B(\mathbf{o}, r)$ of radius $r < r_{\text{conv}}$ are convex, and furthermore,

radius $r(\mathbf{x}) = d(\mathbf{x}, \mathbf{o})$ is a convex function on this ball. It is shown in [Petersen, 2006], page 177 that the convexity radius of a manifold $\mathcal{M}$ is bounded as follows:

$$r_{\text{conv}} \geq \frac{1}{2} \min\left(r_{\text{inj}}, \frac{\pi}{\sqrt{\Delta}}\right) ,$$

where Δ is the maximal sectional curvature of the manifold $\mathcal{M}$.

In [Fletcher et al., 2009], the convergence of the L_1 Weiszfeld algorithm was shown for manifolds of non-negative sectional curvature, provided that all the points lie inside a ball of radius no greater than $r_{\text{conv}}/2$. However, this is not a tight bound, and in the case of rotation averaging, it is possible to prove convergence within a ball of twice this size. The convexity radius for $\text{SO}(3)$ is equal to $\pi/2$, and hence the result of [Fletcher et al., 2009] ensures convergence as long as the points $\mathbf{y}_i$ lie inside a ball of radius $\pi/4$. It will be shown here that convergence to the L_q minimum is assured, as long as the points are within a ball of radius $\pi/2$, that is within a convex ball in $\text{SO}(3)$. To obtain these improved convergence results in the general case, a different concept of convexity is needed.

4.2.3 Weakly Convex Sets

A *weakly convex* set in a manifold $\mathcal{M}$ is a set W with the following properties.

1. For two points $\mathbf{x}$ and $\mathbf{y}$ in W, there is a unique geodesic in W from $\mathbf{x}$ to $\mathbf{y}$.
2. This geodesic is the shortest length path in W from $\mathbf{x}$ to $\mathbf{y}$.

If this segment is the minimizing geodesic in $\mathcal{M}$ joining $\mathbf{x}$ and $\mathbf{y}$, then the set is strongly convex, as previously defined. [1]

In a weakly convex set W, we may define a distance $d_W(\mathbf{x}, \mathbf{y})$ equal to the length of the unique geodesic that joins $\mathbf{x}$ to $\mathbf{y}$ in W. In addition, the logarithm map on W is defined in terms of the unique geodesic between two points, in W.

Lemma 4.1. *For an open weakly convex set W in a Riemannian manifold $\mathcal{M}$,*

1. *The distance function $d_W(\mathbf{x}, \mathbf{y})$ satisfies the triangle inequality, and hence W is a metric space under this metric.*
2. *W does not contain any pair of conjugate points.*
3. *For any point $\mathbf{x} \in W$, the logarithm map $\log_{\mathbf{x}}$ maps W diffeomorphically into the tangent space.*

The triangle inequality follows directly from the fact that distance is equal to the shortest path length in W.

[1] Terminology for convexity properties varies in the literature, e.g. [Chavel, 2006; Cheeger and Ebin, 1975; Klingenberg, 1982; Petersen, 2006]. Our definition of weakly convex is closest to the definition of weakly convex in [Chavel, 2006]. However, we add an extra condition of uniqueness of the connecting geodesic into our definition of weak convexity to ensure the absence of conjugate points, and injectivity of the exponential map, hence uniqueness of the logarithm map defined on W.

If there exist conjugate points $\mathbf{y}$ and $\mathbf{y}'$ in W, then the geodesic from $\mathbf{y}$ to $\mathbf{y}'$ can be extended to a geodesic from $\mathbf{y}$ to $\mathbf{y}''$. However, geodesics are not minimizing beyond conjugate points [Lee, 1997], so the geodesic from $\mathbf{y}$ to $\mathbf{y}''$ is not minimizing in the manifold W, contrary to the definition of a weakly convex set.

The third statement follows from injectivity of the exponential map into W and a standard result about the exponential map on regions without conjugate points [Lee, 1997].

We use Toponogov's theorem to compare distances in W and in its tangent space. Let κ be the sectional curvature of a Riemannian manifold. The following theorem effectively states that the Toponogov's theorem holds in an open weakly convex set with $\kappa \geq 0$.

Theorem 4.2. *Let W be an open weakly convex set in $\mathcal{M}$, a manifold of non-negative sectional curvature. Let $\mathbf{q}$, $\mathbf{p}_1$ and $\mathbf{p}_2$ be three points in W. Then*

$$d_W(\mathbf{p}_1, \mathbf{p}_2) \leq d(\log_{\mathbf{q}}(\mathbf{p}_1), \log_{\mathbf{q}}(\mathbf{p}_2)) = \| \log_{\mathbf{q}}(\mathbf{p}_1) - \log_{\mathbf{q}}(\mathbf{p}_2) \| .$$

A proof of the above theorem is provided in section 4.6.3. The proof is given, because the usual conditions required by Toponogov's theorem to be true are not satisfied in a weakly convex set, so a proof is required.

Analogous to the definition of convexity radius, we define the *weak convexity radius* r_{wcon} to be the largest value such that all open balls $B(\mathbf{o}, r)$ with $r < r_{\text{wcon}}$ are weakly convex, and $r(\mathbf{x})$ is convex. It is easy to see that $2r_{\text{conv}} \geq r_{\text{wcon}} \geq r_{\text{conv}}$. Indeed, by definition, a ball of radius $\rho < r_{\text{wcon}}/2$ is weakly convex. Therefore, two points $\mathbf{x}$ and $\mathbf{y}$ in $B(\mathbf{o}, \rho)$ are connected by a unique geodesic segment γ in $B(\mathbf{o}, \rho)$. However, this must be a minimizing segment; there cannot be another such segment lying in $B(\mathbf{o}, 2\rho)$, since this is weakly-convex, and any other geodesic from $\mathbf{x}$ to $\mathbf{y}$ that exits the ball $B(\mathbf{o}, 2\rho)$ must be longer than γ. Thus, $B(\mathbf{o}, \rho)$ is convex.

As an example, consider the manifold $\mathrm{SO}(3)$. It is shown in [Hartley et al., 2013] that the convexity radius of $\mathrm{SO}(3)$ is $\pi/2$, whereas the weak-convexity radius is π, which is the maximum distance between points in $\mathrm{SO}(3)$. Note that part of this claim is that $d(\mathbf{x}, \mathbf{o})$ is convex on any ball of radius less than π (and in fact on the open ball $B(\mathbf{o}, \pi)$ itself); see [Hartley et al., 2013]. The closed ball $\bar{B}(\mathbf{o}, \pi)$ is equal to the whole of $\mathrm{SO}(3)$, but any smaller ball is weakly convex.

An essential property of open weakly convex sets is the continuity of the logarithm map. The following theorem says that if the logarithm map is defined in terms of the geodesic that lies inside W, then it is continuous as a function of two variables on this region.

Theorem 4.3. *If W is an open weakly convex set in a complete Riemannian manifold $\mathcal{M}$, and $\mathbf{x}, \mathbf{y}$ are two points in W, define $\log_{\mathbf{x}}(\mathbf{y})$ to be the vector $\mathbf{v}$ in $T_{\mathbf{x}}\mathcal{M} \subset T\mathcal{M}$ such that $\exp_{\mathbf{x}}(\mathbf{v}) = \mathbf{y}$, and $\exp_{\mathbf{x}}(t\mathbf{v}) \in W$ for all $t \in [0, 1]$. Then $\log_{\mathbf{x}}(\mathbf{y})$ as a map from $W \times W$ to $T\mathcal{M}$ is continuous in both variables.*

Our search fails to find this theorem in the literature, so we give a proof of this theorem in section 4.6.2.

4.2.4 Convexity and minima of the L_q Cost

The L_q cost function (4.1) will in general have more than one local minimum on an arbitrary manifold. For example, it was shown in [Hartley et al., 2013] that for n points on the manifold $\mathrm{SO}(3)$, there may be up to $O(n^3)$ local minima. The situation becomes much simpler in the case when (4.1) is a convex function on some convex region.

It is shown in [Afsari, 2011] (proof of Theorem 2.1) that if all points $\mathbf{y}_i$ lie in a ball $\bar{B}(\mathbf{o},\rho)$ of radius $\rho < r_{\text{conv}}$, then the global minimum of C_q lies in $\bar{B}(\mathbf{o},\rho)$. Furthermore, if $\mathbf{x}$ is a point not in $\bar{B}(\mathbf{o},\rho)$ then there exists a point $\mathbf{x}' \in B(\mathbf{o},\rho)$ such that $C_q(\mathbf{x}') < C_q(\mathbf{x})$. In fact, a specific construction is given for $\mathbf{x}'$, as follows. Suppose that $\mathbf{x} = \exp_{\mathbf{o}}(r\mathbf{v})$ where $\|\mathbf{v}\|$ is a unit vector, and $r = d(\mathbf{x},\mathbf{o}) > \rho$, then

$$\mathbf{x}' = \begin{cases} \exp_{\mathbf{o}}((2\rho - r)\mathbf{v}) & \text{if } r < 2\rho \\ \mathbf{o} & \text{if } r \geq 2\rho \end{cases} . \tag{4.3}$$

Thus, for $r < 2\rho$, point $\mathbf{x}'$ is the reflection of $\mathbf{x}$ about the boundary of the ball $B(\mathbf{o},\rho)$, along the radial geodesic. Since $r_{\text{wcon}}/2 \leq r_{\text{conv}}$, the cost function C_q has appealing properties on a ball of radius $\rho < r_{\text{wcon}}/2$, as follows.

Theorem 4.4. *If all points $\mathbf{y}_i$, $i = 1,\ldots,n$ lie in a ball $\bar{B} = \bar{B}(\mathbf{o},\rho)$ with $\rho < r_{\text{wcon}}/2$, then*

1. *$C_q(\mathbf{x})$ is convex on $\bar{B}$ and strictly convex unless $q = 1$ and all points $\mathbf{y}_i$ lie on a single geodesic;*
2. *the global minimum of $C_q(\mathbf{x})$ lies in $\bar{B}$;*
3. *the set $S_0 = \{\mathbf{x} \in \mathcal{M} \mid C_q(\mathbf{x}) \leq C_q(\mathbf{o})\}$ is contained in $\bar{B}(\mathbf{o},2\rho)$, which is a weakly-convex ball.*

Proof. If $\mathbf{x}$ and $\mathbf{y}_i$ are both in $\bar{B}$, then $d(\mathbf{x},\mathbf{y}_i) \leq 2\rho < r_{\text{wcon}}$. Hence, $d(\mathbf{x},\mathbf{y}_i)$ is convex as a function of $\mathbf{x}$ on $\bar{B}$, so $d(\mathbf{x},\mathbf{y}_i)$ has positive-semidefinite Hessian at $\mathbf{x}$. By a simple calculation it follows that $d(\mathbf{x},\mathbf{y}_i)^q$ has positive-definite Hessian for $q > 1$, and is hence convex. Summing over all i shows that $C_q(\mathbf{x})$ is convex.

For $q = 1$ the distance $d(\mathbf{x},\mathbf{y}_i)$ is strictly convex at $\mathbf{x}$ except in the direction pointing towards $\mathbf{y}_i$. Unless the directions to all the points $\mathbf{y}_i$ coincide, the sum of the distance functions will be strictly convex.

The second statement was proved in [Afsari, 2011].

The third statement follows from the triangle inequality, since if $C_q(\mathbf{x}) \leq C_q(\mathbf{o})$, then $d(\mathbf{x},\mathbf{y}_i) \leq d(\mathbf{o},\mathbf{y}_i)$ for some i, and $d(\mathbf{x},\mathbf{o}) \leq d(\mathbf{x},\mathbf{y}_i) + d(\mathbf{o},\mathbf{y}_i) \leq 2\rho$. □

The minima of the cost function C_q may be classified as follows.

Lemma 4.5. *Let D be a subset of $\mathcal{M}$ on which C_q (4.1) is convex. A point $\mathbf{x}^* \in D$ is the minimum of C_q in D if and only if it satisfies one of the following conditions:*

1. *$\nabla C_q(\mathbf{x})$ vanishes at $\mathbf{x}^*$, or*
2. *$q = 1$, $\mathbf{x}^* = \mathbf{y}_j$ and the gradient $\nabla \widehat{C}_q(\mathbf{x}^*)$ (omitting point $\mathbf{y}_j$) has norm no greater than 1.*

For the case $q > 1$ the cost function is differentiable and convex. Its minimum occurs when the gradient vanishes. In the case $q = 1$, the further possibility exists that the minimum occurs, under the stated condition, at some $\mathbf{y}_j$ where the cost function is non-differentiable. This is the condition given by Weiszfeld for the Euclidean case, and it carries over easily to the case of a Riemannian manifold.

4.2.5 Convergence Theorem

We are now able to state a convergence theorem for the L_q Weiszfeld algorithm on a Riemannian manifold.

Theorem 4.6. *Consider a set of points $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, $k > 2$, on a complete Riemannian manifold $\mathcal{M}$ with non-negative sectional curvature, such that not all of the given points lie on a single geodesic. Let all points $\mathbf{y}_i$, lie in a ball $B(\mathbf{o}, \rho)$ of radius $\rho < r_{\text{wcon}}/2$ centered at $\mathbf{o}$ and define $\mathcal{D} = \{\mathbf{x} \in \mathcal{M} \mid C_q(\mathbf{x}) \leq C_q(\mathbf{o})\}$.*

Let $(\mathbf{x}^t)$ be a sequence of points starting from $\mathbf{x}^0$ in $\mathcal{D}$, and defined by $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$ where W is defined in (4.2). Then, the sequence $(\mathbf{x}^t)$ converges to the global minimum of C_q unless $\mathbf{x}^t = \mathbf{y}_i$ for some iteration t and point $\mathbf{y}_i$ (in which case, the sequence remains stuck at $\mathbf{y}_i$).

Proof of this theorem will be deferred until section 4.6.1.

Remark: This algorithm applies for the L_q Weiszfeld algorithm in $\mathbb{R}^N$ where the distance function in (4.1) is simply the Euclidean distance. Note that the Euclidean space is a Riemannian manifold of zero curvature and an injectivity radius of infinity.

4.3 Application I: L_q Optimization on $\mathrm{SO}(3)$

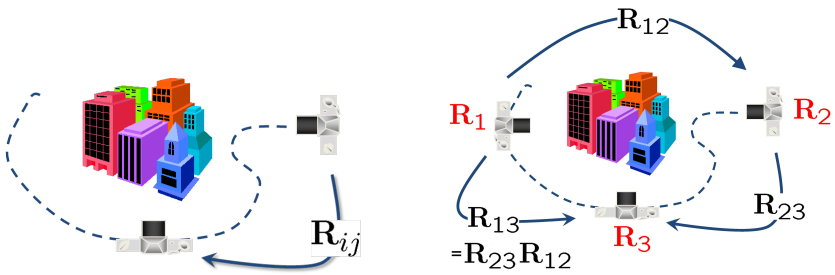
Here we address the problem of L_q rotation averaging, $1 \leq q < 2$, using the proposed L_q optimization method on a Riemannian manifold of non-negative sectional curvature.

Rotation averaging problems can be categorized as either single rotation averaging or multiple rotation averaging; see fig. 4.1. In single rotation averaging, several estimates of a rotation are found and then the L_q Weiszfeld algorithm is applied to find their L_q mean. In multiple rotation averaging, one is given a set of noisy relative rotations $\mathtt{R}_{ij}$ between frames (cameras) indexed by i and j. The task is to find absolute rotations $\mathtt{R}_i$, $\mathtt{R}_j$ that are consistent with the relative rotations: $\mathtt{R}_{ij} = \mathtt{R}_j \mathtt{R}_i^{-1}$. In applications, the relative rotations $\mathtt{R}_{ij}$ may be computed using single rotation averaging as well.

Mathematically, the single rotation averaging problem is as follows. Given rotations $\mathtt{R}_i \in \mathrm{SO}(3)$, the L_q mean, $1 \leq q \leq 2$, is equal to

$$\mathtt{S}^* = \underset{\mathtt{S} \in \mathrm{SO}(3)}{\operatorname{argmin}} \sum_{i=1}^{k} d(\mathtt{R}_i, \mathtt{S})^q \,.$$

The L_1 and L_2 means are the two most useful or common cases. Although L_2 averaging has been considered extensively, the L_q averaging problem in general has been relatively unex-



(a) Single Rotation Averaging (b) Multiple Rotation Averaging

Figure 4.1: *Rotation Averaging: (a) represents two cameras with* $\mathtt{R}_{ij}$ *as a relative rotation between them. We obtain several estimates of the rotation between these two cameras and then perform averaging on them to get a better estimate. (b), we apply the rotation averaging algorithm to estimate absolute rotations,* $\mathtt{R}_i$*, from previously computed relative rotations* $\mathtt{R}_{ij}$*.*

plored. A gradient-descent algorithm for L_1 averaging using line-search in the tangent space was given in [Dai et al., 2009]. However, line-search is costly and cumbersome to implement. In addition, no proof of convergence was given in that paper. Here, we present a simple geodesic L_q averaging algorithm for SO(3), based on the proposed L_q Weiszfeld optimization method on a Riemannian manifold.

Metrics: We consider two metrics commonly used for distance measurement in the rotation group SO(3). These are

1. The geodesic or angle metric $\theta = d_\angle(\mathtt{R},\mathtt{S})$, which is the angle of the rotation $\mathtt{R}\mathtt{S}^{-1}$.
2. The chordal metric

$$d_{\text{chord}}(\mathtt{R},\mathtt{S}) = \|\mathtt{R}-\mathtt{S}\|_F = 2\sqrt{2}\sin(\theta/2)$$

where $\|\cdot\|_F$ represents the Frobenius norm.

These metrics are bi-invariant, in that they satisfy the condition $d(\mathtt{R},\mathtt{S}) = d(\mathtt{TR},\mathtt{TS}) = d(\mathtt{RT},\mathtt{ST})$ for any rotation $\mathtt{T}$. For small values of $\theta = d_\angle(\mathtt{R},\mathtt{S})$ the metrics are the same, to first order, except for a scale factor.

Logarithm and Exponential Maps: The tangent space of SO(3) may be indentified with the set of skew-symmetric matrices, denoted by $\mathfrak{so}(3)$. The Riemannian logarithm and exponential maps may be written in terms of the matrix exponential and logarithm as follows.

Denote by $[\mathbf{v}]_\times$ the skew-symmetric matrix corresponding to $\mathbf{v}$. The Riemannian exponential and logarithm are then defined as

$$\begin{aligned}\exp_{\mathtt{S}}([\mathbf{v}]_\times) &= \mathtt{S}\exp([\mathbf{v}]_\times)\\ \log_{\mathtt{S}}(\mathtt{R}) &= \log(\mathtt{S}^{-1}\mathtt{R})\end{aligned}$$

where log and exp (without subscripts) represent matrix exponential and logarithm. The matrix exponential of a skew-symmetric matrix may be computed using the Rodrigues formula

[Hartley and Zisserman, 2004]. The distance $d(\mathtt{S},\mathtt{R})$ is computed by

$$d(\mathtt{S},\mathtt{R}) = (1/\sqrt{2})\|\log_{\mathtt{S}}(\mathtt{R})\|_F$$

where $\|\cdot\|_F$ represents the Frobenius norm, and the scale factor $1/\sqrt{2}$ is present so that $d(\mathtt{S},\mathtt{R})$ is equal to the angular distance from $\mathtt{S}$ to $\mathtt{R}$. If $\log_{\mathtt{S}}(\mathtt{R}) = [v]_\times$, then $d(\mathtt{S},\mathtt{R}) = \|\mathbf{v}\|$, in terms of the Euclidean norm in $\mathbb{R}^3$.

For a more complete discussion of metrics on rotation space and rotation averaging, see [Hartley et al., 2013].

4.3.1 L_2 Averaging

The rotation averaging problem on $\mathrm{SO}(3)$ under the L_2 norm may be solved in closed form for the chordal metric. However, there is no closed-form algorithm for L_2 rotation averaging under the geodesic metric, but convergent algorithms have been proposed [Afsari et al., 2013; Dai et al., 2009]. We give a brief description of the L_2 averaging algorithms using the geodesic metric and the chordal metric.

L_2 Geodesic Mean: The L_2 geodesic mean of a set of rotations $\mathtt{R}_i$ can only be computed iteratively. The mean is unique provided the given rotations $\mathtt{R}_1,\ldots,\mathtt{R}_k$ do not lie too far apart. A constant step size gradient descent algorithm to find the L_2 mean has been proposed in Afsari et al. [2013]. When $\delta \leq \pi/2$ and all the rotations lie in a ball of radius δ, then a constant step size algorithm with a step size of $\lambda \in (0,1]$ is shown to converge to the L_2 minimum. For a given set of rotations $\{\mathtt{R}_i\}$ and a starting point $\mathtt{S}^0$ in a ball of radius δ, we proceed as,

$$\mathtt{S}^{t+1} = \exp_{\mathtt{S}^t}\left(\lambda \textstyle\sum_{i=1}^k \log_{\mathtt{S}^t}(\mathtt{R}_i)\right) .$$

The above procedure is guaranteed to converge to the L_2 mean. For a Newton-type algorithm to compute this mean see Krakowski et al. [2007].

L_2 Chordal Mean: Computation of the L_2 chordal mean does not require knowledge of a bounding ball for the rotations $\mathtt{R}_i$, and so it is useful as a way to find an initial estimate for an iterative Weiszfeld algorithm.

Let $\mathtt{R}_{\mathrm{sum}} = \sum_{i=1}^k \mathtt{R}_i$, the sum of 3×3 rotation matrices. The L_2 chordal mean $\mathtt{S}$ is obtained using the Singular Value Decomposition. Let $\mathtt{R}_{\mathrm{sum}} = \mathtt{U}\,\mathtt{D}\,\mathtt{V}^\top$ where the diagonal elements of $\mathtt{D}$ are arranged in descending order. If $\det(\mathtt{U}\mathtt{V}^\top) \geq 0$, then set $\mathtt{S} = \mathtt{U}\mathtt{V}^\top$. Otherwise set $\mathtt{S} = \mathtt{U}\,\mathrm{diag}(1,1,-1)\mathtt{V}^\top$.

For justification of this algorithm, see [Hartley et al., 2013].

4.3.2 L_q Geodesic Mean in $\mathrm{SO}(3)$

We now consider the problem of computing the L_q geodesic mean in the group of rotations. The L_q Weiszfeld algorithm will be used to compute the minimum of the cost function

$$C_q(\mathtt{S}) = \sum_{i=1}^k d(\mathtt{R}_i,\mathtt{S})^q . \tag{4.4}$$

To minimize (4.4) we transition back and forth between the rotation manifold, and its tangent space centred at the current estimate via the exponential and logarithm maps as described earlier.

In terms of the matrix exponential and logarithms the update step (4.2)of L_q optimization, may then be written as

$$\begin{aligned} \mathtt{S}^{t+1} &= \mathtt{S}^t \exp\left(\frac{\sum_{i=1}^k w_i^t \log((\mathtt{S}^t)^{-1}\mathtt{R}_i)}{\sum_{i=1}^k w_i^t}\right) \quad \text{if } \mathtt{S}^t \notin \{\mathtt{R}_i\} \\ &= \mathtt{R}_j \quad \text{if } \mathtt{S}^t = \mathtt{R}_j \end{aligned} , \tag{4.5}$$

where $w_i^t = d(\mathtt{S}^t, \mathtt{R}_i)^{q-2}$.

For computational efficiency, it is simpler to work with the quaternion representations $\mathbf{r}_i$ of the rotations $\mathtt{R}_i$, since mapping between quaternions and angle-axis representation is simpler than computing the exponential and logarithm maps. In addition quaternion multiplication is faster than matrix multiplication. Let Q be the unit quaternions, $\hat{\mathbf{v}}$ be a unit vector and θ a scalar representing an angle. The mapping $q : \mathbb{R}^3 \to Q$ given by

$$q : \theta\hat{\mathbf{v}} \mapsto (\cos(\theta/2), \sin(\theta/2)\hat{\mathbf{v}}) ,$$

maps between the angle-axis and quaternion representation of the rotation through angle θ about the axis $\mathbf{v}$. Then the update step above may be expressed as

$$\begin{aligned} \theta_i\hat{\mathbf{v}}_i &= q^{-1}(\bar{\mathbf{s}}^t \cdot \mathbf{r}_i) , \\ \delta &= \frac{\sum_{i=1}^k \theta_i\hat{\mathbf{v}}_i/\theta_i^{2-q}}{\sum_{i=1}^k 1/\theta_i^{2-q}} , \\ \mathbf{s}^{t+1} &= \mathbf{s}^t \cdot q(\delta) , \end{aligned} \tag{4.6}$$

where $\bar{\mathbf{s}}^t$ represents the conjugate (inverse) of the quaternion $\mathbf{s}^t$. A further alternative is to use the Campbell-Baker-Hausdorff formula [Govindu, 2004] to work entirely in angle-axis space, but this is essentially equivalent to the use of quaternions.

According to Theorem 4.6 this sequence of iterates will converge to the L_q mean of the rotations $\mathtt{R}_i$, provided all the rotations and the initial extimate $\mathtt{S}^0$ lie within a ball of radius $\pi/2$.

4.3.3 L_q Multiple Rotation Averaging

We now consider the problem of rotation averaging of a set of relative rotations. More specifically, let $\mathtt{R}_i;\ i = 1, \ldots, k$ be a set of rotations denoting the orientation of different coordinate frames in $\mathbb{R}^3$. The rotations are assumed unknown, but a set of relative rotations $\mathtt{R}_{ij}$ are given, for pairs $(i, j) \in \mathcal{N}$, where $\mathcal{N}$ is a subset of all index pairs. If $(i, j) \in \mathcal{N}$, then also $(j, i) \in \mathcal{N}$, and $\mathtt{R}_{ji} = \mathtt{R}_{ij}^{-1}$. These relative rotation matrices $\mathtt{R}_{ij}$ are provided by some measurement process and are assumed to be corrupted by some degree of noise. The required task is to find the absolute rotations $\mathtt{R}_i, \mathtt{R}_j$ such that $\mathtt{R}_{ij} = \mathtt{R}_j\mathtt{R}_i^{-1}$ for all pairs $(i, j) \in \mathcal{N}$. Of course, since this

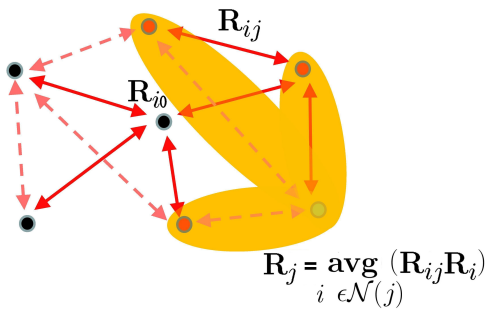


Figure 4.2: *Multiple Rotation Averaging: Nodes of the above graph represent absolute rotations* $\mathtt{R}_i$ *and edges of the graph represent relative rotation* $\mathtt{R}_{ij}$*. After fixing a root node* $\mathtt{R}_{i0}$ *we construct a spanning tree of the graph (represented by solid arrows). For each node* $\mathtt{R}_j$*, we apply a single iteration of the* L_q *rotation averaging algorithm on its neighboring nodes* $\mathcal{N}(j)$ *to get an averaged estimate of* $\mathtt{R}_j$*. This process is repeated for every node of the graph.*

condition can not be fulfilled exactly, given noisy measurements $\mathtt{R}_{ij}$, so the task is to minimize the cost

$$C_q(\mathtt{R}_1, \ldots, \mathtt{R}_M) = \sum_{(i,j)\in\mathcal{N}} d(\mathtt{R}_{ij}\mathtt{R}_i, \mathtt{R}_j)^q \,,$$

where $1 \leq q < 2$. We consider the geodesic distance function $d(\cdot,\cdot) = d_\angle(\cdot,\cdot)$. We may eliminate the obvious gauge freedom (ambiguity of solution) by setting any one of the rotations $\mathtt{R}_i$ to the identity. Generally, minimizing this cost is a difficult problem because of the existence of local minima, but in practice it may be solved in many circumstances with more-or-less acceptable results. Here, we will consider the L_q averaging problem, and demonstrate an algorithm that gives excellent results on large data sets.

Our approach is by successive L_q averaging to estimate each $\mathtt{R}_i$ in turn, given its neighbours. At any given point during the computation, a rotation $\mathtt{R}_i$ will have an estimated value, and so will its neighbors $\mathtt{R}_j$, for $(i,j) \in \mathcal{N}$. Therefore, we may compute estimates $\mathtt{R}_i^{(j)} = \mathtt{R}_{ji}\mathtt{R}_j$, where the superscript (j) indicates that this is the estimate of $\mathtt{R}_i$ derived from its neighbour $\mathtt{R}_j$. We then use our L_q averaging method on $\mathrm{SO}(3)$ to compute a new estimate for $\mathtt{R}_i$ by averaging the estimates $\mathtt{R}_i^{(j)}$, fig. 4.2. In one pass of the algorithm, each $\mathtt{R}_i$ is re-estimated in turn, in some order. Multiple passes of the algorithm are required for convergence.

Since the L_q averaging algorithm on $\mathrm{SO}(3)$ is itself an iterative algorithm, we have the choice of running the L_q averaging algorithm to convergence, each time we re-estimate $\mathtt{R}_i$, or else running it for a limited number of iterations leaving the convergence incomplete, and passing on to the next rotation. To avoid nested iteration, we choose to run a single iteration of the L_q averaging algorithm at each step. The complete algorithm is as follows.

Algorithm 4.1. *Given a set of relative rotations* $\mathtt{R}_{ij}$ *we proceed as:*

1. **Initialization:** *Set some node* $\mathtt{R}_{i_0}$*, with the maximum number of neighbours, to the identity rotation, and construct a spanning tree in the neighbourhood graph rooted at* $\mathtt{R}_{i_0}$*. Estimate the rotations* $\mathtt{R}_j$ *at each other node in the tree by propagating away from*

the root using the relation $\mathtt{R}_j = \mathtt{R}_{ij}\mathtt{R}_i$.

2. **Sweep:** *For each i in turn, re-estimate the rotation $\mathtt{R}_i$ using one iteration of the L_q averaging algorithm. (As each new $\mathtt{R}_i$ is computed, it is used in the computations of the other $\mathtt{R}_i$ during the same sweep.)*

3. **Iterate:** *Repeat this last step a fixed number of times, or until convergence.*

As shown in Tron and Vidal [2009], there can be several choices for the initialization of rotations, such as fixing a node as a base node or using a slightly different distance functions on $\mathrm{SO}(3)$. In this case, we only consider the first method where a base node is set to identity and rest of the rotations are computed relative to the base node. The whole computation is most conveniently carried out using quaternions.

Unlike the single rotation averaging problem considered in section 4.3.2 we can not guarantee convergence of this algorithm to a global minimum, but results will demonstrate good performance.

4.4 Application II: L_q Averaging on SPD manifold

We apply the L_q Weiszfeld algorithm for points on a Riemannian manifold to find the L_q mean of a set of SPD matrices. The space of $n \times n$ SPD matrices is represented by Sym_n^+. Given a set of SPD matrices $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_k\}$, where $\mathtt{Y}_i \in Sym_n^+$, we seek a point $\mathtt{X} \in Sym_n^+$ for which the L_q cost function has a minimum value. The L_q cost function is defined as,

$$C_q(\mathtt{X}) = \sum_{i=1}^{k} d(\mathtt{X}, \mathtt{Y}_i)^q \,, \tag{4.7}$$

where $1 \leq q < 2$ and $d(\cdot, \cdot)$ is a distance function. Since the space of SPD matrices can be endowed with a Riemannian structure, the distance function is a geodesic distance.

4.4.1 Metrics

The space of SPD matrices is not a vector space; instead, it constitutes a manifold. This manifold can be endowed with different Riemannian metrics. Here we will summarize some of the popular metrics such as the Euclidean metric, the affine invariant metric [Pennec et al., 2006; Fletcher and Joshi, 2004; Lenglet et al., 2004; Moakher, 2005] and the Log-Euclidean metric [Arsigny et al., 2007]. We focus more on the Log-Euclidean metric because it is easy to compute than the affine invariant metric and the Riemannian manifold has a zero sectional curvature when endowed with the Log-Euclidean metric.

We are also interested in the matrix logarithm and matrix exponential of an SPD matrix. Note that these maps are different from the logarithm and exponential maps defined on a Riemannian manifold, where these maps are used to transfer points between manifold and its tangent space. We represent the matrix logarithm by log and matrix exponential by exp, without any subscript. These maps will be used to write the induced metrics explicitly in the form of log and exp. Since an SPD matrix can be represented by a positive diagonal matrix in some

orthonormal basis, the respective matrix logarithm and matrix exponential forms are simplified and thus are computed as

$$\begin{aligned}\log(\mathtt{X}) &= \log(\mathtt{R}\,\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)\,\mathtt{R}^T)\\ &= \mathtt{R}\,\mathrm{diag}(\log(\lambda_1),\log(\lambda_2),\ldots,\log(\lambda_n))\,\mathtt{R}^T\,,\end{aligned} \tag{4.8}$$

and

$$\begin{aligned}\exp(\mathtt{X}) &= \exp(\mathtt{R}\,\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)\,\mathtt{R}^T)\\ &= \mathtt{R}\,\mathrm{diag}(\exp(\lambda_1),\exp(\lambda_2),\ldots,\exp(\lambda_n))\,\mathtt{R}^T\,.\end{aligned} \tag{4.9}$$

We now discuss some of the commonly used metrics for computations on SPD manifold.

1. **Euclidean Metric:** This metric is defined by endowing the space of SPD matrices with a Euclidean structure. This leads us to a simple and straightforward metric. The distance function is simply,

   $$d(\mathtt{X},\mathtt{Y}) = \|\mathtt{X}-\mathtt{Y}\|\,,$$

   where $\|\cdot\|$ is the Frobenius norm. Averaging under the Euclidean metric is not adequate in many situations because matrices with non-positive eigenvalues are at a finite distance from other SPD matrices and the Euclidean averaging often suffers from the swelling effect, which means that the determinant of the Euclidean mean can be strictly larger than the original determinants, for details see [Arsigny et al., 2007]. The swelling issue can be resolved either by using other Riemannian metrics or by using the framework proposed in Cetingul et al. [2012]. Because of simplicity, we only explore the first strategy where averaging is performed by using different metrics.

2. **Affine Invariant Metric:** Unlike the Euclidean metric where the space of SPD matrices is endowed with the Euclidean structure, in the case of affine invariant metric a curvature is induced on the space. An advantage of inducing this structure is that the drawbacks of the Euclidean metric are removed; null eigenvalues are at an infinite distance and the swelling effect has disappeared. But these advantages come at a cost of high computational time that eventually leads to slow and hard to implement algorithms.

   In this case the distance function is defined as

   $$d(\mathtt{Y}_1,\mathtt{Y}_2) = \|\log(\mathtt{Y}_1^{-1/2}\,\mathtt{Y}_2\,\mathtt{Y}_1^{-1/2})\|\,, \tag{4.10}$$

   where $\|\cdot\|$ is the Frobenius norm. In this case the logarithm and exponential maps are defined as

   $$\log_{\mathtt{X}}(\mathtt{Y}) = \mathtt{X}^{1/2}\,\log(\,\mathtt{X}^{-1/2}\,\mathtt{Y}\,\mathtt{X}^{-1/2}\,)\,\mathtt{X}^{1/2}\,,$$

   and

   $$\exp_{\mathtt{X}}(\mathtt{Y}) = \mathtt{X}^{1/2}\,\exp(\,\mathtt{X}^{-1/2}\,\mathtt{Y}\,\mathtt{X}^{-1/2}\,)\,\mathtt{X}^{1/2}\,,$$

respectively. For details see [Pennec et al., 2006].

3. **Log-Euclidean Metric:** We focus more on the Log-Euclidean metric [Arsigny et al., 2007] because it is very simple to compute and overcomes the drawbacks of both the Euclidean and affine invariant metrics. The metric is called Log-Euclidean because it is a Euclidean metric in matrix logarithm space. It not only overcomes the drawbacks of the Euclidean metric but is also very efficient to compute when compared with the affine invariant metric.

   In this case the logarithm and exponential maps are defined as

$$\log_{\mathtt{X}}(\mathtt{Y}) = \log(\mathtt{Y}) - \log(\mathtt{X}) \,,$$

   and

$$\exp_{\mathtt{X}}(\mathtt{Y}) = \mathtt{X} \exp(\mathtt{Y}) \,,$$

   respectively. For details see [Pennec et al., 2006]. The Log-Euclidean distance function is then,

$$d(\mathtt{X}, \mathtt{Y}) = \| \log(\mathtt{Y}) - \log(\mathtt{X}) \| \,,$$

   where $\| \cdot \|$ is the norm associated with the metric, that is the Frobenius norm.

4.4.2 L_2 Averaging for SPD Matrices

In this section we discuss several methods for computing the L_2 mean of a set of SPD matrices. Let $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_k\}$ be a given set of SPD matrices, where $\mathtt{Y}_i \in Sym_n^+$. The L_2 mean of these matrices is a point for which the following function is minimum,

$$C_2(\mathtt{X}) = \sum_{i=1}^{k} d(\mathtt{X}, \mathtt{Y}_i)^2 \,, \tag{4.11}$$

where $d(\cdot, \cdot)$ is a distance function that depends on the induced metric.

4.4.2.1 Euclidean Metric

Given a set of SPD matrices $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_k\}$, the Euclidean mean is simply the arithmetic mean and is computed explicitly by

$$\widetilde{\mathtt{X}} = \frac{1}{k} \sum_{i=1}^{k} \mathtt{Y}_i \,.$$

The L_2 mean under the Euclidean metric is very simple to compute but it suffers from several drawbacks, such as swelling effect, null eigenvalues at finite distance, etc, that limit the usability of the Euclidean mean for many applications. For details see [Arsigny et al., 2007; Pennec et al., 2006].

4.4.2.2 Affine Invariant Metric

The L_2 mean of a set of SPD matrices $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_k\}$ under the affine invariant metric is computed iteratively using

$$\mathtt{X}^{t+1} = (\mathtt{X}^t)^{-\frac{1}{2}} \exp\left(\frac{1}{k}\sum_{i=1}^{k} \log((\mathtt{X}^t)^{-\frac{1}{2}}\mathtt{Y}_i(\mathtt{X}^t)^{-\frac{1}{2}})\right)(\mathtt{X}^t)^{-\frac{1}{2}},$$

where $\mathtt{X}^t$ is an estimate of the L_2 mean at iteration t; and exp and log are matrix exponential and logarithm as defined in (4.9) and (4.8), respectively. Although this metric does not suffer from the drawbacks of the Euclidean metric, it is computationally expensive and sometimes hard to implement. But still it makes more sense than the Euclidean metric because it respects the Riemannian structure on the manifold of SPD matrices, that is clearly a natural way of modeling the problem.

4.4.2.3 Log-Euclidean Metric

The Log-Euclidean metric [Arsigny et al., 2007] endows the space of SPD matrices with a Riemannian structure but unlike the affine invariant metric, the Log-Euclidean metric has a simple form and is easy to compute. Given a set of SPD matrices $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_k\}$ the L_2 mean using the Log-Euclidean metric is computed in closed-form as,

$$\widetilde{\mathtt{X}} = \exp\left(\frac{1}{k}\sum_{i=1}^{k}\log(\mathtt{Y}_i)\right),$$

where exp and log are matrix exponential and logarithm, respectively. It is shown in [Arsigny et al., 2007] that the Log-Euclidean mean is similarity-invariant.

4.4.3 L_q Averaging for SPD Matrices

In this section we show that the L_q Weiszfeld algorithm [Aftab et al.] can be applied to the space of SPD matrices to find the L_q mean of a set of SPD matrices, for $1 \leq q < 2$. We endow the space of SPD matrices with the Log-Euclidean metric because of its simplicity, low computational cost, and its zero Riemannian curvature [Arsigny et al., 2007]. The space of SPD matrices has a zero sectional curvature when endowed with the Log-Euclidean metric and has a non-positive sectional curvature under the Affine-Invariant metric. Since, the proof of Theorem 4.6 is true for Riemannian manifolds of non-negative sectional curvature, the L_q Weiszfeld algorithm is provable convergence under the Log-Euclidean metric. However, no such proof exists for the Affine-Invariant metric. But our experimental results show that the algorithm converges nicely for both the metrics.

Given a set of SPD matrices $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_k\}$, such that not all of the given SPD matrices lie on a single geodesic the L_q cost function is defined as,

$$C_q(\mathtt{X}) = \sum_{i=1}^{k} d(\mathtt{X}, \mathtt{Y}_i)^q = \sum_{i=1}^{k} \| \log_{\mathtt{X}}(\mathtt{Y}_i)\|^q,$$

where $\|\cdot\|$ is the Frobenius norm.

According to Theorem 4.6, a current solution of the L_q mean is updated in the descent direction of C_q as,

$$\begin{aligned} \mathtt{X}^{t+1} = W(\mathtt{X}^t) = & \exp_{\mathtt{X}^t}\left(\frac{\sum_{i=1}^k w_i^t \log_{\mathtt{X}^t}(\mathtt{Y}_i)}{\sum_{i=1}^k w_i^t}\right) \quad \text{if } \mathtt{X}^t \notin \{\mathtt{Y}_i\} \\ & = \mathtt{Y}_j \quad \text{if } \mathtt{X}^t = \mathtt{Y}_j \end{aligned}, \tag{4.12}$$

where $w_i^t = \|\log_{\mathtt{X}^t}(\mathtt{Y}_i)\|^{q-2}$.

It is shown in [Arsigny et al., 2007, Corollary 3.10] that under the Log-Euclidean metric the space of SPD matrices is a flat Riemannian manifold, that is its sectional curvature is zero everywhere. Thus, all the discussion of section 4.2 regarding injectivity radius, convexity radius, etc, becomes irrelevant. As a result, from Theorem 4.6, the sequence of points obtained using the update function W (4.12) converges to the minimum of C_q, except when $\mathtt{X}^t = \mathtt{Y}_i$ for some value of t..

In the case of the Log-Euclidean metric, the update function W in (4.12) can explicitly be written as,

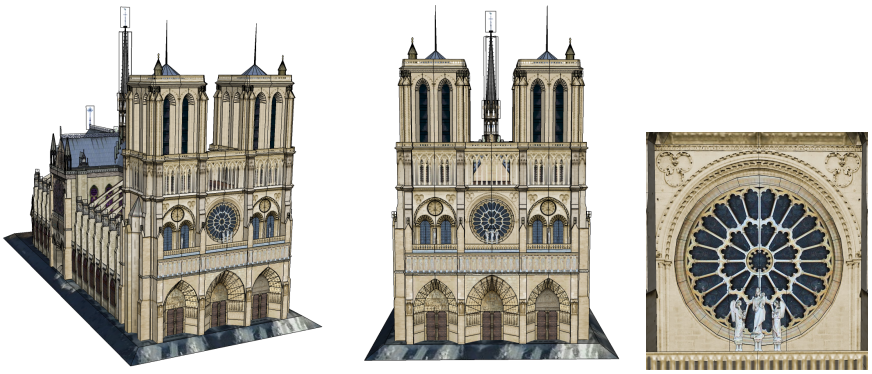
$$\begin{aligned} \mathtt{X}^{t+1} = W(\mathtt{X}^t) = & \exp\left(\frac{\sum_{i=1}^k w_i^t \log(\mathtt{Y}_i)}{\sum_{i=1}^k w_i^t}\right) \quad \text{if } \mathtt{X}^t \notin \{\mathtt{Y}_i\} \\ & = \mathtt{Y}_j \quad \text{if } \mathtt{X}^t = \mathtt{Y}_j \end{aligned}, \tag{4.13}$$

where $w_i^t = \|\log(\mathtt{Y}_i) - \log(\mathtt{X}^t)\|^{q-2}$. Note that the log and exp maps in the above equation are matrix logarithm and exponential maps. Furthermore, it is not necessary to compute the matrix logarithm and exponential of the points at every iteration of the algorithm. Rather, an equivalent strategy is to compute the matrix logarithm only once, perform the iterations of the algorithm in the logarithm space until convergence, and then map the final solution back to the exponential space by using the matrix exponential map. This further simplifies the L_q Weiszfeld algorithm under the Log-Euclidean metric and makes the technique computationally efficient than the Affine-Invariant metric.

4.4.3.1 Discussion Related to the Affine-Invariant Metric

As stated in Theorem 4.6 that the sequence of points $(\mathbf{x}^t)$ converges to the L_q mean if all the points $\{\mathbf{y}_i\}$ lie on a Riemannian manifold of non-negative sectional curvature. The Riemannian structure endowed on the space of SPD matrices has a non-positive sectional curvature under the Affine-Invariant metric. For details see [Pennec et al., 2006]. Therefore, the convergence of the sequence of points obtained using (4.12) is not guaranteed under the Affine-Invariant metric.

Regardless of the fact that the sequence of points is not convergent, we apply the L_q Weiszfeld algorithm to the problem of finding the L_q mean of a set of SPD matrices, under the Affine-Invariant metric. Our experimental results show that the sequence of points obtained using (4.12) converges very nicely to the L_q mean. Thus, in practice the L_q Weiszfeld algorithm can be applied to the problem but unfortunately its convergence to the L_q minimum



(a) Frame 1 (b) Frame 2 (c) Frame 3

Figure 4.3: *Data Set: (a), (b) and (c) shows multiple images of the Notre Dame cathedral.*

is not guaranteed.

4.5 Experimental Results

4.5.1 Rotation Averaging

We demonstrate the utility and accuracy of the L_q rotation averaging methods, for $q = 1$, by applying them to a large-scale reconstruction problem, based on the Notre Dame data set [Snavely et al., 2006] see fig. 4.3. This set has been reconstructed and bundle-adjusted, resulting in estimates of all the camera matrices, which we take to represent ground truth. The set consists of 595 images of $277,887$ points. There exist $42,621$ pairs of images with more than 30 corresponding point pairs, and these were the pairs of images that we used in our tests.

4.5.1.1 Single Rotation Averaging

To test the algorithm for estimating a single rotation from several estimates, we carried out the following procedure.

1. Subsets of five point pairs were chosen and a fast five-point algorithm [Nistér, 2004] was used to estimate the essential matrix from the pair of images, and from this the relative rotation and translation were computed. Only those solutions were retained that satisfied the cheirality constraint that all 5 points lie in front of both estimated cameras. This can be done extremely quickly – in our implementation about $35\mu s$ per 5-point sample.

2. The solutions were tested against 3 further points and only solutions which fitted well against these points were retained.

3. From several subsets of 5 points we obtained several estimates of the relative rotation (a subset can lead to more than one rotation estimate).

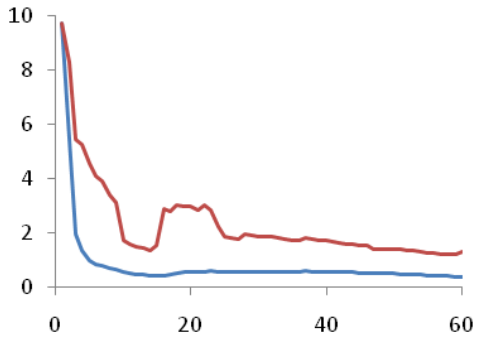


Figure 4.4: *The graph shows the result of L_2 (top curve) and L_1 (bottom curve) rotation averaging, used in computing the relative orientation of two cameras from repeatedly applying the 5-point algorithm to estimate relative rotation. The plots show the error with respect to ground truth as a function of the number of samples taken. As can be seen, the L_1 algorithm converges in this case to close to ground truth with about 10 samples.*

4. The rotation estimates were then averaged to find their L_1 mean. A closed form L_2 rotation averaging was used to find an initial estimate, followed by application of some steps of the Weiszfeld algorithm.

This method was compared with straight L_2 rotation averaging; the L_1 averaging technique gave significantly better results. In addition, the results were compared with those obtained by using non-minimal methods based on the 8-point algorithm, followed by algebraic error or Sampson error minimization, and calibrated bundle adjustment [Hartley and Zisserman, 2004].

It is possible that this averaging technique can be used as an alternative to RANSAC in the case of noisy point correspondences, but we emphasize that this was not the purpose of this experiment. Rather, the point was to demonstrate the advantage of L_1 rotation averaging, and investigate it as a means for computing two-view relative pose.

Results: We carried out experiments in which the relative rotation of two cameras was computed using the 5-point algorithm, followed by averaging the rotation results from many rotation samples computed in this way. In all cases, the L_1 averaging algorithm worked significantly better. In fig. 4.4 is shown a typical result of this estimation procedure, comparing L_1 with L_2 averaging algorithms, for increasing numbers of rotations.

4.5.1.2 Multiple Rotation Averaging

The results of pairwise rotation estimates obtained in the previous section were then used as input to the multiple rotation averaging algorithm described in section 4.3.3.

In carrying out this test, the two-view relative rotation estimates were obtained using several techniques. Generally speaking, more elaborate methods of computing relative rotation led to better results, but the fast methods were shown to give surprisingly good results very fast. The following methods were used for finding pairwise relative rotations $\mathtt{R}_{ij}$.

1. **E-5pt**(m, n)**:** Rotations were obtained from essential matrices computed from m minimal 5-point sets, then averaged using the L_2-chordal algorithm, followed by n steps of

Method	per-pair time(msec)	total time(sec)	L1	L2
E-bundled	281	11932	0.82	0.93
E-algebraic	4.07	173	1.21	1.84
E-Sampson	19	839	1.05	1.85
E-5pt(30,20)	7	296	0.98	1.32
E-5pt(20,10)	4	168	0.93	1.46
L1-averaging	–	36		
L2-averaging	–	10		

Table 4.1: Timing (on a 2.6 GHz laptop) for the computation of the $42,621$ essential matrices using various methods, and also the time taken for L_1 and L_2 averaging over all nodes. This last operation is carried out once only. Columns 2 and 3 show the time per iteration, and total time. The last two columns give the median (over 595 views) rotation error in degrees for L_1 and L_2 averaging. As may be seen, the full bundle adjustment takes a lot more time, though it does lead to slightly better results. We do not count time taken for finding the pairs of overlapping images with sufficiently many matches. Observe that E-5pt(30,20) did better than E-5pt(20,10) for L_1 averaging, but this was by chance.

L_1 averaging using the Weiszfeld algorithm.

2. **E-algebraic:** The algebraic cost $\sum_i (\mathbf{x}_i'^{\top} \mathtt{E} \mathbf{x}_i)^2$ was minimized iteratively over the space of all valid essential matrices. This is an adaptation of the method of [Hartley, 1998] to essential matrices, and is very efficient and fast.

3. **E-Sampson:** The Sampson error

$$\sum_i \frac{(\mathbf{x}_i'^{\top} \mathtt{E} \mathbf{x}_i)^2}{(\mathtt{E}\mathbf{x})_1^2 + (\mathtt{E}\mathbf{x})_2^2 + (\mathtt{E}^{\top}\mathbf{x}')_1^2 + (\mathtt{E}^{\top}\mathbf{x}')_2^2}$$

was minimized over the space of essential matrices.

4. **E-bundled:** Full 2-view bundle adjustment was carried out, initialized by the results of E-algebraic. This method was expected to give the best results (and it did), but requires substantially more computational effort (cf. Tab 4.1).

Given the diversity of image-pair configurations, possible small overlap and general instability, no one method gave perfectly accurate relative rotation estimates for all $42,621$ image pairs. However, in all cases the resulting rotation errors for the 595 cameras were quite accurate. For the E-bundled method, the median camera orientation error was 0.82 degrees.

Detailed results: The results of rotation averaging on the Notre Dame data set [Snavely et al., 2008] are given in fig. 4.5 and fig. 4.6. Pairs of images from this set were chosen if they shared more that 30 points in common (42,621 such pairs). From these pairs, the essential matrix was computed using various different methods as described before. It is evident from fig. 4.5 and fig. 4.6 that the results of the L_1 algorithm by using E-bundled method are superior in term of accuracy than the rest of the methods. However, considering the higher computation

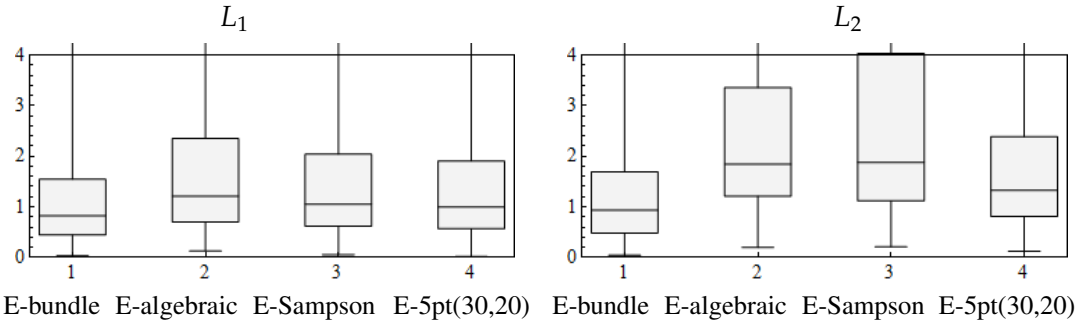


Figure 4.5: *Whisker plots of the absolute orientation accuracy of the* 595 *images of the Notre Dame data set. The top and bottom of the boxes represent the 25% and 75% marks. The left graph shows the result of* L_1 *averaging and the right graph the* L_2 *averaging results. In each graph are shown the results arising from different methods of computing the essential matrices, and hence the rotations.*

cost of the E-bundle than the rest of the techniques this method is not recommended when efficiency is of primary concern. On the other hand, the results of the L_1 averaging method using E-5pt$(30, 20)$ method are slightly higher than the E-bundled method but the E-5pt method is much more efficient than the rest of the techniques. Thus, L_1 averaging together with the E-5pt method is recommended because of its efficiency and accuracy.

4.5.2 SPD Averaging

Now, we discuss experimental results of the L_q averaging methods for SPD matrices. In our experiments we consider both the Log-Euclidean and Affine-Invariant metrics. We perform averaging over a synthetic data of a set of 3×3 SPD matrices. Our objective is to estimate a SPD matrix from a set of its noisy samples. To achieve this purpose we perform L_q averaging and L_2 averaging under both the Log-Euclidean metric and the Affine-Invariant metrics. Although, there is no proof of convergence of the L_q averaging algorithm under the Affine-Invariant metric but our experiments show promising results, comparable to the results under the Log-Euclidean metric. Furthermore, the L_q algorithm under the Affine-Invariant metric converges nicely without showing any abnormalities. We primarily focus on the Log-Euclidean metric, for which the proposed algorithm is provably convergent. In all of our experiments we assume that the ground truth is known and all the errors are computed using the ground truth value.

In our experiments we show following results:

1. *Convergence behavior:* The primary purpose of this experiment is to show the convergence behavior of the averaging algorithms. Results have shown that the L_q averaging algorithm, for $q = 1$, converges nicely under both the the Log-Euclidean metric and the Affine-Invariant metric.

2. *Robustness against different number of outliers:* In this experiment we show the robustness of the L_q averaging algorithm under the Log-Euclidean metric for different values of q. In order to achieve this purpose we add different number of outliers in the dataset and then compare the results of the L_q averaging algorithms, for q ranging from 1 to 2

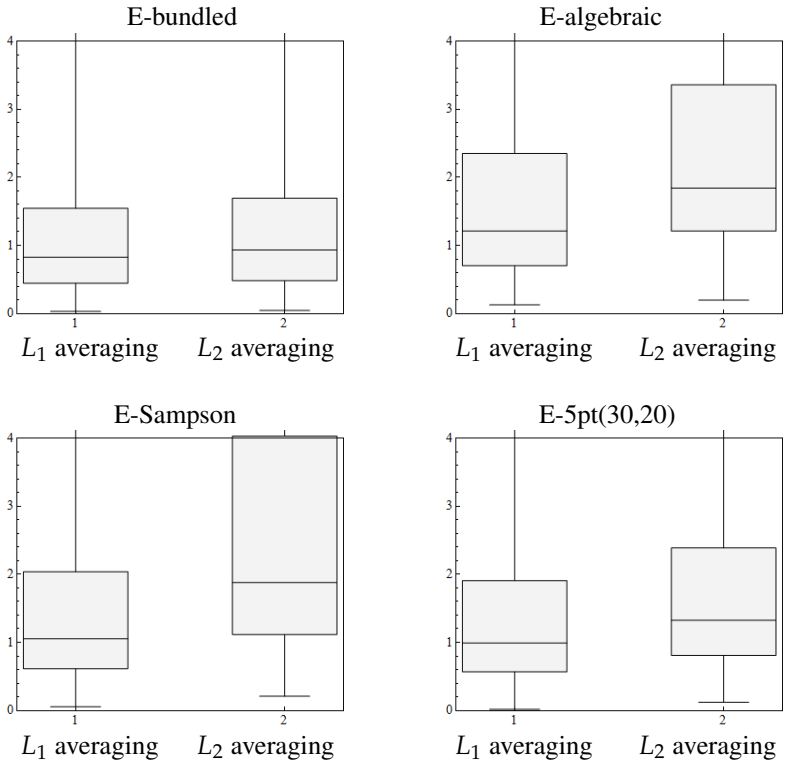


Figure 4.6: *Side-by-side comparison of the results of L_1 and L_2 averaging for each of the four methods of computing relative rotations.*

with a step size of 0.25. Results have shown that the L_1 is more robust to outliers than the L_q averaging algorithms, for $1 < q \leq 2$.

Our experimental results in all of the above scenarios confirm that the L_q averaging methods give superior results than the L_2 averaging methods in the presence of outliers.

Dataset and Noise: We take a set of 30 SPD matrices and try to estimate each matrix from its 20 noisy samples by performing averaging over these samples. For each matrix, we take a random 3×3 SPD matrix as the ground truth value, and generate its noisy samples by adding noise to its eigenvalues. In all of the cases, a Gaussian noise of zero mean and covariance of 0.25 times the eigenvalues is added to the eigenvalues of the ground truth matrix, that is, the i-th eigenvalue λ_i is modified by adding $\mathcal{N}(0, \sigma = 0.25\lambda_i)$ to it, where $\mathcal{N}(\cdot, \cdot)$ is a Gaussian distribution function. However, the percentage of outlier in the dataset vary in different experiments; outlier points are constructed by adding a noise of high magnitude to the eigenvalues of the original matrix.

Error Measure: In all of our experiments we assume that the ground truth is known. We take the original matrix as the ground truth and the results of averaging algorithms are compared with the ground truth value. Let $\mathtt{X}_i$ be the known ground truth values and $\mathtt{Y}_i$ be estimated means, the RMS error is then computed as,

$$\text{RMS Error} = \sqrt{\textstyle\sum_{i=1}^{k} \|\mathtt{X}_i - \mathtt{Y}_i\| / k}\,, \tag{4.14}$$

where $\|\cdot\|$ is the Frobenius norm and k is the total number of matrices for which averaging is performed.

Iterations and Starting Point: In case of the L_q averaging algorithms we terminate the algorithms after 30 iterations. Note that after first $10 - 15$ iterations of the L_q algorithms the change in error is very less and algorithms become stable. The proposed L_q averaging algorithms being iterative optimization techniques require a starting point. Instead of taking a random starting point we take the L_2 mean as a starting point. That is the reason why, in our results in fig. 4.7, the plots of the L_q averaging methods start from the L_2 mean.

4.5.2.1 Convergence Behavior

This experiment shows the convergence behavior of the L_q averaging algorithms, specifically L_1 averaging algorithm and L_2 averaging algorithm under the Log-Euclidean metric and the Affine-Invariant metric. In addition to noise, we modify 30% of the data points to represent outliers. We take the L_2 averaging result as a starting point for the L_1 averaging algorithms. Fig. 4.7 shows the results of the L_2 and L_1 averaging algorithms. The x-axis and y-axis show iterations and RMS error, respectively, where error is computed using (4.14). It is evident from the plots that the estimates of the L_1 averaging algorithms are closer to the ground truth than the estimates of the L_2 averaging algorithms.

We allow the L_1 algorithms to execute for 30 iterations but the change in error after 15 iterations is very less, fig. 4.7. Note that even after the few first iterations of the L_1 algorithms,

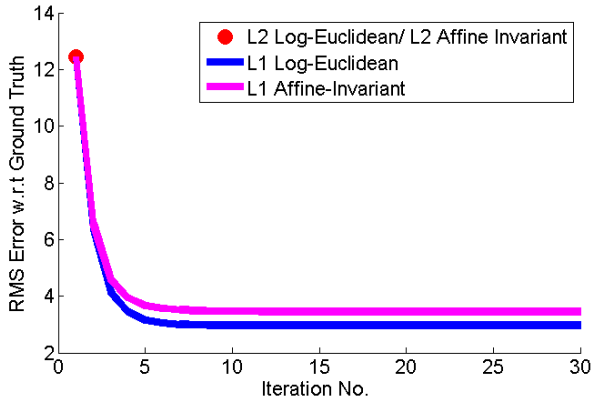


Figure 4.7: *Convergence Behavior: The above figure shows results of the L_2 averaging and L_1 averaging algorithms under the Log-Euclidean and Affine-Invariant metrics. The L_2 averaging methods have similar results for both the metrics and is therefore represented by a single (red) point in the above figure. However, the results of L_1 averaging algorithms are represented by blue and pink lines. In addition to noise, we modify 30% of the data to represent outliers. The above plots confirm the fact that in the presence of outliers the L_1 method gives superior results than the L_2 method. Furthermore, it shows that the results of the L_q averaging method under the Affine-Invariant metric are comparable to the results under the Log-Euclidean metric. However, the L_q averaging algorithm under the Log-Euclidean metric is preferred because of its provable convergence and simple form.*

the estimated SPD matrices are significantly closer to the ground truth than the L_2 means. Thus, the L_1 algorithms even for a few iterations are better than the L_2 averaging algorithms. Although, there is no proof of convergence of the L_1 averaging algorithm under the Affine-Invariant metric, the results of the algorithm are still promising. Furthermore, it is obvious from the convergence plot of the algorithm that the algorithm converges nicely without showing any abnormalities. Therefore, the algorithm can be used to find the L_1 mean. However, it is preferred to use the L_1 averaging algorithm under the Log-Euclidean metric because of its simple form and provable convergence.

4.5.2.2 Robustness Against Different Number of Outliers

In this experiment we compare the robustness of the L_q averaging methods for different values of q, specifically, q ranging from 1 to 2 with an increment of 0.25. In this case we only consider the L_q averaging algorithm under the Log-Euclidean metric. In order to show the robustness of the algorithms, in addition to noise, we add different number of outliers in the dataset. The percentage of outliers is varied from 0% to 40%, with an increment of 10%. RMS error between the estimated value and the known ground truth is computed using (4.14).

It is obvious from the plot in fig. 4.8 that in the presence of outliers the results of the L_1 averaging algorithm are more superior than the L_q averaging algorithms, for $1 < q \leq 2$. Furthermore, the results of the L_1 averaging does not vary much with change in percentage of outliers in data. As shown in fig. 4.8, the only point where the error of L_1 mean has higher

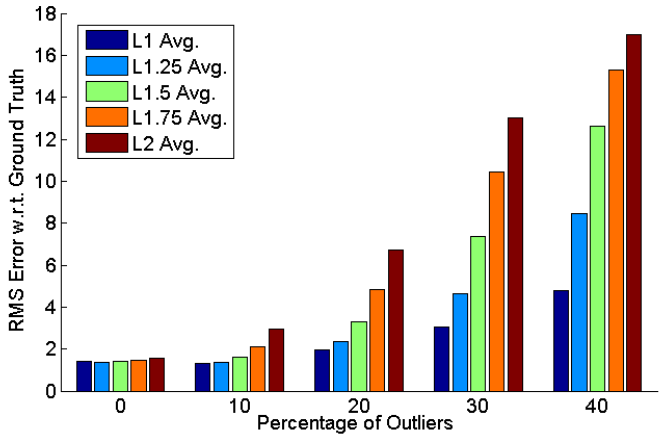


Figure 4.8: *Robustness against different proportion of Outliers (Log-Euclidean metric): The above figure shows the results of the L_q averaging algorithm for different values of q, specifically, for q ranging from* 1 *to* 2 *with an increment of* 0.25*. In addition to noise, we add different number of outliers to the dataset. In this case the percentage of outliers in the dataset is varied from* 0% *to* 40% *with an increment of* 10%*. The number of outliers is varied over the x-axis of the above figure. We only consider the L_q optimization under the Log-Euclidean metric. The above plots show that in the presence of outliers the results of the L_1 averaging algorithm are closer to the ground truth than the L_q averaging algorithms for $1 < q \leq 2$. However, when there are no outliers in data then the results of all the algorithms are roughly the same. Thus, the L_1 averaging method is preferred over other methods because of its robustness to outliers.*

error the rest of the algorithm is when there are no outliers in the data, a situation that rarely occurs in real data. Thus, in the absence of outliers the L_2 algorithm is recommended because it finds the solution in closed form. But in practice, having data free of outliers is not likely, thus the proposed L_1 averaging algorithm is very practical and is preferred.

4.6 Proofs

We dedicate this section to prove the convergence of the L_q Weiszfeld algorithm for points on a Riemannian manifold, specifically a proof of Theorem 4.6. Furthermore, we give proofs of two more theorems related to the continuity of logarithm map, that is Theorem 4.3 and Toponogov's Theorem for $\kappa \geq 0$, that is Theorem 4.2.

4.6.1 Convergence Theorem: Proof of Theorem 4.6

In this section we prove the convergence of the L_q optimization method for points on a complete Riemannian manifold with non-negative sectional curvature. Theorem 4.6 states that, given a set of points $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, $k > 2$, on a complete Riemannian manifold $\mathcal{M}$ with non-negative sectional curvature, such that not all of the given points lie on a single geodesic. Let all points $\mathbf{y}_i$, lie in a ball $B(\mathbf{o}, \rho)$ of radius $\rho < r_{\text{wcon}}/2$ centered at $\mathbf{o}$ and define $\mathcal{D} = \{\mathbf{x} \in \mathcal{M} \mid C_q(\mathbf{x}) \leq C_q(\mathbf{o})\}$. Starting from an initial point $\mathbf{x}^0$ in $\mathcal{D}$ the sequence of

points $(\mathbf{x}^t)$ obtained using (4.2), defined as,

$$\begin{aligned} W(\mathbf{x}^t) &= \exp_{\mathbf{x}^t}\left(\frac{\sum_{i=1}^k w_i^t \log_{\mathbf{x}^t}(\mathbf{y}_i)}{\sum_{i=1}^k w_i^t}\right) \quad \text{if } \mathbf{x}^t \notin \mathcal{Y} \\ &= \mathbf{y}_j \quad \text{if } \mathbf{x}^t = \mathbf{y}_j \end{aligned}, \tag{4.15}$$

where

$$w_i^t = d(\mathbf{x}^t, \mathbf{y}_i)^{q-2},$$

converges to the global minimum of C_q, or it will get stuck at some point with $\mathbf{x}^t = \mathbf{y}_i$. Note that the logarithm map is as defined in Theorem 4.3, that is the logarithm with respect to the weakly convex set W. Here, we give a proof that the sequence of points $(\mathbf{x}^t)$ converges to the L_q mean. Our proof of the convergence theorem takes place in several steps, according to the following outline based on Theorem 3.4 (p. 29) and Theorem 3.5 (p. 30) of chapter 3.

Outline 4.1. *Given an update function W and a strictly convex function C_q, to prove that the sequence $(\mathbf{x}^t)$ obtained using $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$ is convergent to the minimum of C_q we proceed as follows:*

1. *The update function W is continuous function, defined on a compact domain $\mathcal{D}$ and maps $\mathcal{D}$ to itself.*
2. *The value of C_q is non-increasing at every iteration.*
3. *The set S of Theorem 3.4 (p. 29) is a finite set containing the L_q minimum point and $\{\mathbf{y}_i\}$.*
4. *Since there is a finite number of accumulation points, the sequence $(\mathbf{x}^t)$ is in fact convergent, see Theorem 3.5 (p. 30).*
5. *If $(\mathbf{x}^t)$ converges to one of the given points, $\mathbf{y}_j$, then this is the minimum, except when $\mathbf{x}^t \in \{\mathbf{y}_i\}$ for any of the intermediate iterates.*
6. *Therefore, unless $\mathbf{x}^t$ gets stuck at $\mathbf{y}_i$, it converges to the L_q minimum.*

The proof will be completed by verifying each step of this outline in the following subsections, numbered in accord with the steps in the outline.

4.6.1.1 Continuous Update function W

The domain $\mathcal{D}$ is compact and lies in the open weakly convex set $B(\mathbf{o}, 2\rho)$. Since (as will be shown in the next point) the update function W decreases the cost function at every step, it maps $\mathcal{D}$ into $\mathcal{D}$. The update function W is continuous as a function of $\mathbf{x}$, since according to Theorem 4.3 both $\exp_{\mathbf{x}^t}(\mathbf{y}_i)$ and $\log_{\mathbf{x}^t}(\mathbf{y}_i)$ are continuous functions of $\mathbf{x}^t$. The apparent discontinuity when $\mathbf{x}^t = \mathbf{y}_i$ for some i is a removable singularity, as remarked in section 2.3 (p. 18).

4.6.1.2 Non-Increasing L_q Cost

Below we show in lemma 4.7 that the cost function C_q is a non-increasing function under the update function W, that is $C_q(W(\mathbf{x})) \leq C_q(\mathbf{x})$ with equality only when $W(\mathbf{x}) = \mathbf{x}$. In the update step, an optimal update is first computed in the tangent space and then this updated point is projected back to the manifold. It is therefore important to show that a decrease in the L_q cost in the tangent space also results in a decrease in the L_q cost on the manifold. The following lemma shows that the update function (4.2) results in a decrease in the value of C_q.

Lemma 4.7. *For the update function W defined in (4.2), we have $C_q(W(\mathbf{x})) \leq C_q(\mathbf{x})$, where equality holds only when $W(\mathbf{x}) = \mathbf{x}$.*

Proof. The inequality will be shown for any point $\mathbf{x}^t$ and $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$. Let $C_2^{\mathbf{w}}$ be a weighted L_2 function on $\mathcal{M}$,

$$C_2^{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{k} w_i^t \, d(\mathbf{x}, \mathbf{y}_i)^2$$

where $w_i^t = 1/d(\mathbf{x}^t, \mathbf{y}_i)^{q-2}$. Let $\tilde{\mathbf{y}}_i$ and $\tilde{\mathbf{x}}^{t+1}$ be the corresponding points in the tangent space at $\mathbf{x}^t$, under the logarithm map. Note that $\tilde{\mathbf{x}}^t = \mathbf{0}$. A corresponding weighted L_2 cost function $\widetilde{C}_2^{\mathbf{w}}$ is defined in the tangent space $T_{\mathbf{x}^t}\mathcal{M}$ of $\mathcal{M}$, as

$$\widetilde{C}_2^{\mathbf{w}}(\tilde{\mathbf{x}}) = \sum_{i=1}^{k} w_i^t \, d(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_i)^2 \, . \tag{4.16}$$

Note that $d(\mathbf{x}^t, \mathbf{y}_i) = d(\tilde{\mathbf{x}}^t, \tilde{\mathbf{y}}_i)$, so $\widetilde{C}_2^{\mathbf{w}}(\tilde{\mathbf{x}}^t) = C_2^{\mathbf{w}}(\mathbf{x}^t)$. The updated point $\tilde{\mathbf{x}}^{t+1}$ in the tangent space is the global minimizer of (4.16), so

$$\widetilde{C}_2^{\mathbf{w}}(\tilde{\mathbf{x}}^{t+1}) \leq \widetilde{C}_2^{\mathbf{w}}(\tilde{\mathbf{x}}^t) = C_2^{\mathbf{w}}(\mathbf{x}^t) = C_q(\mathbf{x}^t) \, . \tag{4.17}$$

From the Toponogov Comparison Theorem for open weakly convex sets, Theorem 4.2, the distance between two points on a positively curved manifold is less than the distance between their images under the log map, that is $d(\mathbf{x}^{t+1}, \mathbf{y}_i) \leq d(\tilde{\mathbf{x}}^{t+1}, \tilde{\mathbf{y}}_i)$. This implies that

$$C_2^{\mathbf{w}}(\mathbf{x}^{t+1}) \leq \widetilde{C}_2^{\mathbf{w}}(\tilde{\mathbf{x}}^{t+1}) \leq C_q(\mathbf{x}^t) \, ,$$

the last inequality from (4.17). The conclusion of the theorem now follows from lemma 3.6 (p. 35) by setting $a_i = d(\mathbf{x}^t, \mathbf{y}_i)$ and $b_i = d(\mathbf{x}^{t+1}, \mathbf{y}_i)$. □

Thus, under the update function W (4.2) the L_q cost function is a non-increasing function.

4.6.1.3 Finite set S

Under the update function W (4.2) the value of C_q remains constant in successive iterations either when $\mathbf{x} = \mathbf{y}_j$ where $W(\mathbf{y}_j) = \mathbf{y}_j$, or when $\mathbf{x}$ is a minimum of C_q. From Theorem 4.4, the L_q minimum $\mathbf{x}^*$ is unique in B, and for this point $W(\mathbf{x}^*) = \mathbf{x}^*$. The set S in Theorem 3.4 (p. 29) is a finite set because it is the union of the L_q minimum point $\mathbf{x}^*$ and $\{\mathbf{y}_i\}$.

4.6.1.4 Convergent Sequence $(\mathbf{x}^t)$

From Theorem 3.5 (p. 30) the sequence $(\mathbf{x}^t)$ is convergent. When the sequence of points $(\mathbf{x}^t)$ converges to a point other than $\mathbf{y}_j$, then $(\mathbf{x}^t)$ converges to the L_q minimum $\mathbf{x}^*$. However, when the L_q minimum is one of the points $\mathbf{y}_j$ then then the following section shows that this point satisfies the minimum point condition of lemma 4.5.

4.6.1.5 Convergence to a point $\mathbf{y}_j$

In this section we show that when $\mathbf{x}^t$ converges to one of the given points $\mathbf{y}_j$ without getting stuck at $\mathbf{y}_i$, then this point satisfies the conditions of lemma 4.5 and is a stationary point of C_q. When a minimum point is not one of the given points $\mathbf{y}_j$ then the L_q function is differentiable at the minimum point, even for $q = 1$. However, when the L_q minimum is one of the points $\mathbf{y}_j$ then the following lemma shows that for $1 < q < 2$ the gradient of the cost function vanishes at this point, while for $q = 1$ the gradient of the L_q function omitting the entry corresponding to $\mathbf{y}_j$ has a norm no greater than one.

Lemma 4.8. *For $1 \leq q < 2$, if the limit of the sequence $(\mathbf{x}^t)$ is one of the points $\mathbf{y}_j$, then $\mathbf{y}_j$ is the minimum point of C_q, except when any of the intermediate iterates $\mathbf{x}^t$ is equal to one of the $\mathbf{y}_i$ and the iteration gets stuck.*

Proof. Suppose that the sequence $\mathbf{x}^t$ converges to one of the points $\mathbf{y}_i$, which we take to be $\mathbf{y}_1$ for simplicity. Our goal is to invoke lemma 4.5 to show that $\mathbf{y}_1$ is a minimum.

Recall the definition of limit in a topological space: $x_i \to x^*$ if for every open set $\mathcal{O}$ containing x^* there exists an N such that $x_i \in \mathcal{O}$ for $i > N$.

For simplicity, write $\log(\mathbf{x}, \mathbf{y})$ instead of $\log_{\mathbf{x}}(\mathbf{y})$, and recall from Theorem 4.3 that as a function from $D \times D$ into $T\mathcal{M}$, this is continuous in both arguments. The update function is defined as

$$\mathbf{x}^{t+1} = \exp_{\mathbf{x}^t}\left(\frac{\sum_{i=1}^k w_i^t \log(\mathbf{x}^t, \mathbf{y}_i)}{\sum_{i=1}^k w_i^t}\right),$$

or equivalently,

$$\log(\mathbf{x}^t, \mathbf{x}^{t+1}) = \frac{\sum_{i=1}^k w_i^t \log(\mathbf{x}^t, \mathbf{y}_i)}{\sum_{i=1}^k w_i^t},$$

where $w_i^t = \| \log(\mathbf{x}^t, \mathbf{y}_i)\|^{q-2}$ and $\mathbf{x}^t$ is the estimate of the L_q minimum at iteration t. By a small rearrangement one sees that

$$\begin{aligned}(\log(\mathbf{x}^t, \mathbf{x}^{t+1}) - \log(\mathbf{x}^t, \mathbf{y}_1))\, w_1^t = \\ \sum_{i=2}^k w_i^t \log(\mathbf{x}^t, \mathbf{y}_i) - \log(\mathbf{x}^t, \mathbf{x}^{t+1}) \sum_{i=2}^k w_i^t .\end{aligned} \tag{4.18}$$

As $t \to \infty$ the limit of the right-hand side of (4.18) becomes

$$\sum_{i=2}^{k} \frac{\log(\mathbf{y}_1, \mathbf{y}_i)}{\|\log(\mathbf{y}_1, \mathbf{y}_i)\|^{2-q}} = \nabla \widehat{C}_q(\mathbf{y}_1), \tag{4.19}$$

with notation as in lemma 4.5. Note that this step uses the continuity of the logarithm, and also the continuity of the Riemannian metric, and hence norm on $T\mathcal{M}$. Now, turning to the left hand side, one continues:

$$\begin{aligned} \left\| \nabla \widehat{C}_q(\mathbf{y}_1) \right\| &= \lim_{t\to\infty} \left\| \left(\log(\mathbf{x}^t, \mathbf{x}^{t+1}) - \log(\mathbf{x}^t, \mathbf{y}_1)\right) w_1^t \right\| \\ &= \lim_{t\to\infty} \left\| \left(\log(\mathbf{y}_1, \mathbf{x}^{t+1}) - \log(\mathbf{y}_1, \mathbf{y}_1)\right) w_1^t \right\| \\ &= \lim_{t\to\infty} \frac{d(\mathbf{x}^{t+1}, \mathbf{y}_1)}{d(\mathbf{x}^t, \mathbf{y}_1)}\, d(\mathbf{x}^t, \mathbf{y}_1)^{q-1}. \end{aligned} \tag{4.20}$$

Once more, continuity of the logarithm in the first argument justifies the step to the second line here. The proof of the lemma is now completed using lemma 3.9 (p. 37), in the same way as in lemma 3.8 (p. 36), since the limit is equal to 0 for q > 1 or is less than 1 for q = 1.

Now, the cases $q = 1$ and $q > 1$ must be dealt differently because when $q > 1$ the cost function C_q is differentiable at $\mathbf{y}_1$, whereas when $q = 1$ it is not. Of course, the given limits may not exist in either case. In applying lemma 3.9 (p. 37), to the right hand side of (4.20), however, the limit is known to exist and equal $\|\nabla \widehat{C}_q(\mathbf{y}_1)\|$.

Consider the case $q > 1$. Then in (4.20) the term $\|\mathbf{x}^t - \mathbf{y}_1\|^{q-1}$ converges to zero, since $\mathbf{x}^t \to \mathbf{y}_1$. It follows from lemma 3.9 and (4.20) that

$$\|\nabla C_q(\mathbf{y}_1)\| = \|\nabla \widehat{C}_q(\mathbf{y}_1)\| = 0$$

so $\mathbf{y}_1$ is a stationary point (hence global minimum) of C_q.

In the case $q = 1$, lemma 3.9 and (4.20) yield

$$\|\nabla \widehat{C}_q(\mathbf{y}_1)\| = \lim_{t\to\infty} \frac{d(\mathbf{x}^{t+1}, \mathbf{y}_1)}{d(\mathbf{x}^t, \mathbf{y}_1)} \leq 1$$

which is the condition given in lemma 4.5 for $\mathbf{y}_1$ to be a minimum of the cost function. □

4.6.1.6 Convergence to the L_q minimum

Thus, the sequence of points $(\mathbf{x}^t)$ obtained using the update function W (4.2) converges to the L_q minimum, except when some $\mathbf{x}^t \in \{\mathbf{y}_i\}$ for any of the intermediate iterates.

This completes the proof of Theorem 4.6.

4.6.2 Continuity of Logarithm map: Proof of Theorem 4.3

In this section we give a proof of Theorem 4.3, which shows the continuity of a logarithm map defined on an open weakly convex set. If W is an open weakly convex set in a complete Riemannian manifold $\mathcal{M}$, and $\mathbf{x}, \mathbf{y}$ are two points in W, define $\log_{\mathbf{x}}(\mathbf{y})$ to be the vector $\mathbf{v}$ in $T_{\mathbf{x}}\mathcal{M} \subset T\mathcal{M}$ such that $\exp_{\mathbf{x}}(\mathbf{v}) = \mathbf{y}$, and $\exp_{\mathbf{x}}(t\mathbf{v}) \in W$ for all $t \in [0,1]$. Then $\log_{\mathbf{x}}(\mathbf{y})$ as a map from $W \times W$ to $T\mathcal{M}$ is continuous in both variables.

Proof. It is well known (e.g. [Lang, 1999, p. 107]) that the Riemannian exponential is continuous as a map $\exp\colon T\mathcal{M} \to \mathcal{M}$. Let $\pi : T\mathcal{M} \to \mathcal{M}$ be the projection to the base space of the tangent bundle, and define the continuous map $\overline{\exp}\colon T\mathcal{M} \to \mathcal{M} \times \mathcal{M}$ by $X \mapsto (\pi(X), \exp(X))$. Note that X is mapped to the start and end points of the geodesic defined by X.

Now let $W \subset \mathcal{M}$ be an open weakly convex set in $\mathcal{M}$, and for each $s \in [0,1]$ define

$$O_s^W = \{X \in T\mathcal{M} \mid \overline{\exp}(sX) \in W \times W\},$$

where sX is in the tangent space $T_{\pi(X)}$. Further, define

$$O_\cup^W = \{(s, X) \in [0,1] \times T\mathcal{M} \mid \overline{\exp}(sX) \in W \times W\}\,.$$

Each O_s^W can be seen as a cross-sectional slice of $O_\cup^W$, which itself is a union of all the slices for $s \in [0,1]$. Since $\overline{\exp}(sX)$ is continuous in both s and X, and $W \times W$ is open, $O_\cup^W$ is open in $[0,1] \times T\mathcal{M}$. Next, define

$$\begin{aligned} O_\cap^W &= \{X \in T\mathcal{M} \mid \overline{\exp}(sX) \in W \times W \text{ for all } s \in [0,1]\} \\ &= \bigcap_{s \in [0,1]} O_s^W\,. \end{aligned}$$

This is an infinite intersection, but since $[0,1]$ is compact, and $O_\cup^W$ is open, it follows that $O_\cap^W$ is open in $T\mathcal{M}$.

Note that for $s \in [0,1]$, $\exp(sX)$ traces out the (unique) geodesic in W from $\pi(X)$ to $\exp(X)$. Therefore, $\overline{\exp}$ is an injective map from $O_\cap^W$ onto $W \times W$, and we can define a *Riemannian logarithm*, $\log_W\colon W \times W \to O_\cap^W$ by

$$\log_W(p,q) = \overline{\exp}^{-1}(p,q).$$

Since $\log_W(p,q) \in T_p\mathcal{M}$ and $\exp_p(\log_W(p,q)) = q$ by definition, this is a well defined two-variable version of the usual pointwise Riemannian logarithm.

We show that $\log_W(\cdot,\cdot)$ is continuous by applying *invariance of domain* (e.g. [Dold, 1995, Proposition IV.7.4]), which states that an injective continuous mapping from $\mathbb{R}^m$ to $\mathbb{R}^m$ has a continuous inverse (restricted to its image). This can be applied to the mapping $\overline{\exp}$, which is an injective mapping between two open sets $O_\cap^W$ and $W \times W$. Although these sets are not subsets of any $\mathbb{R}^m$, they are both open subsets of $2n$-dimensional manifolds $T\mathcal{M}$ and $W \times W$ respectively. Since continuity is a local property, and the manifolds are locally homeomorphic to $\mathbb{R}^{2n}$, the result follows. □

4.6.3 Toponogov's Theorem for $\kappa \geq 0$: Proof of Theorem 4.2

We now give a proof of Theorem 4.2 that compares distances in an open weakly convex set in a Riemannian manifold of non-negative sectional curvature with distances in the tangent space. Let W be an open weakly convex set in $\mathcal{M}$, a manifold of non-negative sectional curvature. Let $\mathbf{q}$, $\mathbf{p}_1$ and $\mathbf{p}_2$ be three points in W. Then, Theorem 4.2 states that the following inequality holds,

$$d(\mathbf{p}_1, \mathbf{p}_2) \leq d(\log_{\mathbf{q}}(\mathbf{p}_1), \log_{\mathbf{q}}(\mathbf{p}_2)) = \| \log_{\mathbf{q}}(\mathbf{p}_1) - \log_{\mathbf{q}}(\mathbf{p}_2) \| .$$

Proof of Theorem 4.2: Consider a *hinge* in W consisting of the two geodesics from $\mathbf{q}$ to $\mathbf{p}_1$ and $\mathbf{p}_2$, meeting at $\mathbf{q}$. Under the logarithm map $\log_{\mathbf{q}}$ the three points map to $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{p}}_i$, where $\tilde{\mathbf{q}} = \mathbf{0}$ in the vector space $T_{\mathbf{q}}\mathcal{M}$. These three points form a hinge with the same parameters as the hinge formed by $\mathbf{q}$, $\mathbf{p}_1$ and $\mathbf{p}_2$ in W, namely, $d(\mathbf{q}, \mathbf{p}_i) = d(\tilde{\mathbf{q}}, \tilde{\mathbf{p}}_i)$, and $\angle(\mathbf{q}\mathbf{p}_i, \mathbf{q}\mathbf{p}_2) = \angle(\tilde{\mathbf{q}}\tilde{\mathbf{p}}_i, \tilde{\mathbf{q}}\tilde{\mathbf{p}}_2)$.

The desired conclusion is then seen as a particular case of Toponogov's theorem [Cheeger and Ebin, 1975] which states that $d(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2) \geq d(\mathbf{p}_1, \mathbf{p}_2)$.

However, Toponogov's theorem is true only under certain conditions. One required condition is that one of the geodesics $\mathbf{q}\,\mathbf{p}_i$ is minimizing. In the present case, this is true in the sense that the geodesic is minimizing in W, but not necessarily in $\mathcal{M}$. A further condition, seemingly always present in statements of the theorem, is that the manifold is geodesically complete, which may be true for $\mathcal{M}$, but it is not true for W. Therefore, we cannot apply Toponogov's theorem directly in its usual stated form, either to $\mathcal{M}$ or to W. Consequently, it is necessary to verify that Toponogov's theorem holds in this particular case.

We sketch a proof below, based on the proof given in [Meyer, 1989]. Since the main outline of the proof is the same as in that paper, it is sufficient to give a somewhat brief proof here, referring the reader to [Meyer, 1989] for more details.

The proof set out in [Meyer, 1989] can be significantly simplified for the present purposes. First, Meyer deals with the case where the sectional curvature satisfies $\kappa \geq \kappa_0$. We are only interested in the case $\kappa \geq 0$, which simplifies things. Secondly, Meyer accounts for the case when the distance function is not differentiable. This can occur when there are conjugate points along the geodesics, and it complicates the proof substantially. In our case, because of the absence of conjugate points in W, this complication can be avoided.

The proof is set out in three steps.

Step 1. Triangle distance inequality. Consider the triangle formed by points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{q}$ in W and the geodesics joining them. Let $\tilde{\mathbf{p}}_1$, $\tilde{\mathbf{p}}_2$ and $\tilde{\mathbf{q}}$ be the vertices of a triangle in $\mathbb{R}^n$ with the same side lengths (note: this is not a corresponding hinge, in the sense that the angles are not the same). Existence of this triangle is guaranteed because the sides of the triangle in W satisfy the triangle inequality, according to lemma 4.1. Now, parametrize the two edges $\mathbf{p}_1\mathbf{p}_2$ and $\tilde{\mathbf{p}}_1\tilde{\mathbf{p}}_2$ by arc-length; they have the same length by construction. Let $\mathbf{p}_t$ and $\tilde{\mathbf{p}}_t$ be corresponding points with the same parameter value t. The first step of the proof is to show that $d(\mathbf{q}, \mathbf{p}_t) \geq d(\tilde{\mathbf{q}}, \tilde{\mathbf{p}}_t)$.

For a fixed $\mathbf{q}$, define $f(\mathbf{p}) = d(\mathbf{q}, \mathbf{p})$ and $g(\mathbf{p}) = f(\mathbf{p})^2/2$. Similarly, define $\tilde{f}$ and $\tilde{g}$ in terms of distances in $\mathbb{R}^n$. The major technical point concerns the Hessian of the function $g(\mathbf{p})$

in $\mathcal{M}$. Provided $\kappa \geq 0$ in W, the following operator inequality holds.

$$\mathrm{Hess}_g \leq \mathtt{I} \,. \tag{4.21}$$

where $\mathtt{I}$ is the identity. This is to be interpeted as meaning that if Hess_g is expressed as a matrix in a local Riemannian coordinate system, then, $\mathtt{I} - \mathrm{Hess}_g$ is positive semi- definite. Therefore, for two vector fields X and Y, the inequality $\mathrm{Hess}_g(X,Y) \leq \langle X, Y\rangle$ holds.

This result is shown to hold for a "local" distance function (such at the distance d_W defined as geodesic length in the open set W), provided the path from $\mathbf{q}$ to $\mathbf{p}$ does not extend beyond the first conjugate point. Since there are no conjugate points in W, this inequality holds there. This is the main technical part of the proof (for details see [Meyer, 1989]), and the only place where Riemannian geometry, or any information about the curvature is used.

Now, let $\gamma(t)$ be a geodesic in $\mathcal{M}$, parametrized by arc length, and let $h(t) = g(\gamma(t))$; thus, h is a function from $\mathbb{R}$ to $\mathbb{R}$. Its second derivative is given by

$$h''(t) = \mathrm{Hess}_g(\gamma', \gamma') \leq \langle \gamma', \gamma' \rangle = 1 \,.$$

On the other hand, for a straight line in R^n, it is easily computed that $\tilde{h}''(t) = 1$.

Now, apply this to the geodesic γ, defined for $t \in [0, c]$, joining $\mathbf{p}_1$ to $\mathbf{p}_2$. Similarly, let $\tilde{\gamma}$ be the line (of the same length) joining $\tilde{\mathbf{p}}_1$ to $\tilde{\mathbf{p}}_2$ in R^n. With $h = g \circ \gamma$ and $\tilde{h} = \tilde{g} \circ \tilde{\gamma}$ defined as above, let $\lambda(t) = h(t) - \tilde{h}(t)$.

We wish to prove that $\lambda(t) \geq 0$ for all $t \in [0, c]$. This means that $g(\gamma(t)) \geq \tilde{g}(\tilde{\gamma}(t))$, or $f(\gamma(t)) \geq \tilde{f}(\tilde{\gamma}(t))$. In other words,

$$d(\tilde{\mathbf{q}}, \tilde{\gamma}(t)) \leq d(\mathbf{q}, \gamma(t)) \,. \tag{4.22}$$

Since the two triangles have sides of the same length, it follows that $h(0) = \tilde{h}(0) = \mathrm{dist}(\mathbf{q}, \mathbf{p}_1)$. Similarly, $h(c) = \tilde{h}(c) = \mathrm{dist}(\mathbf{q}, \mathbf{p}_2)$. Thus $\lambda(0) = \lambda(c) = 0$. The proof is completed by showing that λ is a concave function. This follows from the estimates of $h''(t) \leq 1$ and $\tilde{h}''(t) = 1$, since they imply that $\lambda''(t) \leq 0$.

Step 2. Angle inequality The next step is to extend this length inequality to any chord of the triangle, that is, any geodesic between points on two sides of the triangle. Thus, let $\mathbf{a}_s$ be a point on the edge $\mathbf{q}\mathbf{p}_1$ and $\tilde{\mathbf{a}}_s$ the corresponding point on the side $\tilde{\mathbf{q}}\tilde{\mathbf{p}}_1$. Then applying the argument of the previous section to the triangle $\mathbf{q}\mathbf{p}_1\mathbf{p}_t$ a simple argument (see [Meyer, 1989]) shows that $d(\mathbf{a}_s, \mathbf{p}_t) \geq d(\tilde{\mathbf{a}}_s, \tilde{\mathbf{p}}_t)$. By letting these points approach the vertex $\mathbf{p}_1$ one deduces that $\tilde{\alpha}_1 \leq \alpha_1$ where these are the angles of the triangle at $\tilde{\mathbf{p}}_1$ and $\mathbf{p}_1$ respectively.

By symmetry, this inequality holds equally well for all angles of the triangle.

Step 3. Hinge inequality Toponogov's hinge inequality now follows directly from the following lemma.

Lemma 4.9. *In $\mathbb{R}^n$, let $\mathbf{a}$ and $\mathbf{b}$ be points on two lines meeting at a point $\mathbf{c}$. Let α be the angle between the two lines. Then $d(\mathbf{a}, \mathbf{b})$ increases monotonically as α increases from 0 to π.*

Since the angle $\angle\tilde{\mathbf{q}} \leq \angle\mathbf{q}$, but $d(\mathbf{p}_1, \mathbf{p}_2) = d(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2)$, it follows that if the angle $\angle\tilde{\mathbf{q}}$ is increased to equal $\angle\mathbf{q}$, then the distance $d(\tilde{\mathbf{p}}_1, \tilde{\mathbf{p}}_2)$ is increased as well, giving the required

result.

4.7 Remarks and Extensions

4.7.1 Bounds for Convergence on $\mathrm{SO}(3)$

The convergence of the L_q algorithm to a global minimum was shown under the condition that all the points $\mathbf{y}_i$ lie in a convex ball of radius $\rho < r_{\text{wcon}}/2$ around the initial estimate. This is probably the best possible for $\mathrm{SO}(3)$, as it shows that the algorithm will converge to the minimum if the points lie in a ball $B(\mathbf{x}^0, \pi/2)$. If this condition does not hold, then one can easily find examples where the minimum lies outside of a ball containing the $\mathbf{y}_i$; see [Hartley et al., 2013].

4.7.2 A Flexible Approach

In the algorithm the logarithm map $\log_{\mathbf{x}^t}(\mathbf{y}_i)$ used is the one defined by geodesics lying in the weakly-convex set $B(\mathbf{o}, 2\rho)$. This is not always the logarithm of smallest norm, unless $\mathbf{x}$ remains inside a convex (not just weakly-convex) set. However, this can be assured as long as the $\mathbf{y}_i$ lie in a ball of radius less than $r_{\text{conv}}/2$. Such a ball is potentially half the size of the one of radius $\rho < r_{\text{wcon}}/2$ used in the theorem.

The condition can be relaxed to a condition that the $\mathbf{y}_i$ lie inside a ball of radius $\rho < r_{\text{conv}}$ by addition of one extra step to the algorithm. Note that $r_{\text{conv}} \geq r_{\text{wcon}}/2$, so this may strengthen the result. The extra step uses (4.3). If an intermediate estimate $\mathbf{x} = \mathbf{x}^t$ lies outside of the convex ball $B(\mathbf{o}, \rho)$, then it may be replaced by the point $\mathbf{x}'$ defined in (4.3). The mapping $\mathbf{x} \mapsto \mathbf{x}'$ is continuous. By adding this correction to the update step, all iterations remain inside the convex ball $B(\mathbf{o}, \rho)$.

4.7.3 More on $\mathrm{SO}(3)$

Finally, in $\mathrm{SO}(3)$, if all points $\mathbf{y}_i$ lie inside a ball of radius $\rho < r_{\text{conv}} = \pi/2$, then the update function W will map $B(\mathbf{o}, \rho)$ into itself always, so the algorithm works without modification in this case, and convergence is assured. This is because the update step may be thought of as finding a weighted centroid in the tangent space $T_{\mathbf{x}^t}\mathcal{M}$ of the points $\log_{\mathbf{x}^t} \mathbf{y}_i$. This weighted centroid must remain inside the convex hull of the $\log_{x^t} \mathbf{y}_i$, and this convex hull is mapped back by $\exp_{\mathbf{x}^t}$ to a point in the convex hull of the points $\mathbf{y}_i$, and hence back into the ball $B(\mathbf{o}, \rho)$.

4.7.4 More on the Initial Point

Generally a randomly selected starting point will result in convergence of the L_q Weiszfeld algorithm to the L_q minimum without getting stuck at $\mathbf{y}_i$. However, if such a condition occurs where $\mathbf{x}^t = \mathbf{y}_j$ then the L_q Weiszfeld algorithm, and even the original Weiszfeld algorithm, gets stuck at that point. A simple strategy to escape this situation is to move the current solution $\mathbf{x}^t$ in the descent direction and continue with the algorithm. This condition is not very likely to occur, and moreover can be avoided by a careful selection of a starting point $\mathbf{x}^0$, as explained below in Algorithm 4.2.

Algorithm 4.2. *Given a set of points, $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\} \in \mathbb{R}^N$ or $\mathcal{M}$, $k > 1$. Let $d(\mathbf{y}_i, \mathbf{y}_j)$, be the distance between two points, $\mathbf{y}_i$ and $\mathbf{y}_j$.*

1. *Among the $\mathbf{y}_i$ select the one with minimum cost: $\mathbf{x}^* = \operatorname{argmin}_j C_q(\mathbf{y}_j)$, where $C_q(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q$.*
2. *Compute the gradient of C_q and check $\mathbf{x}^* = \mathbf{y}_j$ for the minimality condition according to lemma 4.5. If it satisfies the condition then $\mathbf{x}^*$ is the required minimum and the algorithm is complete.*
3. *Otherwise, displace $\mathbf{x}^*$ in the downhill gradient direction of $\widehat{C}_q(\mathbf{x})$ to obtain $\mathbf{x}^0$. Backtrack if necessary to ensure $C_q(\mathbf{x}^0) < C_q(\mathbf{x}^*)$.*
4. *Repeat, $\mathbf{x}^{t+1} = W(\mathbf{x}^t)$ until convergence, where W is defined in (4.2).*

The initial point $\mathbf{x}^0$ so found has cost less than any of the points $\mathbf{y}_i$, and iterations of the algorithm from $\mathbf{x}^0$, can not again approach any of the $\mathbf{y}_j$. Thus Algorithm 4.2 ensures that the non-differentiability condition never occurs by a careful selection of a starting point for the algorithm. Hence the L_q Weiszfeld algorithm is guaranteed to converge to the minimum of $C_q(\mathbf{x})$.

4.8 Summary

In summary, we proposed an extension of the L_q Weiszfeld algorithm to find the L_q mean of a set of points on a Riemannian manifold of non-negative sectional curvature. In addition to the proof of convergence, we further relaxed the bounds on the maximum distance between points on manifold. In order to show the applicability of the proposed algorithm we solved the L_q rotation averaging problem and L_q averaging problem for SPD matrices. Our experiments strongly confirmed that L_q averaging gives superior and more robust results than L_2 averaging, and still at very competitive cost of time. As shown in the case of the rotation averaging, it is possible to get very good rotation estimates very quickly (3 minutes for the Notre Dame set) with a median accuracy of about one degree.

Chapter 5

Lq Optimization for Subspaces

In many computer vision applications information can be represented by affine subspaces, for example, multibody structure and motion, objects under different illumination settings, etc. This chapter presents Weiszfeld-inspired methods to find the L_q-closest-point, "intersection" point, to a set of affine subspaces, for $1 \leq q < 2$. Given a set of subspaces, the L_q-closest-point is defined as a point for which the sum of the q-th power of orthogonal distances to all the subspaces is the minimum. We propose two strategies to find the L_q-closest-point to a set of subspaces. We refer to the first approach as the *gradient descent approach* and the second approach as the *general approach*. We give theoretical proofs for the convergence of the proposed algorithms to a unique L_q minimum. In order to show the applicability of the proposed algorithm in computer vision we solve the triangulation problem by finding the L_q-closest-point to a set of lines in 3D space for the dinosaur dataset. Results have shown that in the presence of outliers the proposed L_q methods give superior results to the L_2-closest-point and L_2 bundle-adjustment methods.

In the following section we introduce the problem of finding the L_q-closest-point to subspaces followed by a brief discussion on its applications in computer vision. We then formulate the L_q-closest-point problem, define the notation and discuss some properties of the L_q cost function in section 5.2. After the problem formulation we propose and give proofs of convergence of the proposed L_q-closest-point algorithms, in section 5.4 and section 5.5. In the end, we present our experimental results for the triangulation problem on the Dinosaur dataset.

5.1 Introduction / Literature Review

This chapter presents provably convergent methods, based on the L_q Weiszfeld algorithm [Aftab et al.], to find the *L_q-closest-point* to a set of affine subspaces in $\mathbb{R}^N$, for $1 \leq q < 2$. Given a set of affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, possibly of different dimensions, we seek a point $\mathbf{x}$ for which the sum of the q-th power of orthogonal distances to the subspaces is the minimum. The minimization function is,

$$\min_{\mathbf{x} \in \mathbb{R}^N} \sum_{i=1}^{k} d(\mathbf{x}, \mathcal{S}_i)^q \,,$$

where $1 \leq q < 2$ and $d(\mathbf{X}, \mathcal{S}_i)$ is the orthogonal distance of a point $\mathbf{X}$ from the i-th subspace $\mathcal{S}_i$. We refer to the minimum of the above function as the *L_q-closest-point* to subspaces or simply the *L_q minimum*. The distance function $d(\mathbf{X}, \mathcal{S}_i)$ is always the minimum Euclidean-distance from the point to the subspace, equal to $\min_{\mathbf{y} \in \mathcal{S}_i} \|\mathbf{X} - \mathbf{y}\|$, where $\| \cdot \|$ represents the Euclidean 2-norm. Thus, the q in L_q indicates that we are minimizing the q-norm of the error-vector formed by the distances to all the subspaces; we are not considering the q-norm in $\mathbb{R}^N$.

The problem of finding the L_q mean or L_q-closest-point of a set of points (zero dimensional subspaces) in $\mathbb{R}^N$ is a special case of the problem of finding the L_q-closest-point to a set of subspaces. In the case of the L_q-closest-point to affine subspaces, given a set of affine subspaces and a current estimate of the L_q minimum, the sum of the q-th power of orthogonal distances to the subspaces is minimized, iteratively. Note that the orthogonal distances depend on the current estimate of the L_q minimum and its projections on the subspaces. Since the projection points change after every iteration, we take advantage of this property and propose two methods for finding the L_q-closest-point to subspaces. The first approach, referred as the *Gradient Descent approach*, finds an update by keeping the projection points fixed for an iteration. On the other hand, the second approach, referred as the *General approach*, finds an update by keeping both the current estimate and projection points as variables. Both the approaches are Weiszfeld-inspired and compute updates analytically.

In considering the q-norm, we are most interested in the case $q = 1$, which gives a high degree of robustness to outliers. However, considering the case $1 \leq q < 2$ presents no additional difficulty in theory or implementation. It does in fact have an additional advantage, since the distance function $d(\mathbf{X}, \mathcal{S})^q$ is differentiable for $q > 1$, but not for $q = 1$. Thus, one can avoid difficulties by considering values of q close to but not equal to 1, with no significant difference in numerical results.

Nearest subspace problems. In many computer vision applications, data is represented by linear subspaces in some Euclidean space. For example, subspaces are often used to represent multibody structure and motion, objects under different illumination settings, etc. The problem of finding an optimal point of intersection of higher dimensional affine subspaces has become an important component in a wide range of computer vision and pattern recognition applications. The proposed algorithm can be used to solve the problem of triangulation [Hartley and Sturm, 1997], [Triggs et al., 2000b], corner detection through the intersection of planes [Furukawa et al., 2009a], etc.

For $q = 2$ the problem of finding the L_q-closest-point to subspaces can be solved in closed form. Clearly, the L_2 method is more efficient than the proposed iterative technique but it is known that L_2 methods are not as robust to outliers as the L_q methods, for some values of q. Here, we are interested in finding a robust solution of the problem. Thus, we propose L_q Weiszfeld-inspired [Aftab et al.] methods.

In this chapter, we show that the L_q Weiszfeld algorithm [Aftab et al.] for points, that is zero dimensional subspaces, in $\mathbb{R}^N$ can be generalized to find a closest-point to a set of higher dimensional subspaces, such as lines, planes, subspaces or their mixture. Just like the L_q Weiszfeld algorithm, the proposed algorithms are iterative optimization techniques where updates are computed analytically. Moreover, the proposed algorithms are simple to under-

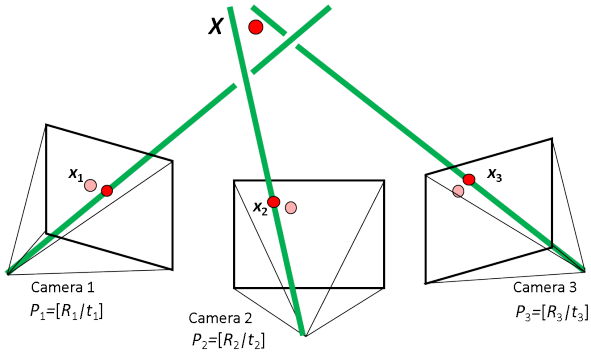


Figure 5.1: *Triangulation: Due to noise in image point measurements lines through the center of cameras and* $\mathbf{y}_i$ *do not intersect at point* $\mathbf{X}$.

stand and easy to code because an existing closed-form L_2 method can be modified to give a more robust L_q solution. In short, the proposed algorithms inherits all features of the L_q Weiszfeld algorithm.

An important point to note here is that the proposed algorithms find the L_q minimum even if the given subspaces have different dimensions. For example, the proposed algorithms can be used to find the L_q minimum to a set of lines and planes in $\mathbb{R}^N$. We give proofs of convergence of the proposed algorithms to the L_q minimum. These proofs follow from the proof of convergence of the L_q Weiszfeld algorithm [Aftab et al.].

In order to show the applicability of the proposed algorithms we consider the problem of triangulation [Hartley and Sturm, 1997; Hartley and Zisserman, 2004]. In triangulation we seek a point in 3D space that best represents a point of intersection of lines, where each line is passing through the center of camera and intersecting the image plane at the corresponding image point. Due to various types of noise in image point measurements these lines are a skewed form of the original lines, as shown in fig. 5.1. Therefore, these lines normally do not intersect at a single point in $3D$ space, possibly these lines may not intersect at all. So the triangulation problem is then reduced to the problem of finding the optimal point of intersection of 1-dimensional subspaces and can be solved using the proposed algorithms.

One can also find the vertices of the objects in the scenes that have dominant planar structure, for example architectural scenes [Schindler and Bauer, 2003; Taillandier, 2005; Dick et al., 2001], indoor scenes [Furukawa et al., 2009b; Müller et al., 2007; Wilczkowiak et al., 2003], aerial images [Ameri and Fritsch, 2000; Remondino and El-Hakim, 2006], Manhattan world [Vanegas et al., 2010; Werner and Zisserman, 2002; Furukawa et al., 2009a], building reconstruction from laser scanning data [Ma, 2004; Henry et al., 2010; Pu and Vosselman, 2009] and others, by finding the point of intersection of planes, each representing an adjacent planar face of the object, shown as red points in fig. 5.2. Ideally, we should be able to find the vertices of a planar object as the point of intersection of planes of the adjacent faces. But in practice, due to texture-poor surfaces, low resolution of images, lens distortion and various types of noise, the estimated planes may not be a good representation of planar faces. Clearly, when the point is defined by three planes then they intersect at a single point, possibly different from the ground truth point. On the other hand, if a corner point lies at the intersection of more

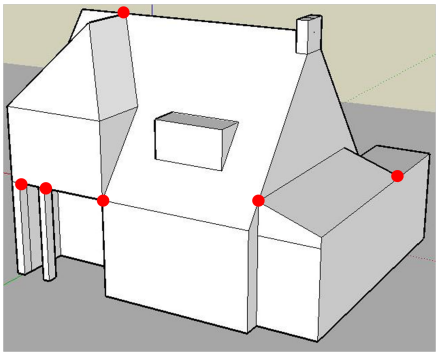


Figure 5.2: *Corner point estimation: The object in the figure has a strong geometric structure and corner points, in red, are at the intersection of more than 3 planar faces. In practice due to various types of noise, planes representing these faces are skewed and may not intersect at a single point, thus we need to find an optimal point of intersection of these planes. Image taken from http://russta.wordpress.com/category/sketch-up*

than three planes, indicated as red points in fig. 5.2, then the estimated planes may not intersect at a single point and may not even generate a single corner point.

The problem, then, is to find an optimal point of intersection of skewed planes, each representing a planar face. Thus, we can apply the proposed algorithms to find the L_q-closest-point to these planes. In order to improve the accuracy of results one can take several estimates for each of the planar faces and then find the intersection of these planes by using the proposed algorithm.

In this chapter we have shown that the L_q Wesizfeld algorithm can be used to find the L_q-closest-point of affine subspaces of any dimension, for $1 \leq q < 2$. The simplicity of the proposed L_q algorithms and the rapidity with which its iterative update may be computed make the proposed methods attractive.

5.2 L_q-Closest-Point to Subspaces

In this section we formulate the problem of finding the L_q-closest-point to subspaces and define notation that is used in the rest of the chapter. The L_q-closest-point, for $1 \leq q < 2$, to a set of subspaces is defined as a point for which the sum of the q-th powers of orthogonal distances to all the subspaces is the minimum, (fig. 5.3).

Given a set of affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, the L_q cost function is defined as

$$C_q(\mathbf{X}) = \sum_{i=1}^{k} d(\mathbf{X}, \mathcal{S}_i)^q = \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q . \tag{5.1}$$

where $1 \leq q < 2$ and $d(\mathbf{X}, \mathcal{S}_i)$ is the orthogonal distance of a point $\mathbf{X}$ from $\mathcal{S}_i$. Let $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ be the orthogonal projection of a point $\mathbf{X}$ onto $\mathcal{S}_i$, then the distance $d(\mathbf{X}, \mathcal{S}_i)$ is simply the Euclidean distance between $\mathbf{X}$ and $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$, that is $\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|$.

For simplicity it is assumed that the affine subspaces $\mathcal{S}_i$ do not intersect and are non-

parallel, though extensions of the algorithms to cover such cases are not difficult.

Orthogonal Projection: In our notation $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ represents the orthogonal projection of a point $\mathbf{X}$ on an affine subspace $\mathcal{S}_i$. We can write $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ as

$$\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}) = \mathbf{C}_i + \mathtt{A}_i\,(\mathbf{X} - \mathbf{C}_i)\,, \tag{5.2}$$

where $\mathtt{A}_i$ is an orthogonal projection matrix and $\mathbf{C}_i \in \mathcal{S}_i$ is taken to be the origin of an orthonormal basis of $\mathcal{S}_i$. Let $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d\}$ be the orthonormal basis of a d-dimensional subspace $\mathcal{S}$, the orthogonal projection matrix $\mathtt{A}$ can be computed as $\mathtt{A} = \sum_{i=1}^{d} \mathbf{e}_i \mathbf{e}_i^T$.

By substituting the value of $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ in $d(\mathbf{X}, \mathcal{S}_i)$, the distance function can be explicitly written as,

$$d(\mathbf{X}, \mathcal{S}_i) = \|\mathbf{X} - (\mathbf{C}_i + \mathtt{A}_i\,(\mathbf{X} - \mathbf{C}_i))\| = \|\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i))\|\,,$$

where $\mathtt{M}_i = \mathtt{I} - \mathtt{A}_i$.

Note that both $\mathtt{A}_i$ and $\mathtt{M}_i$ are orthogonal projection matrices or projectors. For a projection matrix $\mathtt{P}$ we have $\mathtt{P}^2 = \mathtt{P}$. Note that the matrix $\mathtt{P}$ is a symmetric matrix, that is $\mathtt{P} = \mathtt{P}^T$. Thus, we have, $\mathtt{P}^T\mathtt{P} = \mathtt{P}\mathtt{P} = \mathtt{P}^2 = \mathtt{P}$.

The above discussion simplifies the computation of the gradient of the squared distance function,

$$\nabla d(\mathbf{X}, \mathcal{S}_i)^2 = \nabla((\mathbf{X} - \mathbf{C}_i)^T\,\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i)) = 2\,\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i)\,. \tag{5.3}$$

In our formulations we use $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ to represent the orthogonal projection; however, it is not difficult to replace $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ with $(\mathbf{C}_i + \mathtt{A}_i\,(\mathbf{X} - \mathbf{C}_i))$ wherever desired.

Gradient of C_q: The Gradient of the L_q cost function (5.1) is

$$\nabla C_q(\mathbf{X}) = \sum_{i=1}^{k} w_i\,(\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))\,, \tag{5.4}$$

where $w_i = \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^{q-2}$.

5.2.1 Comparison with L_q Optimization for Points

As mentioned before, the problem of finding the L_q-closest-point to a given set of subspaces in $\mathbb{R}^N$ is closely related to the problem of finding the L_q mean of a set of points in $\mathbb{R}^N$. In this section we compare both the techniques.

In the case of the L_q optimization for points in $\mathbb{R}^N$, a point is obtained using a gradient descent approach that minimizes the sum of the q-th power of distances to all the given (fixed) points. Thus, starting from an initial estimate of the L_q minimum, after every iteration the only point that changes its location is the estimate of the L_q minimum, whereas given points are held fixed and are never changed. This process is repeated till convergence.

On the other hand, in the case of the L_q-closest-point to subspaces, once again we start

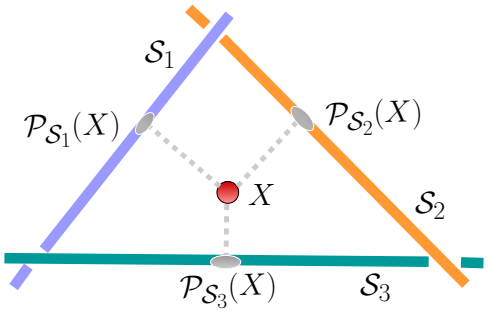


Figure 5.3: *L_q-closest-point to Subspaces: The above figure shows three subspaces (lines), $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. We seek a point $\mathbf{X}$ for which the sum of the q-th powers of orthogonal distances is minimum, that is $\sum_i d(\mathbf{X}, \mathcal{S}_i)^q$. Each grey point represents the orthogonal projection $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ of a red point $\mathbf{X}$, onto a corresponding subspace.*

from an initial estimate of the L_q minimum. This estimate is updated in the descent direction by minimizing the sum of the q-th powers of orthogonal distances to all the subspaces. Although subspaces are fixed in this scenario but we are minimizing the sum of orthogonal distances which are dependent on the current estimate of the minimum and its projection onto the given subspaces. Thus, unlike the L_q optimization for points in $\mathbb{R}^N$ where given points are held fixed during the minimization, in this case the location of these projected points change with a change in the location of the current estimate.

Thus, the problem of finding the L_q-closest-point to subspaces can be thought of as a problem of finding the L_q mean of a given set of points in $\mathbb{R}^N$, but instead of fixed points these points change their position after every iteration. Thus, the problem of finding the L_q-closest-point to subspaces is different from the problem of fining the L_q mean of a set of points.

We take advantage of this property of the L_q cost function for subspaces and define several solution strategies to find the L_q-closest-point, depending on whether projections of a current estimate are kept fixed or not during the estimation of an updated point.

5.2.2 Weighted L_2 Function

Another type of cost function of interest is a weighted L_2 cost function, which iteratively minimizes a weighted sum of squared orthogonal distances to given subspaces. Let $\mathbf{X}^0$ be an initial estimate of the L_q minimum. at iteration t, a weighted L_2 cost function is defined as

$$C_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} w_i^t \, d(\mathbf{X}, \mathcal{S}_i)^2 \,,$$

where $w_i^t = d(\mathbf{X}^t, \mathcal{S}_i)^{q-2}$ and $\mathbf{X}^t$ is an estimate of the L_q minimum at iteration t.

The weighted L_2 function takes two forms depending on whether the projected points are held fixed during the estimation of an updated point or they are kept as variables. Starting from an initial estimate $\mathbf{X}^0$ of the L_q minimum, at iteration t, an update is obtained by minimizing a

weighted L_2 cost function

$$\widetilde{C}_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} w_i^t \left\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\right\|^2, \tag{5.5}$$

where

$$w_i^t = d(\mathbf{X}^t, \mathcal{S}_i)^{q-2} = \left\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\right\|^{q-2} \tag{5.6}$$

and $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$ is the projection of $\mathbf{X}^t$ on $\mathcal{S}_i$. Note that in this case projected points are held fixed during the estimation of an update.

A slightly different form of the cost function $C_2^{\mathbf{w}}$,

$$\widehat{C}_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} w_i^t \left\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\right\|^2, \tag{5.7}$$

where $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ is the projection of $\mathbf{X}$ on subspace $\mathcal{S}_i$. In this case $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ are kept as variables.

The approach to solving for the minimum of the L_q cost (5.1) it to repeatedly find the minimum of the weighted L_2 cost function (5.5) or (5.7), where weights w_i are updated according to (5.6) after every iteration. As will be seen, the minimum of the weighted cost functions (5.5) or (5.7) may be computed in closed form.

A critical part of the proof of convergence of the proposed algorithms is to show that the minimization of the weighted L_2 cost functions results in a decrease in the L_q cost (5.1). To do this, we use a lemma 3.6 (p. 35).

5.2.3 Solution Strategies

In this section we discuss the proposed solution strategies for finding the L_q-closest-point to a given set of affine subspaces. As discussed before that the projection points change after each iteration, we take advantage of this behavior of projection points and propose the following two different minimization strategies:

- **Gradient Descent Approach:** In this approach an update is computed by minimizing $\widetilde{C}_2^{\mathbf{w}}(\mathbf{X})$, as defined in (5.5). Note that during the estimation of the updated solution all of the projection point are held fixed and these projection points depend on the solution at the previous iteration. These fixed projection points are represented by $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$. It will also be shown later in the chapter that the minimization of $\widetilde{C}_2^{\mathbf{w}}(\mathbf{X})$ (5.5) leads to an update function that has the same form as the update function in the case of the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$ [Aftab et al.].

- **General Approach:** Unlike the gradient descent form where projections are held fixed during the estimation of an updated solution, in this case both the estimate of minimum and projection points are updated simultaneously. The general approach minimizes the cost function $\widehat{C}_2^{\mathbf{w}}(\mathbf{X})$, as defined in (5.7).

5.2.4 L_q Cost Function Restricted on a Subspace

At this point it is important to define another variant of the L_q cost function (5.1) that will be used to find the L_q minimum on a subspace by restricting its iterates on it. This function finds a point on the subspace for which the sum of the q-th power of orthogonal distances to the rest of the subspaces is the minimum. A restricted L_q cost function on the j-th subspace is defined as,

$$C_q^j(\mathbf{X}) = \min_{\mathbf{X}\in\mathcal{S}_j} \sum_{i\neq j} d(\mathbf{X}, \mathcal{S}_i)^q \ .$$

Just like the L_q cost function there are two restricted versions of the weighted L_2 cost functions defined in section 5.2.2. These two variations of the weighted L_2 cost function are restricted version of $\widetilde{C}_2^{\mathbf{w}}(\mathbf{X})$ and $\widehat{C}_2^{\mathbf{w}}(\mathbf{X})$ and are represented by $\widetilde{C}_2^j(\mathbf{X})$ and $\widehat{C}_2^j(\mathbf{X})$, respectively.

An advantage of using the restricted functions is that if the L_q minimum lies on one of the subspaces then the minimization of a restricted cost function finds the minimum.

5.2.5 Properties of the L_q Cost Function

In this section we discuss some basic properties of the L_q cost function. Let $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$ be a set of non-parallel affine subspaces. The L_q cost function, being a sum of individual (convex) distance functions, is a convex function. Moreover, the L_q cost function is strictly convex, except when there exists a vector $\mathbf{V}$ that is parallel to all the subspaces. Since we assume that the given subspaces are non-parallel, such a vector does not exist and the cost function has a unique global minimum. The following lemma shows that the L_q cost function is strictly convex.

Lemma 5.1. *Given a set of non-parallel subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, the L_q cost function is strictly convex.*

For a detailed proof regarding the convexity of the L_q cost function see section 5.4.5. In what follows, we shall assume that the L_q cost function has a single global minimum.

Similar to what was shown in Weiszfeld [1937] for the classic Fermat-Weber problem, the minimum of this cost function may be classified as follows.

Lemma 5.2. *A point $\mathbf{X}^* \in \mathbb{R}^N$ is a minimum of the cost function (5.1) if and only if it satisfies one of the following conditions:*

1. *$\nabla C_q(\mathbf{X})$ vanishes at $\mathbf{X}^*$,*
2. *$q = 1$ and $\mathbf{X}^* \in \mathcal{S}_j$ and the gradient (omitting subspace $\mathcal{S}_j$) $\nabla C_q^j(\mathbf{X})$ has norm no greater than 1 and is orthogonal to $\mathcal{S}_j$.*

For the case $q > 1$ this follows easily, since the cost-function is differentiable. For $q = 1$, the proof is similar to the case where each S_i is a single point, as given in [Weiszfeld, 1937], and is therefore omitted.

For the simplicity of formulation, we assume that the L_q minimum does not lie at the intersection of subspaces. However, it is easy to extend the above definitions for such cases.

5.3 L_2-closest-point to Subspaces

The L_2 form of the problem is very simple and can be solved in closed form. Given a set of affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, the L_2-closest-point to subspaces minimizes the sum of the squared distances. The L_2 cost function is defined as,

$$C_2(\mathbf{X}) = \sum_{i=1}^{k} d(\mathbf{X}, \mathcal{S}_i)^2 = \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^2 ,$$

where $\|\cdot\|$ is the L_2 norm.

As described before in section 5.2 that the distance function can be explicitly written as,

$$d(\mathbf{X}, \mathcal{S}_i) = \|\mathbf{X} - (\mathbf{C}_i + \mathtt{A}_i\,(\mathbf{X} - \mathbf{C}_i))\| = \|\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i))\| ,$$

where $\mathtt{M}_i = \mathtt{I} - \mathtt{A}_i$, the matrix $\mathtt{A}_i$ is an orthogonal projection matrix for the i-th subspace and $\mathbf{C}_i \in \mathcal{S}_i$ is taken to be the origin of an orthonormal basis of $\mathcal{S}_i$.

From (5.3), the gradient of the L_2 cost function is,

$$\nabla C_2(\mathbf{X}) = \sum_{i=1}^{k} \nabla((\mathbf{X} - \mathbf{C}_i)^T\,\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i)) = \sum_{i=1}^{k} 2\,\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i) .$$

By equating the gradient of C_2 to zero, the L_2-closest-point to subspaces is computed as,

$$\mathbf{X} = \left(\sum_{i=1}^{k} \mathtt{M}_i\right)^{-1} \sum_{i=1}^{k} \mathtt{M}_i\,\mathbf{C}_i . \tag{5.8}$$

Thus, the L_2-closest-point to subspaces has a closed form solution.

We now turn our focus on the primary goal of this chapter, that is the L_q-closest-point to subspaces. In the rest of the chapter we discuss the algorithms to find the L_q-closest-point to subspaces.

5.4 L_q Solution Strategy I: Gradient Descent Approach

In this section we discuss the Gradient descent approach, as mentioned in section 5.2.3, to find the L_q-closest-point to a given set of subspaces. Given a set of non-parallel affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, at iteration t, a current estimate of the L_q minimum $\mathbf{X}^t$ is updated by keeping the projections of $\mathbf{X}^t$ fixed on the given subspaces. Thus, the update function in this case has the same form as the update function in the case of the L_q Weiszfeld algorithm [Aftab et al.]. The only difference in this case is that after every iteration these projection points are updated in accordance with the new location of the L_q minimum.

Let $\mathbf{X}^0$ be a random initial estimate of the L_q minimum. At iteration t, a current estimate

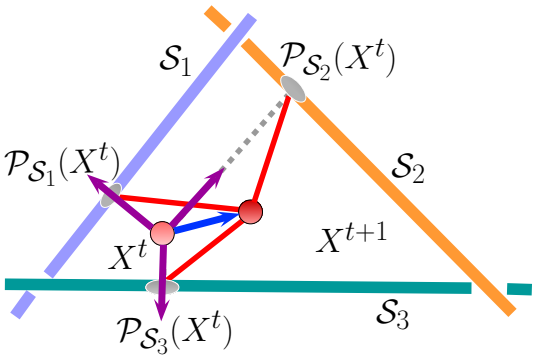


Figure 5.4: *L_q-closest-point to Subspaces (Gradient Descent Approach): The above figure shows three subspaces (lines) $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$; and and a current estimate of the L_q minimum $\mathbf{X}^t$. In the gradient descent approach an updated point $\mathbf{S}^{t+1}$ is computed by keeping the projections $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$ fixed.*

$\mathbf{X}^t$ of the L_q minimum is updated using an update function W defined as,

$$\mathbf{X}^{t+1} = W(\mathbf{X}^t) = \frac{\sum_{i=1}^{k} w_i^t \, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)}{\sum_{i=1}^{k} w_i^t} \,, \tag{5.9}$$

where $w_i^t = \|\mathbf{X}^t - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^{q-2}$.

In the following section we show that under the update function W the L_q cost function is non-increasing. Later, in the section we will give a proof of the convergence of the sequence of points $(\mathbf{X}^t)$ to the global minimum of the cost function $C_q(\mathbf{X})$.

5.4.1 Non-Increasing L_q Cost Function

In this section we prove that under the update function W (5.9) the value of C_q is non-increasing. Let $\widetilde{C}_2^{\mathbf{w}}(\mathbf{X})$ be a weighted least squares cost function,

$$\widetilde{C}_2^{\mathbf{w}}(\mathbf{X}) = \sum_{k=1}^{k} w_i^t \, \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^2 \,, \tag{5.10}$$

where $w_i^t = \|\mathbf{X}^t - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^{q-2}$, as defined in (5.5). Note that the update function W (5.9) finds an exact minimizer of $\widetilde{C}_2^{\mathbf{w}}(\mathbf{X})$. The following lemma proves that under the update function W the value of C_q is non-increasing.

Lemma 5.3. *For an update function W (5.9), $C_q(W(\mathbf{X})) \leq C_q(\mathbf{X})$, where equality holds only when $W(\mathbf{X}) = \mathbf{X}$.*

Proof. We will apply lemma 3.6 (p. 35) to prove that the value of C_q is non-increasing under the update function W defined in (5.5). Lemma 3.6 (p. 35) relates an L_q cost function with a weighted L_2 cost function. Moreover, it shows that for a particular choice of weights a decrease in a weighted L_2 function also results in a decrease in an L_q cost function. Note that W is an update function finds an exact minimizer of $\widetilde{C}_2^{\mathbf{w}}$. Thus, for an updated value of $\mathbf{X}$ we have

$\widetilde{C}_2^{\mathbf{w}}(W(\mathbf{X})) \leq \widetilde{C}_2^{\mathbf{w}}(\mathbf{X})$. To be able to apply lemma 3.6 (p. 35) to the L_q function C_q, and the weighted L_2 function $\widetilde{C}_2^{\mathbf{w}}$, we proceed as follows:

Let $a_i = \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|$ and $b_i = \|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|$ in lemma 3.6 (p. 35). Since W finds an exact minimum of $\widetilde{C}_2^{\mathbf{w}}$, the following relation holds

$$\sum_{i=1}^{k} \frac{\|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^2}{\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^{2-q}} = \widetilde{C}_2^{\mathbf{w}}(W(\mathbf{X})) \leq \widetilde{C}_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \,. \tag{5.11}$$

Then from lemma 3.6 (p. 35) it follows that

$$\sum_{i=1}^{k} \|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \leq \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q = C_q(\mathbf{X}) \,.$$

Since $\|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(W(\mathbf{X}))\|$ is the orthogonal distance of $W(\mathbf{X})$ from $\mathcal{S}_i$, it is always less than or equal to the distance of $W(\mathbf{X})$ from any other point on the subspace $\mathcal{S}_i$, that is $\|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|$. We then have

$$\|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(W(\mathbf{X}))\| \leq \|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\| \,,$$

for all values of i.

From the above two inequalities the desired relation follows as

$$\begin{aligned} C_q(W(\mathbf{X})) &= \sum_{i=1}^{k} \|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(W(\mathbf{X}))\|^q \\ &\leq \sum_{i=1}^{k} \|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \leq C_q(\mathbf{X}) \,. \end{aligned}$$

where equality holds only when $W(\mathbf{X}) = \mathbf{X}$. □

This completes the proof that under the update function W (5.9) the value of C_q decreases after every iteration, except when $W(\mathbf{X}) = \mathbf{X}$.

5.4.2 Algorithm

Given a set of non-parallel affine subspaces of different dimensions, $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\} \in \mathbb{R}^N$, $k \geq 2$. The following algorithm shows that the sequence of points $(\mathbf{X}^t)$ obtained using the algorithm converges to the desired L_q minimum.

Algorithm 5.1. *Our algorithm for finding the L_q minimum is as follows:*

1. *For each j find the point $\mathbf{X}_j^* = \operatorname{argmin}_{\mathbf{X} \in \mathcal{S}_j} C_q^j(\mathbf{X})$, where $C_q^j(\mathbf{X}) = \sum_{i \neq j} d(\mathbf{X}, \mathcal{S}_i)^q$. Note that $d(\mathbf{X}_j^*, \mathcal{S}_j) = 0$. (For details see section 5.4.4.)*

2. *Among the $\mathbf{X}_j^*$ select the one with minimum cost: $\mathbf{X}^* = \operatorname{argmin}_j C_q^j(\mathbf{X}_j^*)$.*

3. *Compute the gradient of C_q, omitting the subspace $\mathcal{S}_j$ containing $\mathbf{X}^*$:*

$$\nabla C_q^j(\mathbf{X}^*) = \sum_{i \neq j} \nabla d(\mathbf{X}^*, \mathcal{S}_i)^q \, .$$

4. *If $\|\nabla C_q^j(\mathbf{X}^*)\| \leq 1$, then $\mathbf{X}^*$ is the required minimum (according to lemma 5.2), and the algorithm is complete.*

5. *Otherwise, displace $\mathbf{X}^*$ in the downhill gradient direction of $\|\nabla C_q^j(\mathbf{X}^*)\|$ to obtain $\mathbf{X}^0$. Backtrack if necessary to ensure $C_q(\mathbf{X}^0) < C_q(\mathbf{X}^*)$.*

6. *Repeat, $\mathbf{X}^{t+1} = W(\mathbf{X}^t)$ until convergence, where W is defined in (5.9).*

The initial point $\mathbf{X}^0$ so found has cost less than any point on the subspaces, and iterations of the algorithm from $\mathbf{X}^0$, can not again approach any of the subspaces. Hence the algorithm is guaranteed to converge to the global minimum of $C_q(\mathbf{X})$.

Practicalities: In reality, it is unnecessary to use the initialization procedure of Algorithm 5.1, which is given only to supply a provably convergent algorithm. Instead, one may start with an arbitrary point $\mathbf{X}^0$, as in the original Weiszfeld algorithm. The likelihood of meeting one of the subspaces is small. If iterations appear to converge towards a given subspace $\mathcal{S}_j$, then one may apply the restricted algorithm to find the minimum cost point on $\mathcal{S}_j$, then determine using lemma 5.2 whether this is the minimum of the complete cost function. If not, one may continue from a point $\mathbf{X}$ with cost lower than that of the minimum on $\mathcal{S}_j$. Because of the decreasing cost, the subspace $\mathcal{S}_j$ will not be encountered a second time.

5.4.3 Proof of Convergence

In this section we give a proof that the sequence of points $(\mathbf{X}^t)$ obtained using Algorithm 5.1 converges to the L_q minimum. The algorithm has two main parts. In the first part a restricted algorithm is applied to find a minimum on each subspace. In this case iterates are restricted to a particular subspace. If the L_q minimum is on one of the subspaces, then this procedures finds that minimum and the algorithm is complete. On the other hand, if the L_q minimum is not on any of the subspaces, a repetitive application of the update function W (5.9) ensures that the algorithm converges to the L_q minimum.

Theorem 5.4. *Algorithm 5.1 converges to the L_q minimum.*

Based on Theorem 3.4 (p. 29) and Theorem 3.5 (p. 30) we will use the following outline to prove the convergence of Algorithm 5.1.

Outline 5.1. *Given an update function W and a strictly convex function C_q, to prove that the sequence $(\mathbf{X}^t)$ obtained using $\mathbf{X}^{t+1} = W(\mathbf{X}^t)$ is convergent to the minimum of C_q we proceed as follows:*

1. *The update function W is continuous, defined on a compact domain $\mathcal{D}$. The update function W is a map from $\mathcal{D}$ to itself.*

2. *The value of C_q is non-increasing at every iteration.*

3. *The set S of Theorem 3.4 (p. 29) is a finite set containing the L_q minimum point.*

4. *Since there are a finite number of accumulation points, the sequence $(\mathbf{X}^t)$ is in fact convergent (see Theorem 3.5 (p. 30)).*

Now, we will verify every step of the above outline to prove Theorem 5.4.

Proof of Theorem 5.4: The L_q minimum is categorized in two ways depending on whether the minimum lies on any subspace or not. We deal with both the cases separately. If the L_q minimum is on one of the subspaces then this point is obtained at the end of iteration 4 of Algorithm 5.1. The algorithm terminates at this step and the desired L_q minimum is retrieved. In section 5.4.4 we discuss specific details of the restricted algorithm. On the other hand, if the L_q minimum does not lie on one of the subspaces then we proceed with the iterative process of updating a current estimate of the L_q minimum by using the update function W defined in (5.9).

Here, we only give a proof for the case when the L_q minimum does not lie on any subspace. Thus, in the proof given below we assume that an initial estimate of the L_q mean is selected by using the first five steps of Algorithm 5.1. A proof of the restricted algorithm is not difficult and follows directly from the proof of unrestricted algorithm given below.

The above discussion simplifies the proof of Theorem 5.4 and we only need to verify that the cost function C_q and the update function W satisfy all the conditions of the steps of Outline 5.1. Below we verify each step of Outline 5.1 to show that the sequence of points obtained by using (5.9) converges to the L_q minimum.

1. The compact domain $\mathcal{D}$ is defined as $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^N \mid C_q(\mathbf{X}) \leq C_q(\mathbf{X}^0)\}$, where $\mathbf{X}^0$ is a starting point selected using Algorithm 5.1. The set $\mathcal{D}$ is compact, except when there exists a vector $\mathbf{V}$ that is parallel with every subspace, where the sub-level set is not bounded. Since we assume that subspaces are not parallel, therefore such a vector does not exist and the set $\mathcal{D}$ is compact. It is easy to see that the update function W (5.9) is continuous because $\lim_{\mathbf{X}\to\mathbf{C}} W(\mathbf{X}) = W(\mathbf{C})$ for all values of $\mathbf{X} \in \mathcal{D}$. Since from lemma 5.3 the cost function C_q is non-increasing under the update function W, the value of $C_q(\mathbf{X}^t)$ will never be greater than $C_q(\mathbf{X}^0)$. Thus, from the definition of $\mathcal{D}$ the update function W maps a point in $\mathcal{D}$ to itself.

2. The cost function C_q is a non-increasing function under the update function W, that is $C_q(W(\mathbf{X})) \leq C_q(\mathbf{X})$ with equality only when $W(\mathbf{X}) = \mathbf{X}$, see lemma 5.3.

3. Under the update function W the value of C_q remains constant in successive iterations only when $\mathbf{X}$ is a stationary point of C_q, where the value of the function stops changing in descent algorithms. The cost function C_q being strictly convex has a unique stationary point $\mathbf{X}^*$. Thus, the set S in Theorem 3.4 (p. 29) contains a single point, that is the stationary point of C_q.

4. Since S is a finite set, from Theorem 3.5 (p. 30) every sequence $(\mathbf{X}^t)$ is convergent to $S = \{\mathbf{X}^*\}$, that is the L_q minimum.

This completes the proof of our claim that the sequence of points $(\mathbf{X}^t)$ converges to the L_q minimum, given that an initial estimate of the L_q minimum is selected by using the first five steps of Algorithm 5.1. □

In the following section we show that the restricted Weiszfeld algorithm converges to the minimum of C_q^j, that is the L_q minimum on the j-th subspace.

5.4.4 Minimization on a Subspace

We now discuss the L_q optimization strategy restricted on a subspace. If the L_q minimum lies on a subspace then this method finds that point, otherwise it gives a good starting point for the unrestricted algorithm.

The minimization of a restricted L_q cost function takes the following form,

$$\min_{\mathbf{X}\in\mathcal{S}_j} C_q^j(\mathbf{X}) = \min_{\mathbf{X}\in\mathcal{S}_j} \sum_{i\neq j} d(\mathbf{X},\mathcal{S}_i)^q \ .$$

This is the same optimization problem as in the unrestricted algorithm, except that the minimization takes place on the affine subspace $\mathcal{S}_j$.

The minimization of a restricted L_q cost function can alternatively be done by minimizing a restricted weighted L_2 cost function. However, to ensure that the L_q cost function is also decreasing along with the weighted L_2 cost function, a similar lemma as lemma 5.3 can easily be proved.

Thus, the following derivation is for a weighted L_2 cost function restricted on a the j-th subspace.

Derivation: If the minimum point lies on the j-th subspace then the cost function to minimize is

$$\mathbf{X}^{t+1} = \operatorname*{argmin}_{\mathbf{X}\in\mathcal{S}_j} \widetilde{C}_2^j(\mathbf{X}) = \operatorname*{argmin}_{\mathbf{X}\in\mathcal{S}_j} \sum_{i\neq j} w_i^t\, d(\mathbf{X},\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t))^2 \ ,$$

subject to

$$\mathtt{E}^T\mathbf{X} + \mathbf{d} = \mathbf{0} \ ,$$

where $w_i^t = \|\mathbf{X}^t - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^{q-2}$ and $\mathtt{E}^T\mathbf{X} + \mathbf{d} = 0$ is the equation for subspace $\mathcal{S}_j$ and $\mathtt{E}$ is a $n \times r$ matrix, with n being the dimension of the ambient space and r is the codimension of subspace $\mathcal{S}_j$ and $\mathbf{d}$ is an r vector.

The Lagrangian is then

$$L(\mathbf{X},\lambda) = \sum_{i\neq j} w_i^t\, d(\mathbf{X},\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t))^2 - (\mathtt{E}^T\mathbf{X} + \mathbf{d})^T\lambda \ ,$$

where λ is an r vector. The gradient of $L(\mathbf{X},\lambda)$ with respect to $\mathbf{X}$ is

$$L_{\mathbf{X}}(\mathbf{X},\lambda) = \sum_{i\neq j} w_i^t\, \nabla\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^2 \ - \mathtt{E}\lambda \ .$$

or equivalently,

$$L_{\mathbf{X}}(\mathbf{X}, \lambda) = 2 \sum_{i \neq j} w_i^t \mathbf{X} - 2 \sum_{i \neq j} w_i^t \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t) - \mathtt{E}\lambda \ .$$

Let $w = 2 \sum_{i \neq j} w_i^t$ and $\mathbf{p} = 2 \sum_{i \neq j} w_i^t \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$, then the above equation can be written as

$$L_{\mathbf{X}}(\mathbf{X}, \lambda) = w\,\mathbf{X} - \mathbf{p} - \mathtt{E}\lambda \ . \tag{5.12}$$

Setting $L_{\mathbf{X}}(\mathbf{X}, \lambda)$ equal to zero and computing the value of $\mathbf{X}$ we get

$$\mathbf{X} = \frac{1}{w} (\mathtt{E}\,\lambda + \mathbf{p}) \ . \tag{5.13}$$

Applying the constraint $\mathtt{E}^T\mathbf{X} + \mathbf{d} = 0$, we get the value of λ as

$$\lambda = -(\mathtt{E}^T\mathtt{E})^{-1}(\mathtt{E}^T\mathbf{p} + w\,\mathbf{d}) \ .$$

By using the value of λ we can compute $\mathbf{X}^{t+1}$ using (5.13). Therefore the solution of $\widetilde{C}_2^j(\mathbf{X})$ is found by solving a set of linear equations. □

The above derivation shows a method to minimize a weighted L_2 cost function restricted on a subspace. However, we require a method that minimizes a restricted L_q cost function. For an updated value of $\mathbf{X}^{t+1}$, we have $\widetilde{C}_2^j(\mathbf{X}^{t+1}) \leq \widetilde{C}_2^j(\mathbf{X}^t)$. A similar argument as in section 5.4.1 can be presented here to show that the value of C_q^j is non-increasing after every iteration of the restricted algorithm.

Once again, we only need to verify all the steps of Outline 5.1 to prove that starting from a random point on a subspace $\mathcal{S}_j$, the sequence of points obtained using (5.13) converges to the minimum of C_q^j. The proof is very simple and is hence omitted.

5.4.5 Strict Convexity of the L_q Cost Function

In this section we prove the strict convexity of the cost function C_q. It is known that a function is strictly convex if the Hessian of that function is positive definite. In order to show that the cost function (5.1) is strictly convex we need to prove that the Hessian of the cost function is positive definite or equivalently, the sum of Hessians of the individual distance functions is positive definite.

Let us first consider a simple case in which a d-dimensional subspace $\mathcal{S}$ in $\mathbb{R}^N$, $d < n$, is aligned with the canonical axis $\mathbf{e}_{r+1}, \mathbf{e}_{r+2}, \ldots, \mathbf{e}_n$, where $r = n - d$ and $\mathbf{e}_i$ is an n-dimensional vector aligned with x_i axis. Now, the orthogonal distance of a point $\mathbf{X} = (x_1\ x_2\ \ldots\ x_n)^T$ from $\mathcal{S}$ is given by the following equation

$$d(\mathbf{X}, \mathcal{S})^q = \left(\sqrt{x_1^2 + x_2^2 + \ldots + x_r^2} \right)^q \ . \tag{5.14}$$

Let $\mathbf{X}' = (x_1\ x_2\ \ldots\ x_r)^T$, that is, the component of $\mathbf{X}$ that contributes in computing the

distance. The Hessian $\mathtt{H}$ of the distance function is an $n \times n$ matrix of the following form.

$$\mathtt{H} = c \begin{bmatrix} \mathtt{H}' & \mathtt{0}^T_{d\times r} \\ \mathtt{0}_{d\times r} & \mathtt{0}_{d\times d} \end{bmatrix},$$

where $c = (x_1^2 + x_2^2 + \ldots + x_r^2)^{\frac{q-4}{2}}$ and $\mathtt{H}'$ is an $r \times r$ matrix defined as

$$\mathtt{H}' = \begin{bmatrix} \mathbf{X'}^T\mathbf{X'} - x_1^2 & -x_1x_2 & \ldots & -x_1x_r \\ -x_1x_2 & \mathbf{X'}^T\mathbf{X'} - x_2^2 & \ldots & -x_2x_r \\ \vdots & \vdots & \ddots & \vdots \\ -x_1x_r & -x_2x_r & \ldots & \mathbf{X'}^T\mathbf{X'} - x_r^2 \end{bmatrix}.$$

We can rewrite $\mathtt{H}'$ as follows,

$$\mathtt{H}' = (\mathbf{X'}^T\mathbf{X'})\mathtt{I} - \mathbf{X'}\mathbf{X'}^T. \tag{5.15}$$

Let $\mathtt{Q}' = [\mathbf{q}_1\ \mathbf{q}_2\ \ldots\ q_{r-1}]$ be a matrix composed of the n-dimensional vectors that form the orthonormal basis of the orthogonal complement of the affine subspace formed by $\mathcal{S}$ and $\mathbf{X'}/\|\mathbf{X'}\|$. Let $\mathtt{Q} = [\frac{\mathbf{X'}}{\|\mathbf{X'}\|}\ \mathbf{q}_1\ \mathbf{q}_2\ \ldots\ \mathbf{q}_{r-1}]$ and such that $\mathtt{Q}^T\mathtt{Q} = \mathtt{Q}\mathtt{Q}^T = \mathtt{I}$. Then, $\mathbf{X'}\mathbf{X'}^T$ can be written in the following form:

$$\mathbf{X'}\mathbf{X'}^T = \mathtt{Q}\ \mathtt{T}\ \mathtt{Q}^T,$$

where $\mathtt{T}$ is an $r \times r$ matrix of the following form:

$$\mathtt{T} = \begin{bmatrix} \mathbf{X'}^T\mathbf{X'} & \mathbf{0}^T_{(r-1)\times 1} \\ \mathbf{0}_{(r-1)\times 1} & \mathtt{0}_{(r-1)\times(r-1)} \end{bmatrix}.$$

By substituting $\mathbf{X'}\mathbf{X'}^T = \mathtt{Q}\ \mathtt{T}\ \mathtt{Q}^T$ and $\mathtt{I} = \mathtt{Q}\mathtt{Q}^T$ in (5.15), we get the following form

$$\mathtt{H}' = \mathtt{Q}\ \Big((\mathbf{X'}^T\mathbf{X'})\mathtt{I}_{r\times r} - \mathtt{T} \Big)\ \mathtt{Q}^T. \tag{5.16}$$

The inner term of the above equation can be written as,

$$(\mathbf{X'}^T\mathbf{X'})\mathtt{I}_{r\times r} - \mathtt{T} = \begin{bmatrix} 0 & \mathbf{0}^T_{(r-1)\times 1} \\ \mathbf{0}_{(r-1)\times 1} & (\mathbf{X'}^T\mathbf{X'})\ \mathtt{I}_{(r-1)\times(r-1)} \end{bmatrix}.$$

Thus, the final form of $\mathtt{H}'$ (5.16) is,

$$\mathtt{H}' = (\mathbf{X'}^T\mathbf{X'})\ \mathtt{Q}'\ \mathtt{Q}'^T. \tag{5.17}$$

Equation (5.17) shows that the Hessian of the distance function depends on the orthonormal vectors that form the basis of the orthogonal complement of the affine subspace formed by $\mathcal{S}$ and $\mathbf{X'}/\|\mathbf{X'}\|$. Let $\mathbf{q}_{i1}\ \mathbf{q}_{i2}\ \ldots\ \mathbf{q}_{i(r-1)}$, $i = 1, 2, \ldots, k$ be the orthogonal basis vectors of the orthogonal complement of an affine subspace spanned by $\mathcal{S}_i$ and $\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$. Since, our

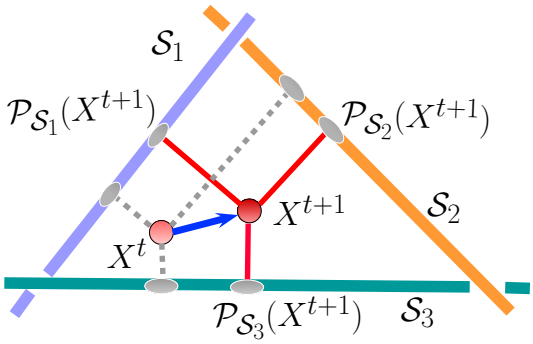


Figure 5.5: *L_q-closest-point to Subspaces (General Approach): The above figure shows three subspaces (lines) $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$; and and a current estimate of the L_q minimum $\mathbf{X}^t$. In the general approach the value of projections $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ is kept as variable and its value changes simultaneously with the value of $\mathbf{X}$ during the estimation of an update $\mathbf{X}^{t+1}$.*

cost function is the sum of individual distance functions, the Hessian is the sum of Hessians of individual distance functions. Therefore, the Hessian of the cost function depends on the basis vectors $\mathbf{q}_{ij}$ of the Hessians of each of the distance functions. If the set of basis vectors, $\mathbf{q}_{ij}$, spans the whole space $\mathbb{R}^N$ then the Hessian of the cost function is a full rank matrix and our cost function is strictly convex, that is, a unique solution exists that minimizes the cost.

5.5 L_q Solution Strategy II: General Approach

In this section we propose a slightly different method than the gradient descent approach to find the L_q-closest-point to a given set of subspaces, we refer to it as the *General approach*. Just like the gradient descent approach, the general approach is also an iterative minimization technique. Given a current estimate $\mathbf{X}^t$ of the L_q minimum, the gradient descent approach computes an update by keeping the projections of $\mathbf{X}^t$ fixed. On the other hand, in the general approach a current estimate of the L_q minimum $\mathbf{X}^t$ is updated by keeping the projection points as variables, as shown in fig. 5.5.

The L_q cost function is defined as,

$$C_q(\mathbf{X}) = \sum_{i=1}^{k} d(\mathbf{X}, \mathcal{S}_i)^q = \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \,,$$

where $\|\cdot\|$ is the L_2 norm. As described before that $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ represents the affine projection of $\mathbf{X}$ on $\mathcal{S}_i$, and can explicitly be written as $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}) = \mathbf{C}_i + \mathtt{A}_i\,(\mathbf{X} - \mathbf{C}_i)$, where $\mathtt{A}_i$ is a linear projection matrix and $\mathbf{C}_i \in \mathcal{S}_i$ is taken to be the origin of the orthonormal basis of $\mathcal{S}_i$. The distance function can be explicitly written as,

$$d(\mathbf{X}, \mathcal{S}_i) = \|\mathbf{X} - (\mathbf{C}_i + \mathtt{A}_i\,(\mathbf{X} - \mathbf{C}_i))\| = \|\mathtt{M}_i\,(\mathbf{X} - \mathbf{C}_i))\| \,,$$

where $\mathtt{M}_i = \mathtt{I} - \mathtt{A}_i$. For details see section 5.2.

In the general approach, given a current estimate of the L_q minimum $\mathbf{X}^t$, an update is

computed as,

$$\mathbf{X}^{t+1} = W(\mathbf{X}^t) = \left(\sum_{i=1}^{k} (w_i^t)^2 \, \mathtt{M}_i\right)^{-1} \left(\sum_{i=1}^{k} (w_i^t)^2 \, \mathtt{M}_i \mathbf{C}_i\right) , \tag{5.18}$$

where $w_i^t = \|\mathbf{X}^t - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^{q-2}$. Note that for $w_i^t = 1$, the above update function is the same as the L_2-closest-point solution (5.8). However, the above equation finds a closed form solution of the weighted L_2 cost function $\widehat{C}_2^{\mathbf{w}}(\mathbf{X})$ defined in (5.7).

Numerical stability. If the current estimate $\mathbf{X}^t$ lies close to one of the subspaces $\mathcal{S}_i$, then the weight w_i^t given by (5.18) can become quite large. If the iterates $\mathbf{X}^t$ converge towards one of the subspaces, then the corresponding weight becomes infinite. One can introduce some degree of stability by dividing all the weights w_i by the maximum weight $w_{\max}$ without changing the problem. Thus, each weight is replaced by $w_i / w_{\max}$.

However, if one of the weights $w_i = w_{\max}$ becomes very large, then the matrix $\mathtt{M}$ becomes poorly conditioned. The matrix in (5.18) being inverted becomes increasingly close to singular. The results will be numerically unstable, and meaningless as $\mathbf{X}^t$ approaches $\mathcal{S}_i$.

The problem here is the use of the method of normal equations to solve the least-squares problem. An alternative (superior) method is to use Singular Value Decomposition (SVD) to minimize $C_2^{\mathbf{w}} = \| \mathtt{M}\mathbf{X} - \mathbf{c} \|^2$, where $\mathtt{M}$ is formed as the stack of the matrices $w_i^t \mathtt{M}_i$, and $\mathbf{c}$ is the stack of vectors $w_i^t \mathtt{M}_i \mathbf{C}_i$. Let $\mathtt{M} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$ be the SVD of matrix $\mathtt{M}$. Then the solution $\mathbf{X}$ is given by

$$\mathbf{X} = \mathtt{V}\,\mathtt{D}^{-1}\mathtt{U}^\top \, \mathbf{c} \, . \tag{5.19}$$

When the weights w_i are of very different orders of magnitude, this SVD method works very much better than the method given by the update (5.18). For more on this topic, see Appendix 5 of Hartley and Zisserman [2004].

Algorithm: It should be noted that if one of the iterates $\mathbf{X}^t$ lands precisely on one of the subspaces $\mathcal{S}_i$, then the update rule is undefined, because the corresponding weight w_i^t is infinite if $q < 2$. Therefore, a similar algorithm as Algorithm 5.1 can be proposed for the update function defined in (5.18). A proof of convergence of that algorithm will be the same as the proof of convergence of Algorithm 5.1 in section 5.4.3, and is therefore not provided here.

An important part of the proof is to show that the value of the L_q cost function is non-increasing under the update function W (5.18). The following discussion shows that the update function W computes the minimum of $\widehat{C}_2^{\mathbf{w}}(\mathbf{X})$ (5.7), and the L_q cost function is non-increasing under the update function W.

Non-increasing L_q Cost: It is easy to show that under the update function W defined in (5.18), the L_q cost function is non-increasing. Note that the update function defined in (5.18)

minimizes a weighted L_2 cost function,

$$\widehat{C}_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} w_i^t \, \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^2 \,, \tag{5.20}$$

where $w_i^t = \|\mathbf{X}^t - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^{q-2}$, as defined before in (5.7).

For an updated point $W(\mathbf{X})$ we have,

$$\begin{aligned}\sum_{i=1}^{k} \frac{\|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(W(\mathbf{X}))\|^2}{\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^{2-q}} &= \widehat{C}_2^{\mathbf{w}}(W(\mathbf{X})) \leq \widehat{C}_2^{\mathbf{w}}(\mathbf{X}) \\ &= \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \,.\end{aligned}$$

The desired relation follows directly from lemma 3.6 (p. 35),

$$\begin{aligned}\sum_{i=1}^{k} \|W(\mathbf{X}) - \mathcal{P}_{\mathcal{S}_i}(W(\mathbf{X}))\|^q = C_q(W(\mathbf{X})) &\leq C_q(\mathbf{X}) \\ &= \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \,,\end{aligned}$$

where equality holds only when $\mathbf{X}^{t+1} = \mathbf{X}^t$. Thus, a similar lemma as lemma 5.3 can easily be proved for the update function defined in (5.18).

5.5.1 Minimization on a Subspace

The first step in this initialization algorithm is a restricted problem of finding the point with minimum cost on each subspace $\mathcal{S}_j$. This is done by applying a L_q Weiszfeld algorithm restricted to this subspace. Starting from a point on a subspace, updates are computed by restricting the solution to the subspace as,

$$\min_{\mathbf{X} \in \mathcal{S}_j} C_q^j(\mathbf{X}) = \min_{\mathbf{X} \in \mathcal{S}_j} \sum_{i \neq j} d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))^q \,.$$

Just like the Gradient descent approach, a weighted L_2 cost function restricted on a subspace can be minimized and a similar lemma as lemma 5.3 can also be presented to show that the value of L_q cost function is non-increasing under the update function computed in this section. Once again, this is a simple problem (minimizing a quadratic function on an affine subspace) and may be solved exactly by linear algebra as follows:

Derivation: If the minimum points lies on the j-th subspace then the weighted L_2 cost function to minimize is

$$\mathbf{X}^{t+1} = \operatorname*{argmin}_{\mathbf{X} \in \mathcal{S}_j} \widehat{C}_2^j(\mathbf{X}) = \operatorname*{argmin}_{\mathbf{X} \in \mathcal{S}_j} \sum_{i \neq j} w_i^t \, d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))^2 \,,$$

subject to

$$\mathtt{E}^T\mathbf{X} + \mathbf{d} = \mathbf{0}\ ,$$

where $w_i^t = \|\mathbf{X}^t - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)\|^{q-2}$ and $\mathtt{E}^T\mathbf{X} + \mathbf{d} = 0$ is the equation for subspace $\mathcal{S}_j$ and $\mathtt{E}$ is a $n \times r$ matrix, with n being the dimension of ambient space and r is the codimension of subspace $\mathcal{S}_j$ and $\mathbf{d}$ is an r vector.

The Lagrangian is then,

$$L(\mathbf{X}, \lambda) = \sum_{i \neq j} w_i^t\, d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))^2 - (\mathtt{E}^T\mathbf{X} + \mathbf{d})^T\lambda\ ,$$

where λ is an r vector. The gradient of $L(\mathbf{X}, \lambda)$ with respect to x is,

$$L_{\mathbf{X}}(\mathbf{X}, \lambda) = \sum_{i \neq j} w_i^t\, \nabla d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))^2\ - \mathtt{E}\lambda\ .$$

The gradient of the squared distance function $d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))$ is $\nabla d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))^2 = 2\mathtt{M}_i\ (\mathbf{X} - \mathbf{C}_i)$. By substituting the value of $\nabla d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}))^2$ in the above equation we get,

$$L_{\mathbf{X}}(\mathbf{X}, \lambda) = 2\ \sum_{i \neq j} w_i^t\, \mathtt{M}_i\mathbf{X} - 2\ \sum_{i \neq j} w_i^t\, \mathtt{M}_i\mathbf{C}_i - \mathtt{E}\lambda\ .$$

Let $\mathtt{P} = 2\ \sum_{i \neq j}\ w_i^t\, \mathtt{M}_i$ and $\mathbf{Q} = 2\ \sum_{i \neq j}\ w_i^t\, \mathtt{M}_i\mathbf{C}_i$, then the above equation can be written as

$$L_{\mathbf{X}}(\mathbf{X}, \lambda) = \mathtt{P}\mathbf{X} - \mathbf{Q} - \mathtt{E}\lambda\ . \tag{5.21}$$

Setting $L_{\mathbf{X}}(\mathbf{X}, \lambda)$ equal to zero and computing the value of $\mathbf{X}$ we get

$$\mathbf{X} = \mathtt{P}^{-1}(\mathtt{E}\lambda + \mathbf{Q})\ . \tag{5.22}$$

Applying the constraint $\mathtt{E}^T\mathbf{X} + \mathbf{d} = 0$, we get the value of λ as

$$\lambda = -(\mathtt{E}^T\mathtt{P}^{-1}\mathtt{E})^{-1}(\mathbf{d} + \mathtt{E}^T\mathtt{P}^{-1}\mathbf{Q})\ .$$

By using the value of λ we can compute $\mathbf{X}^{t+1}$ using (5.22). Therefore the solution of $\widehat{C}_q^j(\mathbf{X})$ is found by solving a set of linear equations. □

Just like before, the above derivation shows a method to minimize a weighted L_2 cost function restricted on a subspace. From lemma 5.3, it is not difficult to show that the value of C_q^j is non-increasing after every iteration of the restricted algorithm.

5.6 Experimental Results: Triangulation

In order to show the applicability of the proposed algorithms we solve the problem of triangulation [Hartley and Sturm, 1997; Triggs et al., 2000b]. Given two or more images of a scene, triangulation is a process of determining a point in $3D$ space from its image points, that is

the projection of the $3D$ point onto multiple images. Each image point $\mathbf{y}$ corresponds to a line in $3D$ space, passing through the center of the camera and intersecting the image plane at $\mathbf{y}$. Ideally, all the lines generated by the corresponding points in different images should intersect at a single $3D$ point and that point should be the same as the original point in $3D$ space. In practice, image points cannot be measured accurately because of various types of noise, lens distortion, interest point detection error, etc. As a result, the lines obtained from the image points are skewed form of original lines and generally do not intersect at a single point, possibly these lines may not intersect at all. The problem then is to find an optimal point of intersection of these skewed lines, that is a point for which the sum of the q-th power of distances to all the lines is the minimum. Therefore, the proposed algorithms can be used to find the L_q-closest-point to these lines.

Dataset and Starting Point of Algorithms: We apply the proposed algorithm on a well know dinosaur dataset. This dataset contains a collection of 4983 track points that are tracked over a set total of 36 images. Here we only consider the track points that are visible in more than 10 images. Thus, a minimum of 10 lines are available to perform triangulation. Since the proposed algorithms are iterative minimization techniques, they require a starting point. We take the L_2-closest-point as a starting point for the L_q algorithms. There is no specific reason for choosing the L_2-closest-point as a starting point, any random point can also be taken as a starting point.

Construction of Lines: A line is uniquely determined by two points. In our experimental setup these two points are the camera center and a back projected image point. Thus, if a camera matrix and an image point is known, a line from the center of camera and passing through the image point can easily be constructed. In the dinosaur dataset both the camera matrices and image measurements are provided. Thus, we can construct lines in $\mathbb{R}^3$ to find their optimal point of intersection.

Error Measure: The measure of accuracy for reconstructed 3D points is taken to be root mean square (RMS) of the L_1-mean of the re-projection errors, that is, the L_1-mean of the distance between reprojected points and measured image points for all the views in which that point is visible. For n reconstructed points $\mathbf{X}_j$, visible in m_j views, the RMS error is computed as follows:

$$\text{RMS error} = \sqrt{\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{e}_{\mathrm{j}}^2 / \mathrm{n}}\,,$$

where

$$e_j = \frac{\sum_{i=1}^{m_j} d(\mathbf{x}_{ij}, \mathbf{x}'_{ij})}{m_j}\,,$$

and $\mathbf{x}'_{ij}$ is the measured image point and $\mathbf{x}_{ij} = \mathtt{P}_i\mathbf{X}_j$ is the reprojected point. Note that the error reported here, that is the re-projection error, is different than the error minimized by the

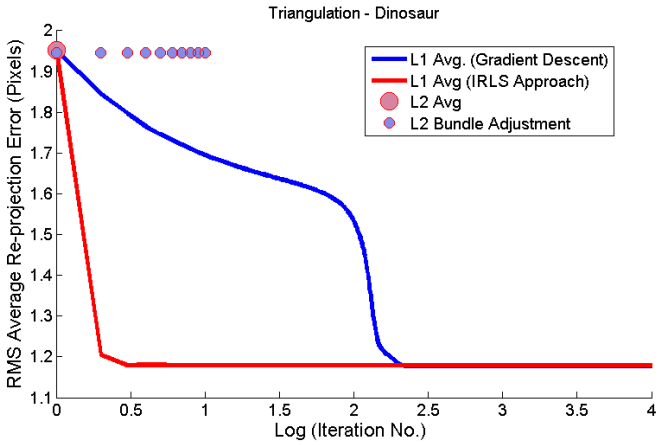


Figure 5.6: *Triangulation results for Dinosaur dataset: The above figure shows re-projection error plots for bundle adjustment, the L_2-closest-point method and the proposed L_1-averaging methods. As can be seen, the results of the L_1-averaging methods have lesser re-projection errors than both the L_2-averaging and bundle adjustment methods. Furthermore, as expected, the convergence rate of the general approach is higher than the gradient descent approach. Therefore, the general approach is recommended because of its higher convergence rate and superior results.*

.

proposed L_q algorithms, that is the q-th power of distances of a point from all of the given lines.

5.6.1 Convergence Behavior

We compare the proposed L_q optimization algorithms, that is, the gradient descent algorithm and the general algorithm, with the L_2-closest-point method and the bundle adjustment algorithm. Here, we only consider the L_q algorithms for $q = 1$. The L_2-closest-point method finds a point for which the sum of squared orthogonal distances to a set of subspaces is the minimum. This is a fairly simple problem and can be solve in closed-form. The bundle adjustment method simultaneously refines the 3D point as well as the camera parameters by minimizing the sum of squared re-projection errors, that is an error between a re-projected 3D point and its corresponding image point measurement [Triggs et al., 2000b]. Since we are only interested in recovering the 3D structure of a scene, we assume that the camera matrices are known, hence are not optimized. Bundle adjustment is carried out by using an open source sparse bundle adjustment package [Lourakis and Argyros, 2009].

A comparison of the RMS error over all iterations of the methods is reported in fig. 5.6. As can be seen, the L_1-closest-point methods have smaller errors than both the L_2-closest-point method and bundle adjustment. Furthermore, the L_2-closest-point method and bundle adjustment have roughly the same re-projection error. The main reason for smaller RMS errors for the L_1 algorithms is their robustness to outliers.

As expected, the L_1 optimization algorithm by using the general approach converges close

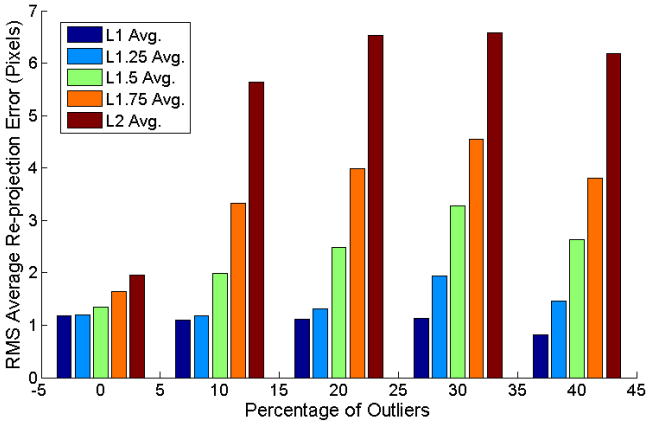


Figure 5.7: *Robustness to outliers: The above figure shows re-projection errors of the L_q-averaging method (the general approach) for several values of q ranging from* 1 *to* 2 *with an increment of* 0.25*. We test the algorithms for their robustness by adding different percentage of outliers in the dinosaur dataset, ranging from* 0% *to* 40% *with an increment of* 10%*. We modify image point correspondences to represent outliers. The above figure shows that the results of L_1 and L_q for q close to* 1 *are stable in the presence of outliers.* **Note:** *the RMS re-projection error is computed without using the modified image point correspondences, that is the outliers.*

.

to the minimum faster than the gradient descent approach. The reason for fast convergence is that the general approach updates both a current solution and its projections simultaneously. On the other hand, in the gradient descent approach projection points are held fixed during the computation of an update. This results in a slow convergence rate of the gradient descent approach, as shown in fig. 5.6. In summary, both the L_q optimization algorithms give superior results than the L_2 and the bundle adjustment methods, but the general approach a has higher convergence rate than the gradient descent approach and is therefore recommended.

5.6.2 Robustness to Outliers

In this experiment we show the robustness of the L_q method, specifically the general algorithm, for different values of q ranging from 1 to 2 with an increment of 0.25. In order to show the robustness of the method, we add different proportion of outliers to the dinosaur dataset, ranging from 0% to 40% with an increment of 10%. We modify some percentage of the image points corresponding to each 3D point to represent outliers. Furthermore, the RMS re-projection error is computed without using the modified image point correspondences, that is the outliers. Here, we only consider the general algorithm for L_q optimization because of its higher convergence rate than the gradient descent algorithm. In fig. 5.7, our experimental results show that the L_q methods are more robust to outliers than the L_2 method. Furthermore, the L_1 method has the least errors compared to the rest of the methods. On the other hand, as expected, the L_2 method is very sensitive to outliers and has larger errors than the L_q methods, for $1 \leq q < 2$. In summary, the L_1 method gives better results than the rest of the methods

and is therefore recommended in the presence of outliers.

5.7 Conclusion

In this chapter we proposed provably convergent iterative methods, based on the L_q Weiszfeld algorithm, to solve the problem of finding an L_q-closest-point to a set of affine subspaces for $1 \leq q < 2$. Moreover, our experimental results for the triangulation problem confirmed the fact that in presence of outliers in data, the minimization of an L_q cost function gives superior results than both the L_2-closest-point and bundle adjustment methods. Ease of implementation and fast iterations make the proposed algorithms attractive wherever L_q optimization is desired.

Chapter 6

Lq-Bundle Adjustment

In this chapter we propose several methods to solve for a robust solution of bundle adjustment, a non-linear parameter estimation problem. Given a set of images of a scene, bundle adjustment simultaneously estimates camera parameters and 3D structure of the scene. Generally, a least squares criterion is minimized by using the Levenberg-Marquardt (LM) method, a non-linear least squares optimization method. We are particularly interested in finding an L_q solution of the problem, for $1 \leq q < 2$. The proposed methods have an advantage of using the Levenberg-Marquardt (LM) method to find a robust solution of the problem, especially an L_q cost function. This being so, the proposed methods trivially fit in the existing literature of the least squares bundle adjustment and have potential of being used as standard methods for bundle adjustment.

In the following section we introduce the bundle adjustment problem followed by its theoretical background. In section 6.3, we formulate the problem and describe our solution strategies followed by a summary of the cost functions minimized. We then give a brief description of the existing L_2 bundle adjustment method in section 6.4. In section 6.5 and section 6.6 we propose methods to minimize robust cost functions. In the end, we present experimental results on the NotreDame set.

6.1 Introduction

We present extremely simple techniques to find a robust solution, especially an L_q solution, of the bundle adjustment problem and therefore contradict the perception that the L_q-bundle adjustment can not be done easily. An advantage of the proposed techniques is that they rely on the Levenberg-Marquardt (LM) method, a least squares minimization technique, to find a desired solution, even an L_q solution. Given a set of error vectors the LM method minimizes the sum of squared errors. We show that the minimization of a desired cost function can be achieved by using a modified difference vector or error vector, that is a vector representing the difference between measured image points and predicted image points, in the LM method.

The minimization of an L_q cost function is achieved by using two different methods. The first method, referred as the L_q *method*, minimizes the sum of the q-th power of errors by using modified error vectors in the LM method. The second method is an *Iterative Re-weighted Least Squares (IRLS)* technique where the sum of the q-th power of errors is minimized by iteratively minimizing a weighted least squares cost function, that is a weighted sum of squared errors. In addition to the L_q cost function, we propose a method to minimize the sum of the L_1 norms of

error vectors, we refer to it as the *Absolute Value* method. Note that the Absolute Value method minimizes a slightly different function than the L_q cost functions, for $q = 1$.

In addition to the L_q cost function, we propose several methods to minimize a robust cost function, specifically the *Huber function*. In robust statistics, the Huber function is a very popular differentiable function that has a hybrid behavior of a linear function and a non-linear function. The hybrid behavior of the Huber function depends on a set threshold value; the function is non-linear for input values that are smaller than the threshold value and is linear for input values that are larger than the threshold value.

A straightforward way of applying the Huber function for vector valued inputs is to apply the Huber function on each component of the input vector. However, we apply the Huber function on error vectors in the following two ways: Firstly, we apply the Huber function on the magnitude of error vectors instead of on each component of error vectors; we refer to this function as the *Isotropic Huber function*. Secondly, we introduce a re-thresholded version of the Huber function, we refer to it as the *Re-thresholded Huber* function. The Re-thresholded Huber function takes advantage of the iterative behavior of the LM method and decreases the threshold value of the Huber function after a fixed number of iterations. As a result, after some iterations the resultant function will have a dominant linear behavior and consequently increased robustness.

Given two or more images of a scene, as shown in fig. 6.1, bundle adjustment [Hartley and Zisserman, 2004; Triggs et al., 2000a; Engels et al., 2006] simultaneously estimates camera matrices and 3D points of the scene and is therefore a non-linear optimization problem. Several techniques exist in the literature to minimize a non-linear least squares problem, such as the Gauss-Newton method, the Leveberg-Marquardt method, Trust Region methods, etc. The Levenberg-Marquardt method has become a fairly standard technique to solve the bundle adjustment problem. In this chapter, we only focus on the Levenberg-Marquardt method for the minimization of desired cost functions.

In practice, image points cannot be measured accurately because of various types of noise, lens distortion, interest point detection error, etc. Consequently, image point correspondences can not be established accurately. As a result, the solution obtained using the least squares bundle adjustment method may be far from the ground truth.

Given a set of image point measurements $\mathbf{x}_{ij}$, bundle adjustment solves for camera matrices $\mathtt{P}_i$ and 3D points $\mathbf{X}_j$ such that $\mathbf{x}_{ij} = \mathtt{P}_i\mathbf{X}_j$. Due to noise in image point measurements reprojected points $\hat{\mathtt{P}}_i\hat{\mathbf{X}}_j$ are not the same as the measured image points $\mathbf{x}_{ij}$. Thus, the problem is to find an optimal estimate of the camera parameters $\hat{\mathtt{P}}_i$ and 3D points $\hat{\mathbf{X}}_j$ such that the error between the measured image points $\mathbf{x}_{ij}$ and the estimated image points $\hat{\mathtt{P}}_i\hat{\mathbf{X}}_j$ is minimized. A least squares form of the problem is

$$\min_{\hat{\mathtt{P}}_i,\hat{\mathbf{X}}_j} \sum_{i,j} d(\hat{\mathtt{P}}_i\hat{\mathbf{X}}_j, \mathbf{x}_{ij})^2 \,, \tag{6.1}$$

where $d(\mathbf{x},\mathbf{y})$ is a geometric distance between the homogeneous image points $\mathbf{x}$ and $\mathbf{y}$. The choice of minimizing the sum of squared distances makes bundle adjustment very sensitive to outliers. Instead of minimizing the L_2 cost function the minimization of an L_q cost function, for $1 \leq q < 2$, increases the robustness of bundle adjustment against outliers. The minimization

function is then,

$$\min_{\hat{\mathrm{P}}_i, \hat{\mathbf{X}}_j} \sum_{i,j} d(\hat{\mathrm{P}}_i \hat{\mathbf{X}}_j, \mathbf{x}_{ij})^q \,. \tag{6.2}$$

The above minimization problem is relatively harder than the least squares problem and is normally solved using complex optimization strategies. On the contrary, the proposed techniques use the Levenberg-Marquardt method to minimize the L_q cost function and several other robust cost functions, such as the Huber function.

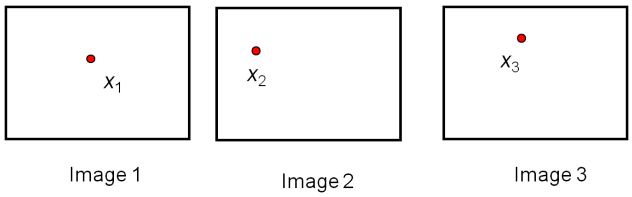
In recent years, the attention of a large group of research community has been directed to robust parameter estimation methods, especially L_1 optimization methods. A recent paper [Strelow, 2012] uses Wiberg algorithm to solve the L_1 bundle adjustment problem but it require very different formulation and does not trivially fits in the paradigm of the existing bundle adjustment literature. On the other hand, we try to minimize this trade-off between simplicity and robustness by presenting methods that use the Levenberg-Marquardt method to find a robust solution of the bundle adjustment problem, especially an L_q solution for $1 \leq q < 2$.

Given an error vector the LM method minimizes its squared norm. We take advantage of the squared norm minimization property of the LM method and modify the error vector such that the squared norm of the modified vector results in the minimization of a desired robust cost function. This modification is done in two ways: Firstly, by modifying each component of the error vector; Secondly, by introducing an attenuation factor that mitigates the effect of outliers. The squared norm of the modified error vectors can still be minimized by using the LM method. Thus, the proposed methods trivially fit in the existing literature of the bundle adjustment and only require a slight modification of the existing bundle adjustment implementations.

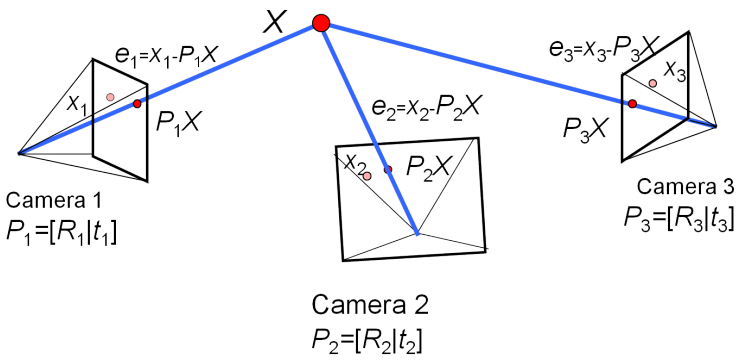
Several strategies have been proposed to make bundle adjustment a practically feasible method, such as using sparse bundle adjustment [Hartley and Zisserman, 2004; Agarwal et al., 2010], hierarchical bundle adjustment [Shum et al., 1999], incremental bundle adjustment [Steedly and Essa, 2001] that tunes parameters after every new frame arrives, spectral partitioning approach [Steedly et al., 2003] for dividing large problem into smaller sub-problems, real time bundle adjustment [Mouragnon et al., 2006], relative camera motion instead of absolute positions [Holmes et al., 2009], probabilistic approach [Eudes and Lhuillier, 2009], etc. In this chapter our focus is on increasing the robustness of bundle adjustment.

Bundle adjustment being an iterative minimization technique can not be guaranteed to converge to an optimal solution from an arbitrary starting point. Since the major contribution of this chapter is not on how to generate a good starting point for bundle adjustment, we assume that the ground truth parameters are known and use modified parameters as a starting point for the bundle adjustment algorithms.

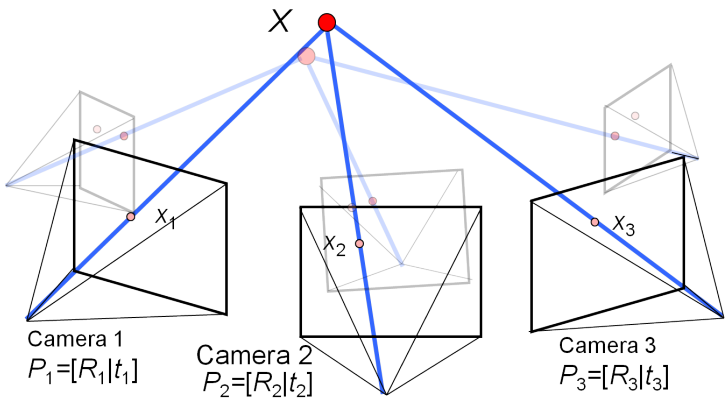
In summary, this chapter presents extremely simple and easily visualizable techniques to find a robust solution of the bundle adjustment problem, especially an L_q solution. As far as the practicalities of the proposed methods is concerned, a simple approach and re-usability of the existing least squares implementation (with minor changes) makes the proposed algorithms a perfect candidate to replace a commonly used least squares bundle adjustment technique.



(a) Image Points



(b) Starting Point



(c) Iteration 1

Figure 6.1: *Bundle Adjustment: (a), shows projections of a 3D point on several images, represented by* $\mathbf{x}_1$, $\mathbf{x}_2$ *and* $\mathbf{x}_3$. *Starting from an initial estimate of camera parameters* P_i *and 3D structure* $\mathbf{X}$, *as shown in (b), bundle adjustment simultaneously updates camera parameters and 3D structure of the scene by minimizing the re-projection error, that is* $\sum_i \|\mathbf{e}_i\|^2$, *as shown in (c).*

6.2 Theoretical Background

The Levenberg-Marquardt (LM) method [Levenberg, 1944; Marquardt, 1963] is a numerical method for solving non-linear least squares problems. The LM method, a fusion of the steepest descent and the Gauss-Newton, outperforms both the steepest descent and the Gauss-Newton methods.

Let $c(\beta)$ be a vector valued function and $g(\beta) = c(\beta)^T c(\beta)$. The LM method minimizes a squared norm of a vector valued function

$$\min_{\beta} \; c(\beta)^T c(\beta) \; . \tag{6.3}$$

Like other numerical optimization methods the Levenberg-Marquardt method also depends on finding an update direction to reduce the cost function. A current estimate of parameters β is updated in a direction δ such that $g(\beta+\delta) \leq g(\beta)$. This process is repeated until convergence. The resultant cost function $g(\cdot)$ is then,

$$g(\beta+\delta) = c(\beta+\delta)^T c(\beta+\delta) \; . \tag{6.4}$$

By using the first order approximation of c, we get

$$g(\beta+\delta) \approx \left[c(\beta) + \mathtt{J}_c\delta\right]^T \left[c(\beta) + \mathtt{J}_c\delta\right] \; , \tag{6.5}$$

where $\mathtt{J}_c$ is the Jacobian of vector $c(\beta)$. Assuming that the function is linear; by taking the derivative and equating to zero we get,

$$\mathtt{J}_c^T \mathtt{J}_c \delta = -\mathtt{J}_c^T c(\beta) \; , \tag{6.6}$$

for details see [Hartley and Zisserman, 2004]. In the *Gauss-Newton* method, the value of δ is obtained as the solution to the above *normal equations*. The assumption of linear function eliminates the need to compute the second-order derivative. The Gauss-Newton method does not guarantee convergence if the initial parameter estimates are far from the optimum value.

In the case of the *Levenberg-Marquardt* method the normal equations of the Gauss-Newton method (6.6) are replaced with the *augmented normal equations* as,

$$(\mathtt{J}_c^T \mathtt{J}_c + \lambda \mathtt{I})\delta = -\mathtt{J}_c^T c(\beta) \; , \tag{6.7}$$

where λ is known as a *damping factor* that enables the LM algorithm to behave like both the steepest descent and Gauss-Newton methods. The value of the damping factor λ, is adjusted at each iteration depending on the change in the function value. If there is a rapid change in the cost function, a smaller value of λ can be used to make the LM algorithm behave more like the Gauss-Newton algorithm. On the other hand, if there is a small or no change in the cost function then the value of λ can be increased to reduce the cost of function by moving in the gradient descent direction. Thus, the LM method has a higher convergence rate than both the Gauss-Newton and steepest descent methods. Due to its high convergence rate the LM algorithm has become a standard method for solving non-linear least squares problems.

6.3 Problem Formulation

Let $\mathbf{x}_i \in \mathbb{R}^2$ be a noisy measurement [1] of a vector $\bar{\mathbf{x}}_i$ and $\mathbf{X}$ be a vector obtained by concatenating all measurements: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$, where N is the total number of measurements. The measurement vector $\mathbf{X}$ is assumed to be related by a non-linear function f to a parameter vector β. The mapping of parameter vector $\beta \in \mathbb{R}^M$ to measurement vectors $\hat{\mathbf{X}} \in \mathbb{R}^{2N}$, that is $\hat{\mathbf{X}} = f(\beta)$. We seek a parameter vector β for which the desired cost function C has minimum value,

$$\min_{\beta} \; C(\mathbf{X}, \beta) \; . \tag{6.8}$$

Typically, the LM method minimizes the squared norm of a vector function. Therefore we can write the above cost function as inner product of two vectors as

$$\min_{\beta} \; \mathbf{E}^T \mathbf{E} \; , \tag{6.9}$$

where $\mathbf{E}$ is an error vector computed from the values of $\mathbf{X}$ and $f(\beta)$. Depending on the cost function to be minimized, there are several ways of computing the error vector $\mathbf{E}$. For example, when minimization of the least squares cost is desired then the vector $\mathbf{E}$ is taken as the difference of vectors $\mathbf{X}$ and $f(\beta)$, $\mathbf{E} = \mathbf{X} - f(\beta)$. Then from (6.9), the least squares cost function to be minimized is

$$C_2(\mathbf{X}, \beta) = \sum_i \; \|\mathbf{e}_i\|^2 \; = \sum_i \; d(\mathbf{x}_i, f_i(\beta))^2 \; , \tag{6.10}$$

where $d(\mathbf{x}_i, f_i(\beta))$ is the Euclidean distance between a measured value $\mathbf{x}_i$ and a predicted value $f_i(\beta)$; and $\mathbf{e}_i \in \mathbb{R}^2$ is computed as difference of the vectors $\mathbf{x}_i$ and $f_i(\beta)$, $\mathbf{e}_i = \mathbf{x}_i - f_i(\beta)$, while it is assumed that the measurement $\mathbf{x}_i$ is related by a function f_i to a parameter vector β. From here on we assume that $\mathbf{e}_i = \mathbf{x}_i - f_i(\beta)$.

Since the minimization of a squared vector norm $\|\mathbf{E}\|^2$ is built into most implementations of the Levenberg-Marquardt method, we need to find a way to use the Levenberg-Marquardt method to minimize some desired robust cost functions, for example an L_q cost function,

$$C_q(\mathbf{X}, \beta) = \sum_i \; \|\mathbf{e}_i\|^q = \sum_i \; d(\mathbf{x}_i, f_i(\beta))^q \; . \tag{6.11}$$

In addition to minimization of the L_q cost, the proposed solution strategies can be used to minimize several other cost functions, such as the Huber function.

6.3.1 Solution Strategies:

In this section we discuss the techniques that are used to minimize a desired robust cost function. The first technique modifies each component of the error vectors $\mathbf{e}_i$ such that the inner product of the modified error vectors result in the minimization of a desired cost function, different than the least squares cost function. However, in the second technique each error vector

[1] In general, the theory applies to measurements in spaces of any dimension.

$\mathbf{e}_i$ is multiplied by a scalar term, commonly referred as the *attenuation factor*, that mitigates the effect of outliers on the overall cost and consequently results in the minimization of a desired robust cost function. These two techniques are explained below:

6.3.1.1 Modified Error Vector

One of the techniques to achieve the minimization of a desired cost function is to replace the error vector $\mathbf{E}$ in (6.9) with a modified error vector $\mathbf{E}'$. This new vector is obtained by modifying each component of $\mathbf{E}$. This modification results in the minimization of a desired cost function different than a least squares cost function. The resultant minimization function is,

$$\min_{\beta} \ (\mathbf{E}')^T(\mathbf{E}') \ , \tag{6.12}$$

or equivalently,

$$\min_{\beta} \ \sum_i \|\mathbf{e}'_i\|^2 \ . \tag{6.13}$$

Since the minimization function is still a squared norm of the modified error vector, the resultant cost function can be minimized by the LM method. Thus, the modified cost function does not require any new optimization method.

6.3.1.2 Attenuation Factor

Another strategy to minimize a desired cost function is to multiply each vector $\mathbf{e}_i$ by an *attenuation factor* α_i, such that the inner product of the modified error vectors $\mathbf{e}'_i = \alpha_i \mathbf{e}_i$ result in minimization of a desired function $\psi(\cdot)$. The minimization function takes the following form,

$$\min_{\beta} \ \sum_i \|\mathbf{e}'_i\|^2 = \min_{\beta} \ \sum_i \alpha_i^2 \|\mathbf{e}_i\|^2 \ , \tag{6.14}$$

or equivalently,

$$\min_{\beta} \ \sum_i \psi(\|\mathbf{e}_i\|) \ . \tag{6.15}$$

The value of the attenuation factor can be computed from the following equality

$$\|\mathbf{e}'_i\|^2 = \psi(\|\mathbf{e}_i\|) = \alpha_i^2 \|\mathbf{e}_i\|^2 \ , \tag{6.16}$$

as

$$\alpha_i = \psi(\|\mathbf{e}_i\|)^{1/2} / \|\mathbf{e}_i\| \ . \tag{6.17}$$

In this case, the value of the attenuation factor determines the impact of the corresponding error vector on the resultant cost function. For example, error vectors corresponding to large values of the attenuation factor will have more influence on the resultant function than error vectors corresponding to small values of the attenuation factor. Thus, the resultant function

can be made robust to outliers by increasing the impact of inliers and decreasing the impact of outliers.

Same as before, the proposed modification does not require any new optimization strategy because (6.15) is a result of the inner product of the modified error vector $\mathbf{E}'$, that can be minimized by using the LM method.

6.3.2 Cost Functions

Now, we summarize the existing and proposed cost functions that are used to solve bundle adjustment. Let $\mathbf{e}_i$ be the i-th error vector and e_i^j be the j-th component of $\mathbf{e}_i$. We particularly consider the following cost functions:

- **Squared Error Function:** minimizes the sum of squared errors

$$C_{SE} = \sum_i \|\mathbf{e}_i\|^2 .$$

- L_q **cost function:** minimizes the sum of errors

$$C_{Lq} = \sum_i \|\mathbf{e}_i\|^q .$$

- L_q **cost function using IRLS technique:** uses an iterative re-weighted least squares technique to find an L_q solution

$$C_{IRLS} = \sum_i \|\mathbf{e}_i\|^q .$$

- **Absolute Value function:** minimizes the sum of the L_1 norm of error vectors,

$$C_{AV} = \sum_i \sum_j |e_i^j| .$$

- **Isotropic Huber function:** For some scalar input, depending on a set threshold value, the Huber function shows a hybrid behavior of a linear function and a non-linear function. Instead of applying the Huber function independently on each component of error vectors, in this case the Huber function is applied on the magnitudes of error vectors

$$C_{IsoH} = \sum_i h(\|\mathbf{e}_i\|)^2 ,$$

  where $h(\cdot)$ is the Huber function. For details see section 6.6.1.

- **Re-thresholded Huber function:** Since, the hybrid behavior of the Huber function depends on a set threshold value, we take advantage of the iterative minimization technique and decrease the threshold value after some fixed number of iterations. As a result, the Huber function will have a dominant linear behavior and increased robustness after few iterations.

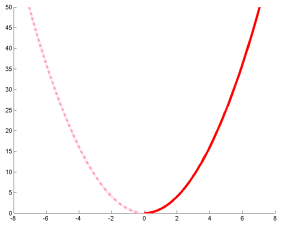


Figure 6.2: *Squared Error function (1D plot): The above figure shows a plot of $\psi_{SE}(\epsilon)$ for a scalar input ϵ. The quadratic behavior of Squared Error function indicates that the impact of values far from the origin, that is outliers, is higher than the impact of values closer to the origin, that is inliers.*

6.4 Existing Techniques

In this section we discuss some of the existing techniques to solve the bundle adjustment problem. We are particularly interested in the Squared Error function and the Huber function. The Squared Error function is a differentiable function and is popular because of the simplicity of the solution. On the other hand, the Huber function, a hybrid of a non-linear function and a linear function, is a more robust function than the least squares function and is commonly used because it is a convex function and is differentiable at the minimum. This section briefly describes the Squared Error function. However, the Huber function is discussed in section 6.6.

In the rest of the chapter we assume that the vector $\mathbf{E}$ is formed by concatenation of error vectors $\mathbf{e}_i$, where $\mathbf{e}_i$ is a vector representing the difference between a measured value $\mathbf{x}_i$ and a predicted value $f_i(\beta)$ and is computed as $\mathbf{e}_i = \mathbf{x}_i - f_i(\beta)$. Furthermore, e_i^j represents the j-th component of the i-th error vector $\mathbf{e}_i$ and ϵ simply represents a scalar value.

6.4.1 L_2 or Squared Error Function (SE):

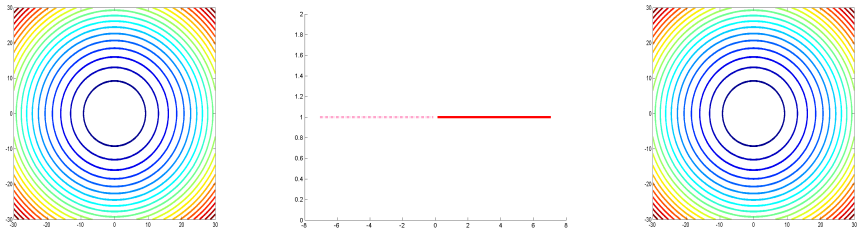
The Minimization of a squared distance function is a fairly standard and widely used technique. For some scalar input ϵ, the *Squared Error (SE)* function ψ_{SE} is,

$$\psi_{SE}(\epsilon) = \epsilon^2 \,. \tag{6.18}$$

A plot of $\psi_{SE}(\epsilon)$ for different values of ϵ is shown in fig. 6.2. An advantage of using the Squared Error function is that it is a continuous function and is at least twice differentiable; but unfortunately it suffers from a major drawback when it comes to robustness against outliers. Typically, an outlier is a point that is distant from the actual observation. Thus, has a large error value. Since the Squared Error function has a quadratic behavior, the impact of large input values, that is outliers, is magnified on the overall cost function.

For some vector $\mathbf{e}_i$, the Squared Error function ψ_{SE} is

$$\psi_{SE}(\|\mathbf{e}_i\|) = \|\mathbf{e}_i\|^2 \,. \tag{6.19}$$



(a) SE Contours (b) SE Attenuation (c) Resultant SE Contours

Figure 6.3: *Squared Error Function (2D): (a) shows an isotropic contour plot of* $\mathbf{e'}_i^T \mathbf{e'}_i / \alpha_i^2$ *for two dimensional error vectors. A plot of the constant attenuation factor* α_i *is shown in (b). Isotropic contours and a constant attenuation factor indicates that both inlier and outlier data is treated equally, therefore the Squared Error function is not robust against outliers and the resultant function also has isotropic contours, as shown in (c).*

The value of α_i from (6.17) is computed as

$$\alpha_i = \frac{\psi_{SE}(\|\mathbf{e}_i\|)^{1/2}}{\|\mathbf{e}_i\|} = \frac{\|\mathbf{e}_i\|}{\|\mathbf{e}_i\|} = 1 \,. \tag{6.20}$$

By replacing $\mathbf{e}_i$ with $\mathbf{e'}_i = \alpha_i \mathbf{e}_i$ in (6.9), the resultant cost function C_{SE}, is

$$C_2(\mathbf{X}, \beta) = \sum_i \mathbf{e'}_i^T \mathbf{e'}_i = \sum_i \mathbf{e}_i^T \mathbf{e}_i = C_{SE}(\beta) \,. \tag{6.21}$$

It can easily be seen that C_2 minimizes the sum of the squared Euclidean distances, that is C_{SE}. Since the Squared Error function is a quadratic function, the effect of outliers on the overall cost function is magnified; as a consequence the resultant cost function is very sensitive to outliers. For 2D vectors $\mathbf{e}_i$, the quadratic behavior of the Squared Error function is evident from the isotropic contours of $\mathbf{e'}_i^T \mathbf{e'} / \alpha_i^2$, as shown in fig. 6.3(a). A constant plot of the attenuation factor in fig. 6.3(b) shows that the Squared Error function assigns equal weights to all the error vectors and does not distinguish between the inlier and outlier data. A constant attenuation factor indicates that the resultant function $\mathbf{e'}_i^T \mathbf{e'}$ is still a quadratic function and has isotropic contours, as shown in fig. 6.3(c).

Since the Squared Error function has a quadratic behavior and the attenuation factor does not distinguish between the outlier and inlier data, the resultant function C_{SE} is not robust enough to outliers. Thus, even a single outlier of higher magnitude can deviate the final solution from its true value.

In the following sections we will show that the minimization of a desired cost function can be achieved either by a careful selection of the attenuation factor or by modifying each component of error vectors in a specific way; and that the resultant cost function can be minimized by using the LM method.

6.5 Proposed L_q Cost Functions

This section presents three ways of minimizing an L_q cost function. We are particularly interested in the following two forms of the L_q cost functions: Firstly, an L_q cost function that minimizes the sum of the q-th power of error, that is

$$\min_{\beta} \sum_{i} \|\mathbf{e}_i\|^q \,.$$

Secondly, an L_1 cost function that minimizes the sum of the L_1 norms of error vectors, as

$$\min_{\beta} \sum_{i,j} e_i^j \,.$$

In order to distinguish both the functions we refer to the first function as the L_q *cost function* and the second function as the *Absolute Value function*. The solution of the L_q cost function is found by using two methods: Firstly, by introducing an attenuation with each of the error vectors; Secondly, by using an Iterative Re-weighted Least Squares technique. On the other hand, the solution of the Absolute Value function is obtained by modifying each component of the error vectors. Below we discuss these techniques in detail.

6.5.1 L_q Cost Function:

The L_q cost function finds the minimum of,

$$C_{Lq} = \sum_{i} \|\mathbf{e}_i\|^q \,,$$

by using the modified error vectors approach discussed before. The minimization of the desired function is achieved by associating an attenuation factor α_i with each corresponding error vector $\mathbf{e}_i$.

For some scalar input ϵ, the L_q function ψ_{Lq} is defined as

$$\psi_{Lq}(\epsilon) = |\epsilon|^q \,. \tag{6.22}$$

Unlike the Squared Error function, where the function has quadratic behavior, the L_q cost function, for $q = 1$, is a linear function. For large values of input the L_q function has a relatively smaller output than the Squared Error function. As a result, the impact of outliers on the final solution is reduced and robustness is increased. Fig. 6.4 shows several plots of ψ_{Lq} against ϵ for different values of q. Note that, for values of ϵ far from the origin, that is outliers, the output of the L_1 cost function is the minimum compared to the rest of the L_q cost functions, for $1 < q \leq 2$. Therefore, the L_1 cost function is more robust to outliers than the rest of the L_q cost functions.

We show that error vectors in the LM method can be modified in a way that their squared norm results in the minimization of the sum of the q-th power of errors, that is $C_{Lq}(\mathbf{X}, \beta)$,

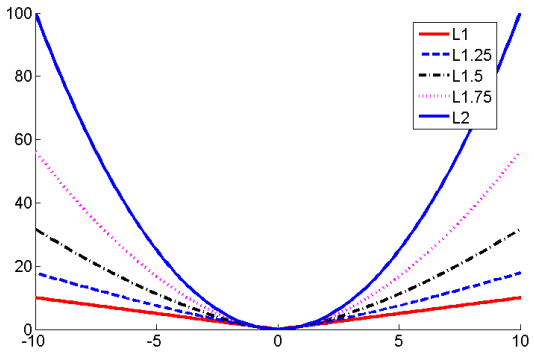


Figure 6.4: *L_q Cost Function (1D plots): The above figure shows plots of $\psi_{Lq}(\epsilon)$ against ϵ for different values of q ranging from 1 to 2. Note that the output of the L_1 cost function is the least compared to the rest of the L_q cost functions, for $1 < q \leq 2$. Therefore, in the case of the L_1 cost function the influence of values far from the origin, that is outliers, is not as magnified as in the case of other L_q cost functions.*

instead of $C_2(\mathbf{X}, \beta)$. For some vector $\mathbf{e}_i$, the L_q function ψ_{Lq} is

$$\psi_{Lq}(\|\mathbf{e}_i\|) = \|\mathbf{e}_i\|^q \,. \tag{6.23}$$

The value of α_i from (6.17) is computed as

$$\alpha_i = \frac{\psi_{Lq}(\|\mathbf{e}_i\|)^{1/2}}{\|\mathbf{e}_i\|} = \frac{\|\mathbf{e}_i\|^{q/2}}{\|\mathbf{e}_i\|} = \|\mathbf{e}_i\|^{(q-2)/2} \,. \tag{6.24}$$

By replacing the value of $\mathbf{e}_i$ with $\mathbf{e'}_i = \alpha_i \mathbf{e}_i$ in (6.9), we get

$$\begin{aligned} C_2(\mathbf{X}, \beta) &= \sum_i \mathbf{e'}_i^T \mathbf{e'}_i = \sum_i \alpha_i^2 \mathbf{e}_i^T \mathbf{e}_i \\ &= \sum_i \|\mathbf{e}_i\|^{q-2} \|\mathbf{e}_i\|^2 = C_{Lq}(\mathbf{X}, \beta) \,. \end{aligned} \tag{6.25}$$

Thus, the proposed modification results in minimization of $C_{Lq}(\mathbf{X}, \beta)$ instead of $C_2(\mathbf{X}, \beta)$. This enables us to use the Levenberg-Marquardt method to minimize the L_q cost, that would otherwise require relatively complicated strategies to find a solution. Although, the proposed algorithm finds an L_q solution for $1 \leq q < 2$, but here we are particularly interested in $q = 1$ where the L_q cost function has a linear behavior and is more robust to outliers than the L_q cost function for the rest of the values of q.

L_1 case: Here we assume that the value of q is 1, however, a similar analysis for other values of q can easily be done. Fig. 6.5(a) shows isotropic contours of $\mathbf{e'}_i^T \mathbf{e'}_i / \alpha_i^2$, that is errors without the attenuation factor. A bell shaped curve of the attenuation factor, in fig. 6.5(b), shows that that the values close to the origin, that is inliers, have higher attenuation factor α_i than the values far from the origin, that is outliers. A higher value of attenuation factor indicates that the

effect of the particular entry on the overall cost function is magnified, whereas a lower value of attenuation factor indicates that the effect of the corresponding entry is reduced. Since, the attenuation factor gives more weight to inliers and less weight to outliers, the proposed L_1 cost function is more robust to outliers than the Squared Error function. The isotropic contours together with the bell shaped curve of the attenuation factor enables the resultant function $\mathbf{e'}_i^T \mathbf{e'}_i$ to behave like a linear function and to mitigate the effect of outliers, as shown in fig. 6.5(c).

6.5.2 L_q Optimization using Iterative Re-Weighted Least Square (IRLS):

The technique proposed in this section uses an *Iteratively Re-Weighted Least Squares (IRLS)* technique to solve for an L_q solution of the problem. The IRLS technique iteratively solves a weighted least squares cost to minimize a relatively robust cost function than the standard least squares cost. The weighting factor in this case has the same role as the attenuation factor in the case of the L_q cost function, that is to increase or decrease the influence of a particular error vector on the resultant function. The robustness of an IRLS cost function depends on the weight values used.

A general form of the IRLS objective function is

$$\min_{\beta} \sum_i w_i \left\| \mathbf{x}_i - f_i(\beta) \right\|^2 , \tag{6.26}$$

where w_i is a scalar value associated with its corresponding error vector $\mathbf{e}_i$. Since this is an iterative minimization method, weights can be updated at each iteration. One can use the parameter values recovered in the previous iteration to update the weights. There can be several choices of how to update the weight values at each iteration but if we choose weights as $\|\mathbf{e}_i\|^{q-2}$, then the IRLS technique results in minimization of the L_q cost function, for details see [Aftab et al.]. If we choose different weights then the resultant function may be more robust to outliers than the Squared Error function or even the L_q function, but in that case the resultant function will no longer be the L_q cost function.

At iteration t, a current estimate of parameters β^t is updated as,

$$\beta^{t+1} = \underset{\beta}{\operatorname{argmin}} \sum_i w_i^t \|\mathbf{x}_i - f_i(\beta)\|^2 , \tag{6.27}$$

where w_i^t is some positive weighting term and is computed as $w_i^t = \|\mathbf{x}_i - f_i(\beta^t)\|^{q-2}$.

Since the Levenberg-Marquardt method is a numerical optimization algorithm, a modified solution of (6.9) at iteration t is computed as

$$C_{IRLS}(\mathbf{X}, \beta) = \sum_i w_i^t \left\| \mathbf{e}_i^t \right\|^2 , \tag{6.28}$$

where $\mathbf{e}_i^t = \mathbf{x}_i - f_i(\beta^t)$ and

$$w_i^t = \left\| \mathbf{e}_i^t \right\|^{q-2} . \tag{6.29}$$

In this case, it is obvious that C_{IRLS} is equivalent to C_q. However, in the case of analytical

optimization techniques if we use $\mathbf{e}_i$ instead of $\mathbf{e}_i^t$ in (6.28), the IRLS technique still results in the minimization of the L_q cost function [Aftab et al.].

L_1 **case:** Once again, we perform our analysis for $q = 1$. Fig. 6.5(d) shows isotropic contours of $\|\mathbf{e}_i^t\|^2$, without the influence of the weighting term, for two dimensional vector inputs. A plot of the weighting term is shown in fig. 6.5(e). Note that the weighting term has the same behavior as the attenuation factor in the case of the L_q cost function, that is to assign lower weights to outliers and higher weights to inliers. This makes the resultant cost function more robust to outliers than the Squared Error function, as shown in fig. 6.5(f).

6.5.3 Absolute Value Function (AV):

The Absolute Value function minimizes the sum of the L_1 norms of error vectors. The desired minimization function takes the following form:

$$\min_{\beta} \sum_i \|e_i\|_1 = \min_{\beta} \sum_{i,j} e_i^j \,,$$

where $\|\cdot\|_1$ is the L_1 norm and e_i^j is the j-th component of the i-th vector. Unlike the L_q cost function discussed before, the minimization of the Absolute Value function requires modification of each component of the error vector $\mathbf{e}_i$, independently. This is done by using the technique discussed in section 6.3.1.

For some scalar input ϵ the Absolute Value function f_{AV} is defined as

$$f_{AV}(\epsilon) = \begin{cases} \sqrt{\epsilon} & \text{if } \epsilon \geq 0 \\ -\sqrt{-\epsilon} & \text{otherwise} \end{cases} \,. \tag{6.30}$$

Since, we are only interested in the squared value of functions, the squared Absolute Value function ψ_{AV} for some scalar input ϵ is,

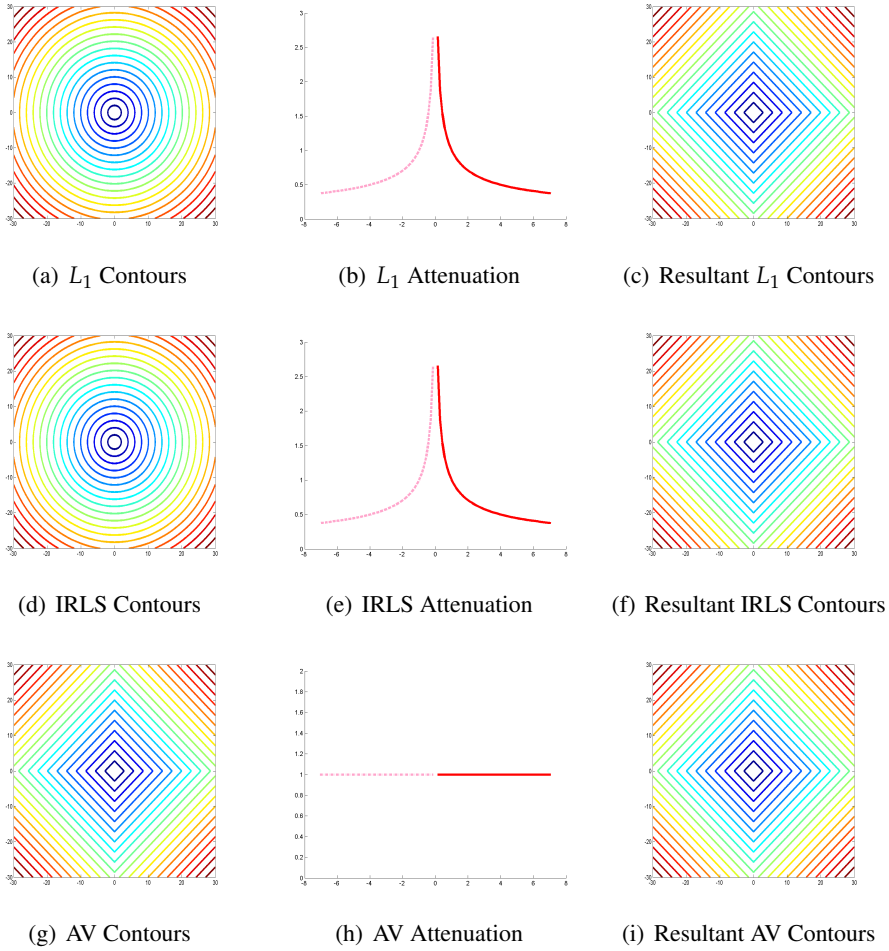
$$\psi_{AV}(\epsilon) = f_{AV}(\epsilon)^2 = |\epsilon| \,. \tag{6.31}$$

The Absolute Value function is a slightly different function than the L_q cost function, for $q = 1$, proposed earlier. However, for one-dimensional data both the functions are the same, as shown in fig. 6.4.

Let $\mathbf{e}'$ be the modified error vector obtained by applying the Absolute Value function f_{AV} on each component of an error vector $\mathbf{e}$. This new vector is computed as,

$$\mathbf{e}' = \left(f_{AV}(e^1), f_{AV}(e^2), \ldots, f_{AV}(e^n)\right)^T \,, \tag{6.32}$$

where e^j is the j-th component of $\mathbf{e}$.



(a) L_1 Contours (b) L_1 Attenuation (c) Resultant L_1 Contours

(d) IRLS Contours (e) IRLS Attenuation (f) Resultant IRLS Contours

(g) AV Contours (h) AV Attenuation (i) Resultant AV Contours

Figure 6.5: *Proposed L_q Functions (2D): The above figure shows plots of the L_1, IRLS and Absolute Value functions. The left column of the above figure shows contour plots of $\mathbf{e'}_i^T \mathbf{e'}_i / \alpha_i^2$ for two dimensional error vectors. The middle column shows plots of attenuation factors α_i. The right column plots $\mathbf{e'}_i^T \mathbf{e'}_i$, that is contour plots obtained by multiplying the left and middle columns. In the case of the L_1 and IRLS functions, robustness against outliers is achieved through the attenuation factor. The bell shaped curve of the attenuation factor indicates that the effect of inliers is magnified and the effect of outliers is reduced on the resultant function. However, in case of the Absolute Value function, non-isotropic contours indicates that the robustness against outliers is encoded in the function itself and is not achieved through the attenuation factor.*

By replacing $\mathbf{e}_i$ with $\mathbf{e'}_i$ in (6.9), we get

$$\begin{aligned} C_2(\mathbf{X},\beta) &= \sum_i \mathbf{e'}_i^T \mathbf{e'}_i \\ &= \sum_i \sum_j f_{AV}(e_i^j) f_{AV}(e_i^j) \\ &= \sum_{i,j} |e_i^j| = C_{AV}(\mathbf{X},\beta)\ . \end{aligned} \tag{6.33}$$

Since the above equation minimizes a squared error function, the Absolute Value function can be minimized by using the LM method.

Fig. 6.5(g) shows a contour plot of $\mathbf{e'}_i^T \mathbf{e'}_i$ for two dimensional input vectors. Since the attenuation factor does not play any role in this case; we take the attenuation factor as a constant of value 1, as shown in fig. 6.5(h). The non-isotropic contours of the Absolute Value function show that unlike the previously proposed techniques where attenuation factor played a major role in achieving robustness against outliers, in this case the robustness against outliers is encoded in the function itself.

6.6 Proposed Huber Cost Functions

In this section we show that the LM method can be used to minimize the Huber function. The reason for choosing the Huber function is that it is a differentiable function that preserves the convexity and is more robust to outliers than the Squared Error function. For some scalar input, depending on a set threshold value, the Huber function shows a hybrid behavior of a linear function and a non-linear function. We propose two ways of using the Huber function to solve the bundle adjustment problem. The first method, referred as the *Isotropic Huber function*, applies the Huber function for vector valued inputs. The second method, referred as the *re-thresholded Huber function*, takes advantage of the iterative behavior of the LM method and changes, specifically decrease, the threshold value of the Huber function after some iterations to increase the robustness of the resultant cost function.

The rest of the section describes the Huber function and a standard way of applying the Huber function to vector inputs, followed by the proposed Isotropic Huber function and the re-thresholded Huber function.

6.6.1 Huber Function:

In robust statistics, a very popular function to minimize is the *Huber* function [Huber, 1964]. Given a threshold value b, the Huber function $h(\cdot)$ for a scalar input ϵ is,

$$h(\epsilon) = \begin{cases} \epsilon & \text{if } |\epsilon| \leq b \\ \sqrt{(2\epsilon - b)b} & \text{if } \epsilon > b \\ -\sqrt{-(2\epsilon + b)b} & \text{if } \epsilon < -b \end{cases}\ . \tag{6.34}$$

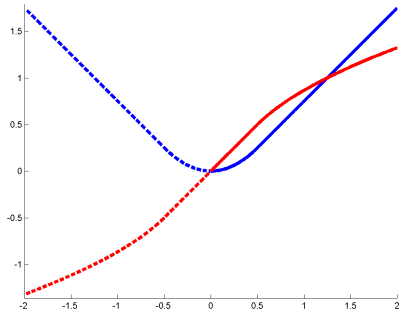


Figure 6.6: *Huber function (1D): The above figure shows plots of the Huber function, $h(\epsilon)$, represented by a red line, and the squared Huber function, $\psi_H(\epsilon)$, represented by a blue line. Note that the blue curve starts as a quadratic curve and its behavior changes from quadratic to linear after some threshold value. However, the red line shows that the function $\psi_H(\cdot)$ is a differentiable function and gradient descent algorithm will converge nicely without getting stuck at a point where function has same value for different parameter values.*

A plot of $h(\cdot)$ for different values of input is shown by the red line in fig. 6.6. Since, we are only interested in a squared function, the squared Huber function $\psi_H(\cdot)$ is,

$$\psi_H(\epsilon) = \begin{cases} \epsilon^2 & \text{if } |\epsilon| < b \\ 2b|\epsilon| - b^2 & \text{otherwise} \end{cases} . \tag{6.35}$$

A plot of the squared Huber function is shown by the blue line in fig. 6.6.

The Huber function shows a hybrid behavior depending on the value of the threshold b; for inliers, that is $|\epsilon| < b$, the Huber function acts as a quadratic function, whereas for outliers, that is $|\epsilon| \geq b$, the Huber function has a linear behavior. The selection of an appropriate threshold value is very important in this case because the dominant behavior of the Huber function is determined by the set threshold value. For smaller threshold values the Huber function behaves more like a linear function, whereas for higher threshold values it has a dominant quadratic behavior. It is evident from the blue line in fig. 6.6 that the the Huber function has a quadratic behavior below the threshold value, that is close to the origin, and a linear behavior far from the origin.

Furthermore, the threshold value also plays an important role in making the resultant cost function robust to outliers. For example, if the threshold value is too low then the dominant behavior of the Huber function is linear and the cost function is more robust to outliers than the L_2 cost function. On the other hand, if the threshold value is very high then the dominant behavior is quadratic and the resultant function is no better than the L_2 cost function in terms of robustness against outliers. Thus, a careful selection of the threshold value is very important in this case.

Another advantage of using the Huber function is that the cost function is differentiable, unlike the L_1 cost. The Huber function minimizes the trade-off between robustness and differentiability without compromising the convexity of the function. A major problem with several other robust functions is that they achieve robustness at the cost of convexity of the function.

6.6.2 Standard Huber Function

A standard way of applying the Huber function on some vector valued input $\mathbf{e}_i$, is to apply the Huber function $h(\cdot)$ on each component e_i^j of the vector as,

$$\mathbf{e'}_i = \Big(h(e_i^1), h(e_i^2), \ldots, h(e_i^n)\Big)^T \ . \tag{6.36}$$

By replacing the value of $\mathbf{e}_i$ with $\mathbf{e'}_i$ in (6.9) we get

$$C_2(\mathbf{X}, \beta) = \sum_i \mathbf{e'}_i^T \mathbf{e'}_i = \sum_{i,j} \psi_H(e_i^j) = C_H(\mathbf{X}, \beta) \ . \tag{6.37}$$

Thus, the minimization of C_2 results in the minimization of a hybrid function that adapts the minimization strategy based on each component of error vectors.

A contour plot of $\mathbf{e'}_i^T \mathbf{e'}_i$ for a two dimensional input vector is shown in fig. 6.7(a). The non-isotropic contours show that the resultant function adapts its behavior depending on the value of individual components of the error vector, that is smaller values of e_i^j are treated with the quadratic function whereas bigger values of e_i^j are treated with the linear function. Thus, instead of treating a whole point $\mathbf{e}_i$ as an outlier or inlier, the influence of the higher components of the error vector is mitigated to increase the robustness of the cost function to outliers. Since each component of error vectors is modified independently, we take the attenuation factor for all error vectors as a constant of value 1, as shown in fig. 6.7(b). Thus, the resultant function has non-isotropic contours, as shown in fig. 6.7(c).

6.6.3 Proposed Isotropic Huber (IsoH) Function

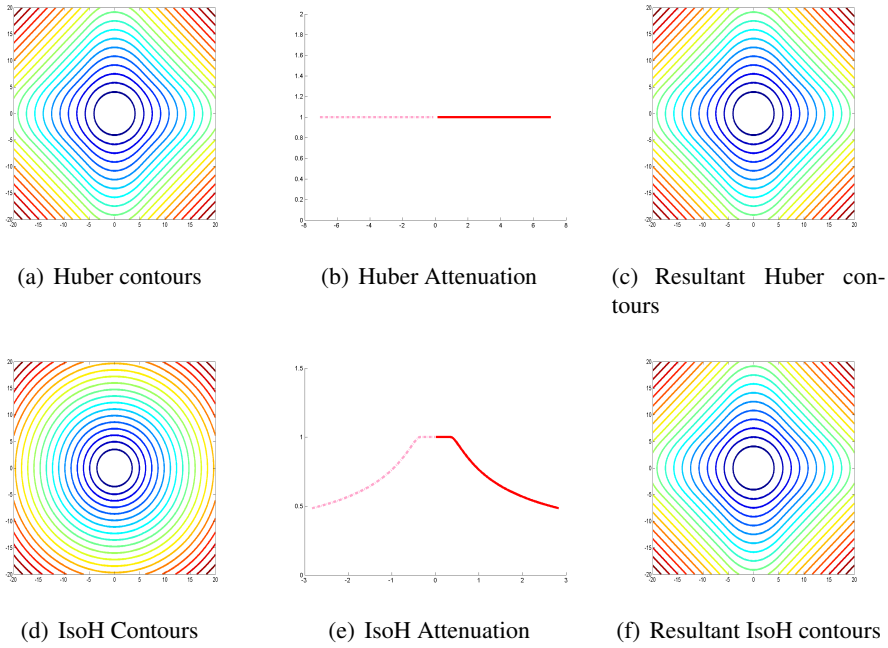
In this section we propose a slightly different application strategy of the Huber function for vector valued inputs, we refer to it as *Isotropic Huber* function because of its isotropic contours. Instead of changing the behavior of the Huber function on the basis of each component of the error vectors as mentioned before, the Isotropic Huber function adapts its behavior on the basis of the magnitudes of the error vectors, that is $\|\mathbf{e}_i\|$. This is done by introducing an attenuation factor with each error vector. Unlike the Huber function, the Isotropic Huber function treats the whole vector $\mathbf{e}_i$ as an outlier or inlier depending on the value of its magnitude. As a result, robustness against outliers is achieved instead of robustness against the high value components of error vectors.

For some input vector $\mathbf{e}_i$, the Isotropic Huber function $\psi_{IsoH}(\cdot)$, is

$$\psi_{IsoH}(\|\mathbf{e}_i\|) = \psi_H(\|\mathbf{e}_i\|) \ . \tag{6.38}$$

The value of α_i from (6.17) is computed as,

$$\alpha_i^2 \|\mathbf{e}_i\|^2 = \psi_H(\|\mathbf{e}_i\|) \ , \tag{6.39}$$



(a) Huber contours (b) Huber Attenuation (c) Resultant Huber contours

(d) IsoH Contours (e) IsoH Attenuation (f) Resultant IsoH contours

Figure 6.7: *Huber Functions (2D): The above figure shows plots of the Huber and Isotropic Huber functions. The left column of the above figure shows contour plots of* $\mathbf{e'}_i^T \mathbf{e'}_i / \alpha_i^2$ *for two dimensional error vectors. The middle column shows plots of attenuation factors* α_i*. The right column plots* $\mathbf{e'}_i^T \mathbf{e'}_i$*, that is contour plots obtained by multiplying the left and middle columns. The non-isotropic contours and a constant attenuation factor of the Huber function indicates that the robustness to outliers is encoded in the function itself and is achieved through the attenuation factor. On the other hand, the isotropic contours and a bell shaped curve of attenuation factor of the Isotropic Huber function indicates that the robustness against outlier is achieved through the attenuation factor.*

or equivalently,

$$\alpha_i = \frac{\psi_H(\|\mathbf{e}_i\|)^{1/2}}{\|\mathbf{e}_i\|} \,. \tag{6.40}$$

Finally, a modified error vector is computed as $\mathbf{e'}_i = \alpha_i \mathbf{e}_i$. By substituting the value of $\mathbf{e'}_i$ in (6.9) we get

$$\begin{aligned} C_2(\mathbf{X}, \beta) = \sum_i \mathbf{e'}_i^T \mathbf{e'}_i &= \sum_i \alpha_i^2 \mathbf{e}_i^T \mathbf{e}_i \\ &= \sum_i \psi_H(\|\mathbf{e}_i\|) = C_{IsoH}(\mathbf{X}, \beta) \,. \end{aligned} \tag{6.41}$$

The Isotropic Huber function C_{IsoH} categorizes the data on the basis of the magnitude of the input vector. Therefore, the influence of the whole data point on the resultant function is reduced or magnified regardless of the values of the individual components as in the case of the

Huber function described before. Thus, the resultant cost function C_{IsoH}, deals with outliers in a more intuitive way than the Huber cost function C_H. Once again the robustness of the proposed algorithm depends on the threshold value. So, by changing the threshold value of the Huber function the robustness of the algorithm can be increase or decreased.

A contour plot of $\mathbf{e'}_i^T \mathbf{e'}_i / \alpha_i$ for two dimensional input vectors is shown in fig. 6.7(d). The isotropic contours indicate that all the inputs are treated equally without any distinction of being an inlier or outlier. However, the bell shaped curve of the attenuation factor in fig. 6.7(e) shows that the robustness to outliers is achieved through an attenuation factor associated with each input vector. The bell shaped attenuation curve together with the isotropic function makes the resultant function $\mathbf{e'}_i^T \mathbf{e'}_i$ robust to outliers by assigning less weight to outliers, as shown in fig. 6.7(f).

6.6.4 Re-Thresholded Huber Function:

The Huber function is a hybrid of a linear function and a quadratic function, where the hybrid behavior is controlled by a set threshold value. Depending on the input value, the Huber function decides on the basis of the set threshold value whether to treat the corresponding entry with a quadratic function or a linear function. The proposed technique updates the threshold value of the Huber function after some iterations to increase the robustness of the Huber function against outliers. Therefore, it is referred as the *Re-Thresholded Huber* function.

The threshold value generally represents the threshold for outliers, that is, if an input value is greater than the threshold then it is treated as outlier, otherwise it is treated as an inlier. Thus, the threshold value plays an important role in defining the robustness of the Huber function. There is no fixed rule to determine the optimum value of the threshold. Depending on the type of problem the optimum value of threshold may vary. The threshold value can not be set to very large or very small values because it determines the dominant behavior of the resultant function. For large values of threshold the influence of outliers is magnified by the quadratic behavior of the Huber function. However, for small values of threshold the resultant function behaves like a linear function and suffers from a problem of being close to non-differentiable. Therefore, a careful selection strategy for threshold is very important for the robustness of the resultant function.

We take advantage of the iterative behavior of the LM method and update, specifically reduce, the threshold value of the Huber function after some fixed number of iteration. Given an initial threshold value, the Re-Thresholded Huber function solves for (6.37) using the LM method and after some iterations the threshold value is reduced by some factor. This way, after some number of iterations the Huber function will have a dominant linear behavior and consequently increased robustness.

6.7 Experiments

We apply the proposed algorithms to a subset (20 views) of the popular and publicly available NotreDame dataset. Our experiments are focused on the following aspects of the proposed algorithms: The first experiment shows the convergence behavior of the proposed algorithms

on real data. The second experiment is designed to show the robustness of the proposed algorithms to outliers. In this case, different percentage of outliers is added to the dataset and a comparison of the results of the proposed algorithms is presented. The last experiment shows the convergence behavior of the L_1-bundle adjustment from different starting points, close to each other.

In the case of the convergence behavior of the proposed algorithms on real data, parameter estimates of the Absolute Value and re-thresholded Isotropic Huber methods are closer to the ground truth than the rest of the algorithms, for details see fig. 6.8. On the other hand, in the presence of outliers, as expected the L_2-bundle adjustment has higher error than the rest of the techniques, and the parameter estimates of the re-thresholded Isotropic Huber method are closer to the ground truth than the rest of the algorithms, as shown in fig. 6.9. In the end, it is shown that the convergence behavior of the L_1-bundle adjustment does not change a lot when different, but sufficiently close, starting points are used, as shown in fig. 6.10.

In our experiments we assume that the intrinsic camera parameters are known. Therefore, we only solve for rotations and translations of cameras, and 3D structure of the scene. Furthermore, we assume that the ground truth parameters are known and errors are computed by using these values. Note that the error curves shown in the figures are not monotonically decreasing because the errors reported are different than the error minimized by the bundle adjustment algorithms, that is the re-projection error. Furthermore, the plots shown in the figures are cropped to enhance the visibility.

Starting Point: Bundle adjustment being an iterative technique requires a good starting point. Since, the primary purpose of this chapter is not on how to generate a good starting point for bundle adjustment, we add noise to the known ground truth parameters and use these parameters as a starting point.

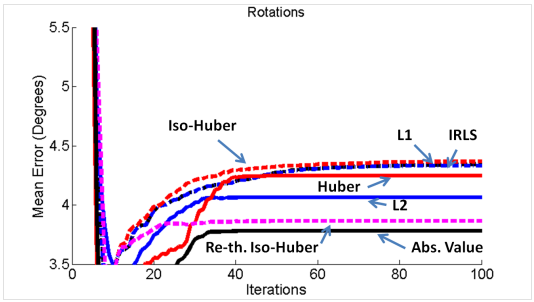
- *Rotation parameters:* Each camera rotation is perturbed by a maximum angle of 5°, by creating a rotation matrix from a random vector (maximum magnitude 5°) and then multiplying with the ground truth rotation.

- 3*D points and Camera centers:* Each component of $\mathbf{X}$ (3D point) and $\mathbf{C}$ (camera center) is scaled by a maximum factor of 0.2. Note that each component is modified independently.

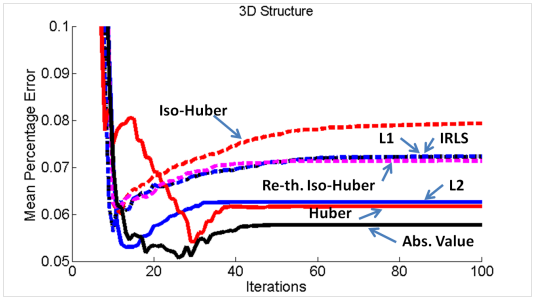
Outliers: The percentage of outliers in the dataset varies in different experiments. We modify some percentage of projection points or image points of every 3D point to represent outliers. Let n be the percentage of outliers to be added in the data. A Gaussian noise of zero mean and standard deviation of 25 is added to $n\%$ of the projections of every 3D point.

Error Measures: Here we discuss error computation methods for rotations, camera centers and 3D points. All errors are computed with respect to the known ground truth values.

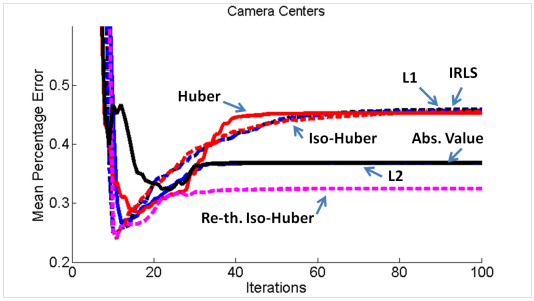
- *Error in Rotations:* Let $\{\mathtt{R}_i\}$ be a set of estimated rotation matrices and $\{\hat{\mathtt{R}}_i\}$ be the corresponding set of ground truth rotation matrices. Let $\mathtt{T}$ be the L_2 mean [Hartley et al., 2011] of the rotation matrices $\mathtt{R}_i\hat{\mathtt{R}}_i^T$. Then, error in degrees for each rotation matrix is



(a) Mean Error Rotations (Degrees)



(b) Mean Percentage Error 3D Points



(c) Mean Percentage Error Camera Centers

Figure 6.8: *Convergence Behavior: The above figures show the convergence behavior of the proposed algorithms on real data, with no outliers added explicitly. The above plots show that the Absolute Value and re-thresholded Isotropic Huber techniques give better results than the rest of the techniques. Thus, whenever there are not many outliers in the dataset then the Absolute value and re-thresholded Isotropic Huber techniques are recommended.*

computed by finding the difference in angle between the ground truth rotation $\hat{\mathtt{R}}_i$ and the transformed rotation $\mathtt{R}_i^{'} = \mathtt{T}\mathtt{R}_i$. Finally, the mean of errors is reported.

- *Error in Camera Centers and 3D points:* We report a mean percentage error between true and estimated 3D point clouds. Let $\{\mathbf{x}_i\}$ be a set of k ground truth points and $\{\mathbf{y}_i\}$ be a set of estimated points. Let $\mathtt{T}$ be the transformation between the points clouds, computed using the Horn's method [Horn et al., 1988]. The mean percentage error e between points is computed as,

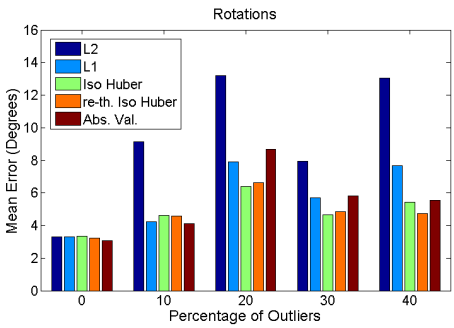
$$e = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{\text{abs}(\|\mathbf{x}_i\| - \|\mathtt{T}\mathbf{y}_i\|)}{\|\mathbf{x}_i\|} \times 100\right) ,$$

The same technique is used for computing errors in camera centers.

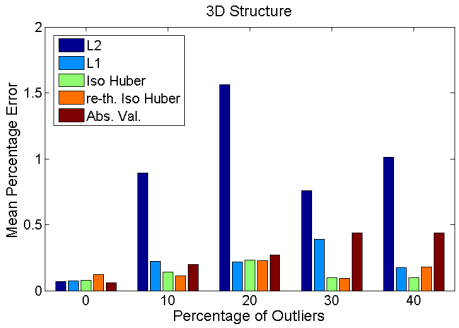
In the rest of the section we will discuss our experimental results.

6.7.1 Convergence Behavior

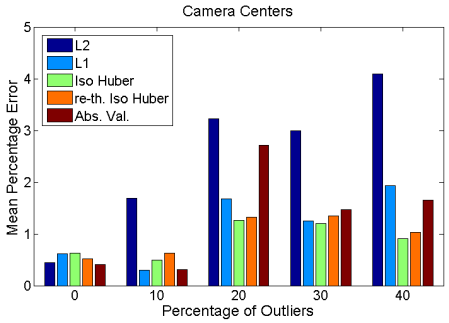
This experiment demonstrates the convergence behavior of the proposed algorithms on real data, with no outliers added explicitly. Since the proposed techniques are advantageous in the



(a) Mean Error Rotations (Degrees)



(b) Mean Percentage Error 3D Points



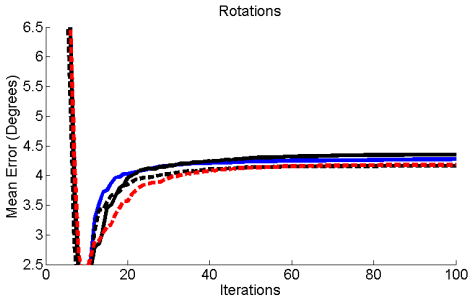
(c) Mean Percentage Error Camera Centers

Figure 6.9: *Robustness to Outliers: The above figures show the results of the proposed methods in the presence of different number of outliers in the dataset. In this case the percentage of outliers is varied from* 0% *to* 40% *with an increment of* 10%*. The results have shown that the re-thresholded Isotropic Huber technique has the least error, except when there are no outliers in the dataset. Furthermore, as expected, the results of the* L_2 *method are far from the ground truth than the rest of the techniques. Therefore, in the presence of outliers the re-thresholded Isotropic Huber technique is recommended because of its superior results and adaptive threshold values.*

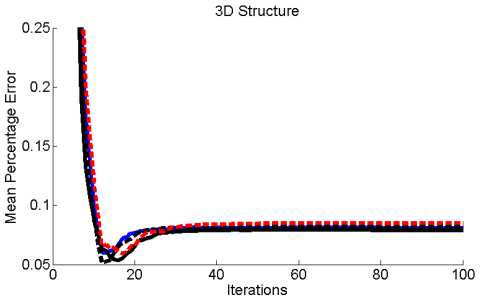
presence of outliers, we do not expect the results of the proposed techniques to be much different than the existing least squares technique. As described before that the starting point of algorithms is generated by modifying the ground truth parameters, where rotation matrices are perturbed by a maximum of 5°, and camera centers and 3D points are scaled by a maximum factor of 0.2. It evident from fig. 6.8 that the results of the Absolute Value and re-thresholded Isotropic Huber techniques are close to the ground truth than the rest of the techniques. Thus, whenever there are not many outliers in the dataset then the Absolute value and re-thresholded Isotropic Huber techniques are recommended. Since the LM method is a numerical optimization method, the results of the L_1 and IRLS methods are the same.

6.7.2 Robustness to Outliers

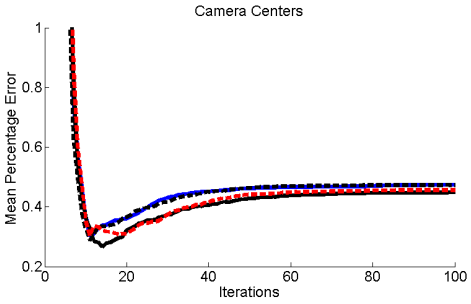
This experiment demonstrates the robustness of the proposed algorithms against outliers. In order to achieve this purpose we add different percentage of outliers to the dataset ranging from



(a) Mean Error Rotations (Degrees)



(b) Mean Percentage Error 3D Points



(c) Mean Percentage Error Camera Centers

Figure 6.10: *Convergence from Different Starting Points (L_1-bundle adjustment): The above figures show the convergence behavior of the L_1 method from different starting points. Each colored line represents a different starting point of the algorithm. Results have shown that the convergence behavior of the L_1 method does not change much with a change in the starting points of the algorithm and that the parameters recovered are generally close to each other.*

0% to 40% with an increment of 10%. We are particularly interested in the L_1, L_2, Absolute Value, Isotropic Huber and re-thresholded Isotropic algorithms. As expected, the results of the L_2 algorithm are far from the ground truth values than the rest of the techniques. However, the parameter estimates of the re-thresholded Isotropic Huber algorithm are closer to ground truth than the other proposed algorithms. Note that the Absolute Value algorithm gives superior results than the rest of the algorithms only when no outliers are added explicitly to the dataset, that is real data. Thus, if the robustness against outliers is desired then the re-thresholded Isotropic Huber algorithm is recommended.

6.7.3 Convergence from Different Starting Points

Since bundle adjustment uses the Levenberg-Marquardt method, a numerical optimization method, to find a solution, the final output of the algorithm depends on the starting points of the algorithm. This experiment shows that the convergence behavior of the L_1-bundle adjustment does not change much with a small change in the starting point of the algorithm. Here, we test the algorithm by taking different starting points close to each other. In order to generate different starting points for our algorithm, in addition to the basic noise in parameters as defined before, we further perturb the rotations by 1°, and camera center and 3D points by a

factor of 0.05. Each colored line in fig. 6.10 indicates the convergence of the L_1 technique from a different starting point. These plots have shown that the results of the L_1-bundle adjustment will not change drastically with a small change in initial parameter estimates and that the final parameter estimates are mostly close to each other.

6.8 Summary

In summary, we proposed techniques to minimize several robust cost functions, especially an L_q cost function, for $1 \leq q < 2$, and the Huber function. An advantage of the proposed techniques is that an existing implementation of the L_2 bundle adjustment can be modified to minimized a desired cost function because the proposed techniques still use the LM method to find a robust solution. Furthermore, our experimental results on the NotreDame set showed that the proposed techniques are more robust to outliers than the L_2 bundle adjustment. A Simple approach and easy implementation makes the proposed algorithms practically feasible. The applicability of proposed techniques is not only limited to bundle adjustment problem but a wide class of non-linear parameter estimation problems, that use the Levenberg-Marquardt method for minimization, can be solved robustly using these techniques.

Chapter 7

Conclusion

We proposed several methods to solve two classes of problems: Firstly, L_q-closest-points problems where we seek a point for which the sum of the q-th power of distances to a given set of measurements, such as points, lines, subspaces, or their mixture, is the minimum. Secondly, non-linear parameter estimation problems where we solve for a solution of the L_q-bundle adjustment problem by minimizing the sum of the q-th power of re-projection errors. Our experimental results, confirm the fact that the proposed L_q techniques, for $1 \leq q < 2$, mitigate the effect of outliers and are more robust to outliers than the L_2 methods.

L_q-closest-point Problems: In the first part of the thesis we proposed several generalizations of the Weiszfeld algorithm. The first generalization, the L_q Weiszfeld algorithm, solves for the L_q mean of a set of points in $\mathbb{R}^N$. Later, this algorithm is generalized to find the L_q mean of a set of points on a Riemannian manifold of non-negative sectional curvature. In addition to the proof of convergence, we showed that the convergence of the proposed algorithm is guaranteed even when the bounds on the maximum distance between points on a manifold are more relaxed than the existing L_1 methods in the literature. In the end, we proposed an algorithm to find the L_q-closest-point to a set of affine subspaces, possibly of different dimensions, in $\mathbb{R}^N$.

The proposed generalizations inherit all the advantages of the Weiszfeld algorithm, such as guaranteed convergence to the L_q minimum, analytical updates, etc. Unlike other gradient descent methods that rely on an expensive line search for an update step, our methods compute updates analytically. Therefore, the L_q algorithms proposed here are substantially simpler than the line-search based gradient-descent algorithms. Below we summarize some features of the closest-point algorithms:

- Provable convergence to the L_q minimum, for $1 \leq q < 2$.
- Updates are computed analytically.
- For $q > 1$, L_q cost functions are differentiable at the minimum.
- Simple to understand methods these algorithms iteratively solve weighted L_2 cost functions to find the L_q solution.
- Easy to code, because an existing least squares implementation can be modified to find the desired solution.

In addition to the theoretical proof for the convergence of the L_q-closest-point algorithms to the L_q minimum, we showed the applicability of the proposed algorithms by solving the problems of rotation averaging, symmetric positive-definite matrices averaging, and triangulation. Our experimental results showed that the results of the L_q algorithms, for $1 < q \leq 2$, are closer to the ground truth than the L_2 algorithms. Although, in some cases the error minimized is different than the error reported but still the results of the L_q methods, especially for $q = 1$, are superior than the L_2 methods.

Parameter Estimation Problems: In the last part of the thesis we proposed several methods to find a robust solution of the bundle adjustment problem, especially an L_q solution, for $1 \leq q < 2$. Once again, the proposed methods use a non-linear least squares method, namely the Levenberg-Marquardt method, to find a robust solution, even an L_q solution. Therefore, an existing implementation of the L_2 bundle adjustment is used, with slight modifications, to find a desired robust solution. Our experimental results on the NotreDame set confirmed that the proposed algorithms are more robust to ouliters than the least squares bundle adjustment.

The Simple approach and easy implementation of the proposed L_1-bundle adjustment makes it practically useful. The applicability of proposed techniques is not only limited to bundle adjustment problem but a more general class of non-linear problems, that uses the Levenberg-Marquardt method for minimization, can be solved robustly using these techniques.

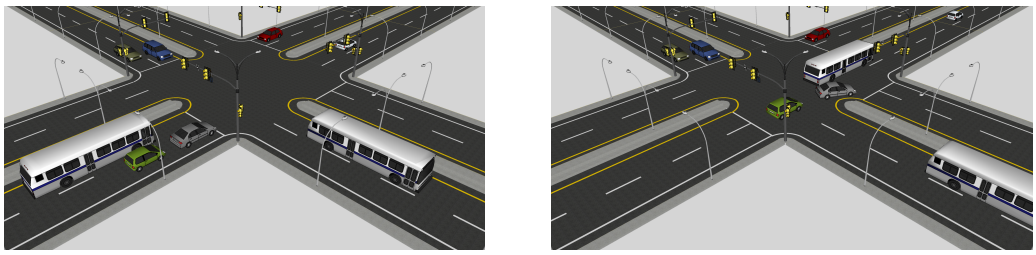
Ease of implementation and fast iteration make the proposed algorithms attractive wherever L_q optimization is desired. An interesting observation is that the L_q methods, for $q = 1$, tend to emphasize (and largely ignore) outliers by allowing large individual errors to occur, whereas L_2 methods will strive to keep all errors low. For this reason the L_1 methods may be useful in identifying outliers. A question that remained partially unanswered is, which computer vision problems can be solved using this technique?

7.1 Future Work

Here we discuss some future research directions of the proposed algorithms. Note that future work for some contributions of this thesis have already been discussed in the corresponding chapters, and are not repeated here.

7.1.1 RANSAC-style Algorithm

In the presence of a large proportion of outliers in data a RANSAC-style algorithm, based on L_q algorithms, can be proposed to robustly estimate parameters of a model. RANSAC is an iterative method to estimate parameters of a model from a set of measurements and is designed to cope with a large proportion of outliers in data. Instead of using as much of the data as possible to obtain an initial solution, RANSAC generates candidate solutions by using the minimum number of observations required to estimate the underlying model parameters. It uses the smallest set possible to obtain an initial solution and proceeds to enlarge this set with consistent data points.



(a) Frame 1 (b) Frame 2

Figure 7.1: *Multi-body Structure and Motion: The above figure shows two images of a scene where the position of camera is fixed, while objects (vehicles) in the scene undergo motion in different directions independent of each other.*

As mentioned before, L_q methods are generally more robust to outliers than L_2 methods. For example, the geometric median or L_1 mean of a set of points has a breakdown point of 50%, which means that the L_1 mean remains unaffected by outliers unless more than 50% of points are outliers. Therefore, a RANSAC-inspired approach, based on L_q optimization techniques, can be used to find a robust solution of a problem in the presence of a large proportion of outliers in data. A RANSAC-inspired approach can be used to generate some candidate models by selecting a subset (not the minimal set) of measurements, estimating parameters of a model using an L_q algorithm and repeating the procedure for a different subset of points. Finally, from among the candidate models, a model is selected that has the minimum error. Clearly, this procedure requires lesser number of iterations than the RANSAC algorithm that estimates model parameters from a minimal set of points.

7.1.2 Multi-body Structure and Motion (MSaM) through Model Selection

Multi-body structure and motion (MSaM) is closely related to the problem of structure and motion (SaM). Structure and motion is the problem of estimating geometric parameters from multiple images of a scene under the assumption that there is a single dominant motion in scene. It is a well studied problem in computer vision and a variety of method have been developed to solve this problem [Longuet-Higgins, 1981; Hartley, 1992]. On the other hand, Multi-body structure and motion recovers parameters of a dynamic scene where objects in the scene undergo a rigid body motion. Several methods have been proposed to solve the problem of MSaM such as multi-body fundamental matrix [Vidal and Sastry, 2003, 2002], multi-body homographies [Vidal and Ma, 2004], etc.

An alternative approach to tackle the MSaM problem is use a clustering based approach where points are clustered based on their motions [Schindler and Suter, 2006]. This approach is based on recover-and-select scheme where several candidate models are recovered based on Monte-Carlo sampling of the image point correspondences and models are selected on the basis of their residual errors. The output of the clustering based method depends on the robustness of the method that is used to instantiate the motion models. Thus, we can use L_q minimization techniques to generate candidate models. Once candidate models are generated, models that best describe the measurements can be selected.

Bibliography

(cited on page 46)

AFSARI, B., 2011. Riemannian L^p center of mass: Existence, uniqueness, and convexity. *Proceedings of the American Mathematical Society*, 139 (2011), 655–673. (cited on pages 46 and 51)

AFSARI, B.; TRON, R.; AND VIDAL, R., 2013. On the convergence of gradient descent for finding the Riemannian center of mass. *SIAM Journal on Control and Optimization*, 51, 3 (2013), 2230–2260. (cited on page 54)

AFTAB, K.; HARTLEY, R.; AND TRUMPF, J. Generalized Weiszfeld algorithms for Lq optimization. Submitted. (cited on pages 60, 79, 80, 81, 85, 87, 115, and 116)

AGARWAL, S.; SNAVELY, N.; SEITZ, S.; AND SZELISKI, R., 2010. Bundle adjustment in the large. In *Computer Vision âĂŞ ECCV 2010* (Eds. K. DANIILIDIS; P. MARAGOS; AND N. PARAGIOS), vol. 6312 of *Lecture Notes in Computer Science*, 29–42. Springer Berlin / Heidelberg. ISBN 978-3-642-15551-2. (cited on page 105)

AMERI, B. AND FRITSCH, D., 2000. Automatic 3d building reconstruction using plane-roof structures. *ASPRS, Washington DC*, (2000). (cited on page 81)

ANDO, T.; LI, C.-K.; AND MATHIAS, R., 2004. Geometric means. *Linear algebra and its applications*, (2004). (cited on page 46)

ARNAUDON, M.; DOMBRY, C.; PHAN, A.; AND YANG, L., 2012. Stochastic algorithms for computing means of probability measures. *Stochastic Processes and their Applications*, (2012). (cited on page 46)

ARSIGNY, V.; FILLARD, P.; PENNEC, X.; AND AYACHE, N., 2007. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM journal on matrix analysis and applications*, 29, 1 (2007), 328–347. (cited on pages 9, 46, 47, 57, 58, 59, 60, and 61)

BHATIA, R. AND HOLBROOK, J., 2006. Riemannian geometry and matrix geometric means. *Linear algebra and its applications*, (2006). (cited on page 46)

BINI, D.; MEINI, B.; AND POLONI, F., 2010. An effective matrix geometric mean satisfying the ando-li-mathias properties. *Mathematics of Computation*, (2010). (cited on page 46)

BINI, D. A. AND IANNAZZO, B., 2011. Computing the karcher mean of symmetric positive definite matrices. *Linear Algebra and its Applications*, (2011). (cited on page 47)

BONNABEL, S.; COLLARD, A.; AND SEPULCHRE, R., 2013. Rank-preserving geometric means of positive semi-definite matrices. *Linear Algebra and its Applications*, (2013). (cited on page 46)

BONNABEL, S. AND SEPULCHRE, R., 2009. Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM Journal on Matrix Analysis and Applications*, (2009). (cited on page 46)

BOROUCHAKI, H.; GEORGE, P. L.; HECHT, F.; LAUG, P.; AND SALTEL, E., 1997. Delaunay mesh generation governed by metric specifications. part i. algorithms. *Finite elements in analysis and design*, 25, 1 (1997), 61–83. (cited on pages 9 and 46)

BOUMAL, N.; SINGER, A.; AND ABSIL, P.-A., 2013. Robust estimation of rotations from relative measurements by maximum likelihood. In *IEEE Conference on Decision and Control*. (cited on page 46)

BRIMBERG, J., 2003. Further notes on convergence of the weiszfeld algorithm. *Yugoslav Journal of Operations Research*, 13, 2 (2003), 199–206. (cited on page 21)

BRIMBERG, J. AND CHEN, R., 1998. A note on convergence in the single facility minisum location problem. *Computers & Mathematics with Applications*, 35, 9 (1998), 25–31. (cited on page 21)

BRIMBERG, J. AND LOVE, R. F., 1993. Global convergence of a generalized iterative procedure for the minisum location problem with lp distances. *Operations Research*, 41, 6 (1993), 1153–1163. (cited on page 21)

BROX, T.; ROUSSON, M.; DERICHE, R.; AND WEICKERT, J., 2003. Unsupervised segmentation incorporating colour, texture, and motion. In *Computer analysis of images and patterns*, 353–360. Springer. (cited on pages 9 and 46)

CETINGUL, H. E.; AFSARI, B.; WRIGHT, M. J.; THOMPSON, P. M.; AND VIDAL, R., 2012. Group action induced averaging for hardi processing. In *International Symposium on Biomedical Imaging*. IEEE. (cited on pages 46, 47, and 58)

CHARTRAND, R. AND YIN, W., 2008. Iteratively reweighted algorithms for compressive sensing. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conferenc on*, 3869 –3872. doi:10.1109/ICASSP.2008.4518498. (cited on page 4)

CHAVEL, I., 2006. *Riemannian Geometry*. Cambridge University Press, 2nd edn. (cited on page 49)

CHEEGER, J. AND EBIN, D. G., 1975. *Comparison theorems in Riemannian geometry*, vol. 365. AMS Bookstore. (cited on pages 49 and 75)

COLLARD, A.; BONNABEL, S.; PHILLIPS, C.; AND SEPULCHRE, R., 2012. An anisotropy preserving metric for DTI processing. *CoRR*, abs/1210.2826 (2012). http://arxiv.org/abs/1210.2826. (cited on pages 46 and 47)

CUCURINGU, M.; LIPMAN, Y.; AND SINGER, A., 2012a. Sensor network localization by eigenvector synchronization over the euclidean group. *ACM Transactions on Sensor Networks (TOSN)*, 8, 3 (2012), 19. (cited on pages 9 and 46)

CUCURINGU, M.; SINGER, A.; AND COWBURN, D., 2012b. Eigenvector synchronization, graph rigidity and the molecule problem. *Information and Inference*, 1, 1 (2012), 21–67. (cited on pages 9 and 46)

DAI, Y.; TRUMPF, J.; LI, H.; BARNES, N.; AND HARTLEY, R., 2009. Rotation averaging with application to camera-rig calibration. In *Proc. Asian Conference on Computer Vision, Xian*. .. /Papers/PDF/Yuchao:ACCV09.pdf. (cited on pages 46, 53, and 54)

DAUBECHIES, I.; DEVORE, R.; FORNASIER, M.; AND GUNTURK, S., 2008. Iteratively re-weighted least squares minimization: Proof of faster than linear rate for sparse recovery. In *42nd Annual Conference on Information Sciences and Systems, 2008.*, 26 –29. doi:10.1109/CISS.2008.4558489. (cited on page 4)

DICK, A. R.; TORR, P. H.; RUFFLE, S. J.; AND CIPOLLA, R., 2001. Combining single view recognition and multiple view stereo for architectural scenes. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, 268–274. IEEE. (cited on page 81)

DOLD, A., 1995. *Lectures on Algebraic Topology*. Springer. (cited on page 74)

ECKHARDT, U., 1980. Weber's problem and weiszfeld's algorithm in general spaces. *Mathematical Programming*, 18 (1980), 186–196. http://dx.doi.org/10.1007/BF01588313. 10.1007/BF01588313. (cited on page 21)

ELDAR, Y. AND MISHALI, M., 2009. Robust recovery of signals from a structured union of subspaces. *IEEE Transactions on Information Theory*, 55, 11 (nov. 2009), 5302 –5316. doi:10.1109/TIT.2009.2030471. (cited on page 4)

ENGELS, C.; STEWÃL'NIUS, H.; AND NISTÃL'R, D., 2006. Bundle adjustment rules. In *In Photogrammetric Computer Vision*. (cited on pages 11 and 104)

EUDES, A. AND LHUILLIER, M., 2009. Error propagations for local bundle adjustment. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009.*, 2411–2418. IEEE. (cited on page 105)

FILLARD, P.; ARSIGNY, V.; PENNEC, X.; HAYASHI, K. M.; THOMPSON, P. M.; AND AYACHE, N., 2007. Measuring brain variability by extrapolating sparse tensor fields measured on sulcal lines. *Neuroimage*, 34, 2 (2007), 639–650. (cited on pages 9 and 46)

FIORI, S., 2009. Learning the fréchet mean over the manifold of symmetric positive-definite matrices. *Cognitive Computation*, 1, 4 (2009), 279–291. (cited on page 47)

FLETCHER, P. T. AND JOSHI, S., 2004. Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, 87–98. Springer. (cited on pages 9, 47, and 57)

FLETCHER, P. T. AND JOSHI, S., 2007. Riemannian geometry for the statistical analysis of diffusion tensor data. *Signal Processing*, 87, 2 (2007), 250–262. (cited on pages 9 and 46)

FLETCHER, P. T.; VENKATASUBRAMANIAN, S.; AND JOSHI, S., 2009. The geometric median on riemannian manifolds with applications to robust atlas estimation. *Neuroimage*, 45 (1 Suppl) (2009), 143–152. doi:doi:10.1016/j.neuroimage.2008.10.052. (cited on pages 8, 13, 21, 22, 23, 45, 46, 47, and 49)

FURUKAWA, Y.; CURLESS, B.; SEITZ, S.; AND SZELISKI, R., 2009a. Manhattan-world stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1422 – 1429. doi:10.1109/CVPR.2009.5206867. (cited on pages 80 and 81)

FURUKAWA, Y.; CURLESS, B.; SEITZ, S. M.; AND SZELISKI, R., 2009b. Reconstructing building interiors from images. In *12th IEEE International Conference on Computer Vision, 2009*, 80–87. IEEE. (cited on page 81)

GOVINDU, V. M., 2001. Combining two-view constraints for motion estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 218–225. doi:http://doi.ieeecomputersociety.org/10.1109/CVPR.2001.990963. (cited on page 46)

GOVINDU, V. M., 2004. Lie-algebraic averaging for globally consistent motion estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 684–691. doi:http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.147. (cited on pages 46 and 55)

HARTLEY, R.; AFTAB, K.; AND TRUMPF, J., 2011. L1 rotation averaging using the Weiszfeld algorithm. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 3041 – 3048. doi:10.1109/CVPR.2011.5995745. (cited on pages 21, 46, and 123)

HARTLEY, R. AND SCHAFFALITZKY, F., 2004. L_∞ minimization in geometric reconstruction problems. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Washington DC*, I–504–509. doi:doi:10.1109/CVPR.2004.1315073. http://dx.doi.org/10.1109/CVPR.2004.1315073. (cited on page 46)

HARTLEY, R.; TRUMPF, J.; DAI, Y.; AND LI, H., 2013. Rotation averaging. *International Journal of Computer Vision*, (2013). doi:doi:10.1007/s11263-012-0601-0. ../Papers/PDF/Hartley-Trumpf:Rotation-averaging:IJCV.pdf. (cited on pages 21, 50, 51, 54, and 77)

HARTLEY, R. I., 1992. Estimation of relative camera positions for uncalibrated cameras. In *Proc. 2nd European Conference on Computer Vision, Santa Margharita Ligure, Italy*, LNCS 588, 579–587. Springer-Verlag. doi:doi:10.1007/3-540-55426-2_62. http://dx.doi.org/10.1007/3-540-55426-2_62. (cited on page 131)

HARTLEY, R. I., 1998. Minimizing algebraic error. *Philosophical Transactions of the Royal Society of London, SERIES A*, 356, 1740 (1998), 1175–1192. (cited on page 64)

HARTLEY, R. I. AND STURM, P., 1997. Triangulation. *Computer Vision and Image Understanding*, 68, 2 (November 1997), 146–157. doi:doi:10.1006/cviu.1997.0547. http://dx.doi.org/10.1006/cviu.1997.0547. (cited on pages 80, 81, and 98)

HARTLEY, R. I. AND ZISSERMAN, A., 2004. *Multiple View Geometry in Computer Vision – 2nd Edition*. Cambridge University Press. (cited on pages 11, 54, 63, 81, 96, 104, 105, and 107)

HENRY, P.; KRAININ, M.; HERBST, E.; REN, X.; AND FOX, D., 2010. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, vol. 20, 22–25. (cited on page 81)

HOLMES, S.; SIBLEY, G.; KLEIN, G.; AND MURRAY, D., 2009. A relative frame representation for fixed-time bundle adjustment in sfm. In *IEEE International Conference on Robotics and Automation, 2009.*, 2264–2269. IEEE. (cited on page 105)

HORN, B. K. P.; HILDEN, H.; AND NEGAHDARIPOUR, S., 1988. Closed-form solution of absolute orientation using orthonormal matrices. *JOURNAL OF THE OPTICAL SOCIETY AMERICA*, 5, 7 (1988), 1127–1135. (cited on page 124)

HUBER, P., 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35, 1 (1964), 73–101. (cited on page 118)

KAHL, F., 2005. Multiple view geometry and the L_∞-norm. In *Proc. International Conference on Computer Vision*, 1002–1009. (cited on page 46)

KAUCIC, R.; HARTLEY, R. I.; AND DANO, N. Y., 2001. Plane-based projective reconstruction. In *Proc. 8th International Conference on Computer Vision, Vancouver, Canada*, I–420–427. doi:doi:10.1109/ICCV.2001.10058. http://dx.doi.org/10.1109/ICCV.2001.10058. (cited on page 46)

KLINGENBERG, W., 1982. *Riemannian Geometry*. Walter de Gruyter. (cited on page 49)

KRAKOWSKI, K.; HÜPER, K.; AND MANTON, J., 2007. On the computation of the karcher mean on spheres and special orthogonal groups. In *RoboMat 2007, Workshop on Robotics and Mathematics*. Coimbra, Portugal. (cited on page 54)

KULIN, H. W. AND KUENNE, R. E., 1962. An efficient algorithm for the numerical solution of the generalized weber problem in spatial economics. *Journal of Regional Science*, 4, 2 (1962), 21–33. (cited on page 16)

LANG, S., 1999. *Fundamentals of Differential Geometry*. Springer. (cited on page 74)

LEE, H.; LIM, Y.; AND YAMAZAKI, T., 2011. Multi-variable weighted geometric means of positive definite matrices. *Linear Algebra and its Applications*, (2011). (cited on page 46)

LEE, J. M., 1997. *Riemannian manifolds: an introduction to curvature*, vol. 176. Springer. (cited on page 50)

LENGLET, C.; ROUSSON, M.; DERICHE, R. D.; FAUGERAS, O.; ET AL., 2004. Statistics on multivariate normal distributions: A geometric approach and its application to diffusion tensor mri. (2004). (cited on pages 9, 47, and 57)

LEVENBERG, K., 1944. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, 2 (1944), 164–168. (cited on page 107)

LONGUET-HIGGINS, H. C., 1981. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293 (September 1981), 133–135. (cited on page 131)

LOURAKIS, M. A. AND ARGYROS, A., 2009. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36, 1 (2009), 1–30. doi:http://doi.acm.org/10.1145/1486525.1486527. (cited on page 100)

LUENBERGER, D. G., 2003. *Linear and nonlinear programming*. Springer. (cited on page 29)

MA, R., 2004. *Building model reconstruction from LiDAR data and aerial photographs*. Ph.D. thesis, The Ohio State University. (cited on page 81)

MANTON, J. H., 2004. A globally convergent numerical algorithm for computing the centre of mass on compact Lie groups. In *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision*, 2211–2216. Kunming, China. (cited on page 46)

MARQUARDT, D. W., 1963. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11 (1963), 431–441. (cited on page 107)

MARTINEC, D. AND PAJDLA, T., 2007. Robust rotation and translation estimation in multiview reconstruction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. doi:10.1109/CVPR.2007.383115. (cited on page 46)

MEYER, W., 1989. ToponogovŠs theorem and applications. *Lecture Notes, Trieste*, (1989). (cited on pages 75 and 76)

MOAKHER, M., 2002. Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl.*, 24, 1 (2002), 1–16 (electronic). (cited on page 46)

MOAKHER, M., 2005. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 26, 3 (2005), 735–747. (cited on pages 9, 47, and 57)

MOAKHER, M., 2006. On the averaging of symmetric positive-definite tensors. *Journal of Elasticity*, 82, 3 (2006), 273–296. (cited on pages 9 and 46)

MORRIS, J. G. AND VERDINI, W. A., 1979. Technical noteŮminisum lp distance location problems solved via a perturbed problem and weiszfeld's algorithm. *Operations Research*, (1979). (cited on page 21)

MOURAGNON, E.; LHUILLIER, M.; DHOME, M.; DEKEYSER, F.; AND SAYD, P., 2006. Real time localization and 3d reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006.*, vol. 1, 363–370. IEEE. (cited on page 105)

MÜLLER, P.; ZENG, G.; WONKA, P.; AND VAN GOOL, L., 2007. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26, 3 (2007), 85. (cited on page 81)

MYERS, S., 1945. Arcs and geodesics in metric spaces. *Transactions of the American Mathematical Society*, 57, 2 (1945), 217–227. (cited on page 48)

NIINIMAA, A.; OJA, H.; AND TABLEMAN, M., 1990. The finite-sample breakdown point of the oja bivariate median and of the corresponding half-samples version. *Statistics & Probability Letters*, 10, 4 (1990), 325–328. (cited on page 15)

NISTÉR, D., 2004. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26, 6 (2004), 756–777. doi:http://dx.doi.org/10.1109/TPAMI.2004.17. (cited on page 62)

NOCEDAL, J. AND WRIGHT, S. J., 1999. *Numerical optimization.* Springer Verlag, New York, NY,. (cited on page 29)

OJA, H., 1983. Descriptive statistics for multivariate distributions. *Statistics & Probability Letters*, 1, 6 (1983), 327–332. (cited on page 15)

PENNEC, X.; FILLARD, P.; AND AYACHE, N., 2006. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66, 1 (2006), 41–66. (cited on pages 9, 46, 47, 57, 59, and 61)

PETERSEN, P., 2006. *Riemannian geometry*, vol. 171. Springer. (cited on page 49)

PETZ, D. AND TEMESI, R., 2005. Means of positive numbers and matrices. *SIAM journal on Matrix Analysis and Applications*, (2005). (cited on page 46)

PLASTRIA, F. AND ELOSMANI, M., 2008. On the convergence of the weiszfeld algorithm for continuous single facility location–allocation problems. *Top*, 16, 2 (2008), 388–406. (cited on page 16)

PORIKLI, F.; TUZEL, O.; AND MEER, P., 2006. Covariance tracking using model update based on lie algebra. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006*, vol. 1, 728–735. IEEE. (cited on pages 9 and 46)

PU, S. AND VOSSELMAN, G., 2009. Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64, 6 (2009), 575–584. (cited on page 81)

REMONDINO, F. AND EL-HAKIM, S., 2006. Image-based 3d modelling: A review. *The Photogrammetric Record*, 21, 115 (2006), 269–291. (cited on page 81)

ROTHER, C. AND CARLSSON, S., 2001. Linear multi view reconstruction and camera recovery. In *Proc. 8th International Conference on Computer Vision, Vancouver, Canada*, I–42–49. (cited on page 46)

SALENÇON, J., 2001. *Handbook of continuum mechanics: general concepts-thermoelasticity*. Springer. (cited on pages 9 and 46)

SARLETTE, A. AND SEPULCHRE, R., 2009. Consensus optimization on manifolds. *SIAM J. Control Optim.*, 48, 1 (2009), 56–76. (cited on page 46)

SCHINDLER, K. AND BAUER, J., 2003. A model-based method for building reconstruction. In *First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003*, 74–82. IEEE. (cited on page 81)

SCHINDLER, K. AND SUTER, D., 2006. Two-view multibody structure-and-motion with outliers through model selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28, 6 (june 2006), 983 –995. doi:10.1109/TPAMI.2006.130. (cited on page 131)

SHUM, H.; KE, Q.; AND ZHANG, Z., 1999. Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999*, vol. 2. IEEE. (cited on page 105)

SIM, K. AND HARTLEY, R., 2006. Recovering camera motion using L_∞ minimization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City*, 1230 – 1237. doi:doi:10.1109/CVPR.2006.247. http://dx.doi.org/10.1109/CVPR.2006.247. (cited on page 46)

SMALL, C. G., Dec. 1990. A survey of multidimensional medians. *International Statistical Review / Revue Internationale de Statistique*, 58 (Dec. 1990), 263–277. http://www.jstor.org/stable/1403809. (cited on pages 15 and 18)

SNAVELY, N.; SEITZ, S. M.; AND SZELISKI, R., 2006. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06* (Boston, Massachusetts, 2006), 835–846. ACM, USA. doi:http://doi.acm.org/10.1145/1179352.1141964. (cited on page 62)

SNAVELY, N.; SEITZ, S. M.; AND SZELISKI, R., 2008. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80, 2 (2008), 189–210. doi:http://dx.doi.org/10.1007/s11263-007-0107-3. (cited on page 64)

STEEDLY, D. AND ESSA, I., 2001. Propagation of innovative information in non-linear least-squares structure from motion. In *Eighth IEEE International Conference on Computer Vision, 2001*, vol. 2, 223–229. IEEE. (cited on page 105)

STEEDLY, D.; ESSA, I.; AND DELLAERT, F., 2003. Spectral partitioning for structure from motion. In *Proceedings of the 2003 9th IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 996–1003. (cited on page 105)

STRELOW, D., 2012. General and nested wiberg minimization. In *Computer Vision and Pattern Recognition*. http://www.dennis-strelow.com/papers/documents/strelow_cvpr12.pdf. (cited on page 105)

TAILLANDIER, F., 2005. Automatic building reconstruction from cadastral maps and aerial images. *International Archives of Photogrammetry and Remote Sensing*, 36, Part 3 (2005), W24. (cited on page 81)

TRIGGS, B.; MCLAUCHLAN, P. F.; HARTLEY, R. I.; AND FITZGIBBON, A. W., 2000a. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, 298–372. Springer-Verlag, London, UK. (cited on pages 11 and 104)

TRIGGS, W.; MCLAUCHLAN, P. F.; HARTLEY, R. I.; AND FITZGIBBON, A., 2000b. Bundle adjustment for structure from motion. In *Vision Algorithms: Theory and Practice*, 298–372. Springer-Verlag. doi:doi:10.1007/3-540-44480-7_21. http://dx.doi.org/10.1007/3-540-44480-7_21. (cited on pages 80, 98, and 100)

TRON, R.; AFSARI, B.; AND VIDAL, R., 2011. Average consensus on riemannian manifolds with bounded curvature. In *Conference on Decision and Control and European Control Conference*. IEEE. (cited on page 46)

TRON, R.; AFSARI, B.; AND VIDAL, R., 2012. Intrinsic consensus on so (3) with almost-global convergence. In *CDC*. (cited on page 46)

TRON, R.; AFSARI, B.; AND VIDAL, R., 2013. Riemannian consensus for manifolds with bounded curvature. *Transactions on Automatic Control*, (2013). (cited on page 46)

TRON, R. AND VIDAL, R., 2009. Distributed image-based 3-d localization of camera sensor networks. In *IEEE Conference on Decision and Control, 2009*, 901–908. IEEE. (cited on pages 9, 46, and 57)

TRON, R. AND VIDAL, R., 2011. Distributed computer vision algorithms. *Signal Processing Magazine*, (2011). (cited on page 46)

TRON, R.; VIDAL, R.; AND TERZIS, A., 2008. Distributed pose averaging in camera networks via consensus on se (3). In *International Conference on Distributed Smart Cameras*. IEEE. (cited on page 46)

TUKEY, J. W., 1975. Mathematics and the picturing of data. In *Proceedings of the international congress of mathematicians*, vol. 2, 523–531. (cited on page 15)

TYAGI, A. AND DAVIS, J. W., 2008. A recursive filter for linear systems on riemannian manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008*, 1–8. IEEE. (cited on page 47)

TYAGI, A.; DAVIS, J. W.; AND POTAMIANOS, G., 2008. Steepest descent for efficient covariance tracking. In *IEEE Workshop on Motion and video Computing, 2008*, 1–6. IEEE. (cited on pages 9 and 46)

VANEGAS, C. A.; ALIAGA, D. G.; AND BENES, B., 2010. Building reconstruction using manhattan-world grammars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, 358–365. IEEE. (cited on page 81)

VIDAL, R. AND MA, Y., 2004. A unified algebraic approach to 2-d and 3-d motion segmentation. 1–15. (cited on page 131)

VIDAL, R. AND SASTRY, S., 2003. Optimal segmentation of dynamic scenesfrom two perspective views. (cited on page 131)

VIDAL, S. M. Y., R. SOATTO AND SASTRY, S., 2002. Segmentation ofdynamic scenes from the multibody fundamental matrix. (cited on page 131)

WANG, L. AND SINGER, A., 2012. Exact and stable recovery of rotations for robust synchronization. *arXiv preprint arXiv:1211.2441*, (2012). (cited on page 46)

WEBER, A., 1909. *Uber den Standort der Industrien, Erster Teil: Reine Theorie des Standortes*. (cited on page 16)

WEISZFELD, E., 1937. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Math. Journal*, 43 (1937), 355–386. (cited on pages 4, 13, 16, 20, 21, 26, 28, 35, and 86)

WEISZFELD, E. AND PLASTRIA, F., 2009. On the point for which the sum of the distances to n given points is minimum. *Annals of Operations Research*, 167, 1 (2009), 7–41. (cited on page 16)

WERNER, T. AND ZISSERMAN, A., 2002. New techniques for automated architectural reconstruction from photographs. In *ECCV 2002*, 541–555. Springer. (cited on page 81)

WILCZKOWIAK, M.; TROMBETTONI, G.; JERMANN, C.; STURM, P.; AND BOYER, E., 2003. Scene modeling based on constraint system decomposition techniques. In *Ninth IEEE International Conference on Computer Vision, 2003*, 1004–1010. IEEE. (cited on page 81)

YANG, L., 2010. Riemannian median and its estimation. *LMS Journal of Computation and Mathematics*, 13 (2010), 461–479. (cited on pages 21, 23, and 46)