Submitted to the IJCV, manuscript No.
(will be inserted by the editor)

Lq-Closest-Point to Affine Subspaces using the Generalized Weiszfeld Algorithm

Khurrum Aftab · Richard Hartley · Jochen Trumpf

Received: date / Accepted: date

Abstract This paper presents a method for finding an L_q-closest-point to a set of affine subspaces, that is a point for which the sum of the q-th power of orthogonal distances to all the subspaces is minimized, where $1 \leq q < 2$. We give a theoretical proof for the convergence of the proposed algorithm to a unique L_q minimum. The proposed method is motivated by the L_q Weiszfeld algorithm, an extremely simple and rapid averaging algorithm, that finds the L_q mean of a set of given points in a Euclidean space. The proposed algorithm is applied to the triangulation problem in computer vision by finding the L_q-closest-point to a set of lines in 3D. Our experimental results for the triangulation problem confirm that the L_q-closest-point method, for $1 \leq q < 2$, is more robust to outliers than the L_2-closest-point method.

Keywords L_q Weiszfeld algorithm · affine subspaces · triangulation · L_q mean

1 Introduction / Literature Review

We propose a method, based on the L_q Weiszfeld algorithm [1], for finding an L_q-*closest-point* to a set of affine subspaces in $\mathbb{R}^N$, where $1 \leq q < 2$ and the L_q-*closest-point* to subspaces is defined as a point for which the sum of the q-th power of orthogonal distances to the subspaces is minimized. In addition to a theoretical proof for the convergence of the proposed algorithm to a unique L_q minimum, we show that the proposed algorithm can be used to solve the triangulation problem by finding the L_q-closest-point to a set of lines in 3D. Our experimental results for the Dinosaur set show that in the presence of outliers the proposed L_q method gives superior results to the L_2 method and an L_2 bundle-adjustment algorithm.

A d-dimensional affine subspace of $\mathbb{R}^N$ is a set $S = \{\sum_{i=1}^{d+1} a_i \mathbf{x}_i \mid \sum_{i=1}^{d+1} a_i = 1\}$ for some points $\{\mathbf{x}_i\}$. 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 achieves its 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 $\mathbb{R}^N$.

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.

Weiszfeld Algorithms. The problem of finding the L_q-closest-point to a set of subspaces is closely related to the problem of finding the L_q mean of a set of points in $\mathbb{R}^N$. For

K. Aftab and R. Hartley
College of Engineering and Computer Science, Australian National University and National ICT Australia.
E-mail: khurrum.aftab@anu.edu.au, richard.hartley@anu.edu.au

J. Trumpf
Research School of Engineering, Australian National University.
E-mail: jochen.trumpf@anu.edu.au

points $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k\}$ in some metric space, the L_q-mean is a point for which the sum of the q-th power of distances to the $\mathbf{y}_i$ is minimized. The L_q cost-function in this case is defined as

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

where $d(\cdot,\cdot)$ is a distance function. The problem of finding the L_q mean of a set of points in $\mathbb{R}^N$ goes back to the Fermat-Weber problem, where a point in $\mathbb{R}^N$ is desired for which the sum of distances to a given set of points in a plane is minimized. A classic algorithm that solves for the L_1 mean of a set of points in $\mathbb{R}^N$ is the Weiszfeld algorithm [34]. This is a form of gradient descent with a specifically defined step size, not requiring either line search or computation of gradients. A generalization of the Weiszfeld algorithm to solve for the L_q solution, for $1 \leq q < 2$, of the problem is proposed in [1].

A Weiszfeld-inspired solution strategy to solve for the minimum of the L_q norm of the problem has also been proposed in [5,4,3]. Note that in [5] the sum of L_q norm is minimized, that is $\min_{\mathbf{x}} \sum_{i=1}^{k} w_i \|\mathbf{x} - \mathbf{y}_i\|_q$. That is different from the type of problems solved in [1], or here, where the sum of the q-th power of distances is minimized.

The Weiszfeld algorithm has been generalized to L_1-closest-point problems in Banach spaces [10], rotation space $\mathrm{SO}(3)$ [15,16] and general Riemannian manifolds [12,37].

The problem of finding the L_q-closest-point to affine subspaces and its solution by using the L_q Weiszfeld algorithm must not be confused with the Iterative Re-weighted Least Squares (IRLS) technique in compressed sensing (CS) [8,6,11]. In compressive sensing the problem is to find a point on an affine constraint subspace closest to a union of linear subspaces (defined by all possible permutations of the k non-zero element positions for k-sparse vectors). On the other hand, we have solved quite a different problem of finding a point for which the sum of the q-th power of distances to all the given affine subspaces is the minimum. Furthermore, the convergence of the IRLS algorithms (in compressed sensing) to the L_q solution is only guaranteed when the solution is sparse. This assumption of sparsity limits the applicability of the IRLS algorithms only to the class of problems having sparse solutions. On the contrary, a more general class of problems can be solved using the proposed algorithm. In other words, in our problems of interest it is robustness, and not sparsity, that is the goal.

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 [17,32], corner detection through the intersection of planes [13], etc.

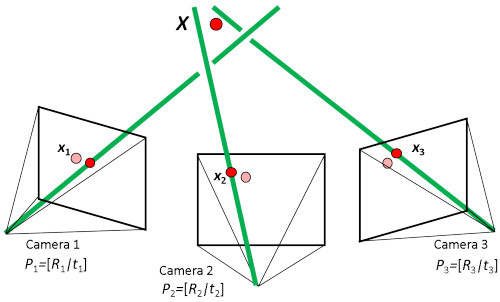


Fig. 1 *Triangulation: Due to noise in image point measurements lines through the center of cameras and* $\mathbf{y}_i$ *do not intersect at point* $\mathbf{X}$.

The problem of finding the L_2-closest-point to subspaces can be solved in closed form. Clearly, the L_2 method is more efficient than the proposed iterative L_q 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. In this paper, we are interested in finding a robust solution of the problem. Thus, we propose an L_q Weiszfeld-inspired [1] approach.

In this paper, we show that the L_q Weiszfeld algorithm [1] 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, for example lines, planes, subspaces or a mixture of these. Just like the L_q Weiszfeld algorithm, the proposed algorithm is an iterative optimization technique where updates are computed analytically. Moreover, the proposed algorithm is simple to understand 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 algorithm inherits all features of the L_q Weiszfeld algorithm.

An important point to note here is that the proposed algorithm finds the L_q minimum even if the given subspaces have different dimensions. For example, the algorithm can be used to find the L_q minimum distance to a set of lines and planes in $\mathbb{R}^N$. In this paper, we give a proof of convergence of the proposed algorithm to the L_q minimum.

A similar problem known as the Heron problem is named after the Heron of Alexandria. The Heron problem is to find a point on a line in a plane where the sum of distances to two given points is minimized. The Heron problem is a special case of the L_q-closest-point to subspaces problem, where the minimum is constrained to be on a subspace (a line). The Heron problem has been generalized to find a point in a closed convex set that minimizes the sum of distances to given convex sets. This problem is referred as the generalized Heron problem and several iterative techniques have been proposed recently in [24,25,7] to solve it.

In order to show the applicability of the proposed algorithm we consider the problem of triangulation [17,18]. 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. 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 algorithm.

One can also find the vertices of the objects in the scenes that have dominant planar structure, for example architectural scenes [29,31,9], indoor scenes [14,26,36], aerial images [2,28], Manhattan world [33,35,13], building reconstruction from laser scanning data [23,19,27] 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 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 than three planes, indicated as red points in fig 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 algorithm 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 paper 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 L_q Weiszfeld algorithm and the rapidity with which its iterative update may be computed makes the proposed method attractive.

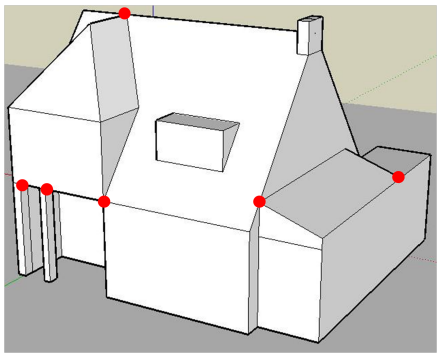


Fig. 2 *Corner point estimation: The object in the figure has a strong geometric structure and corner points, in red, are at the intersection of more than 3 planar faces, though some may actually be coplanar. A segmentation algorithm will find these planar segments individually without the knowledge that they are in fact coplanar. Due to noise, extracted planes representing these faces may be skewed. Since these points are at the intersection of more than 3 planes, these skewed planes may not intersect at a single point. Thus, the problem then is to find an optimal point of intersection of these planes. Image taken from http://russta.wordpress.com/category/sketch-up*

2 L_q Optimization for Points

As mentioned before, the problem of finding the L_q-closest-point, for $1 \leq q < 2$, to a set of affine subspaces is related to the problem of finding the L_q mean of a set of points. In this section we start by reviewing the technique for L_q averaging for points. Given a set of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$ in some metric space, the L_q mean is a point $\bar{\mathbf{x}}$ that minimizes the sum of the q-th power of distances to all given points. Thus,

$$\bar{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}} \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{y}_i)^q \,, \tag{2}$$

where $d(\mathbf{x}, \mathbf{y}_i)$ is the distance between $\mathbf{x}$ and $\mathbf{y}_i$.

Starting from some initial point, the current estimate of the L_q minimum is updated using an update function W, as

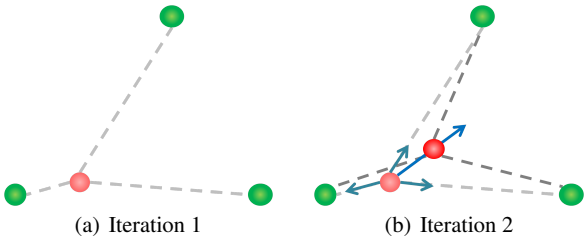
$$\begin{aligned} \mathbf{x}^{t+1} = W(\mathbf{x}^t) &= \frac{\sum_{i=1}^{k} w_i^t \mathbf{y}_i}{\sum_{i=1}^{k} w_i^t} && \text{if } \mathbf{x}^t \notin \{\mathbf{y}_i\} \\ &= \mathbf{y}_j && \text{if } \mathbf{x}^t = \mathbf{y}_j \end{aligned} \,, \tag{3}$$

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

It is shown in [1] that this update function results in a cost decrease, and that starting from a random point $\mathbf{x}^0 \in \mathbb{R}^N$, the sequence of points $(\mathbf{x}^t)$ obtained using (3) will either converge to the L_q minimum, or it will stop at some point with $\mathbf{x}^t = \mathbf{y}_i$.

The algorithm is a form of gradient descent with explicitly defined step size; it may alternatively be seen as an iteratively reweighted least-squares algorithm.

The characteristic of the Weiszfeld algorithm and its generalizations is that they are provably convergent iterative optimization algorithms that do not require computation of gradients or line-search. As such, they are very easy to understand, and code. The iterative update is very quick to compute and in practice, the algorithms are quick to converge.



(a) Iteration 1 (b) Iteration 2

Fig. 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.*

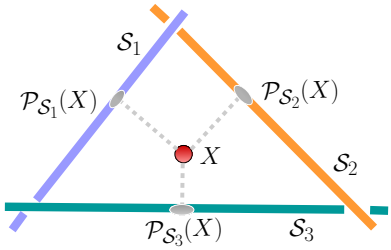


Fig. 4 *L_q-closest-point to Subspaces: The above figure shows three subspaces (lines), $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. We seek a point $\mathbf{X}$ for which the sum of the q-th powers of orthogonal distances is minimum, that is $\sum_i d(\mathbf{X}, \mathcal{S}_i)^q$. Each grey point represents the orthogonal projection $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ of a red point $\mathbf{X}$, onto a corresponding subspace.*

3 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 paper. 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, as shown in fig 4. We refer to this point as the L_q *minimum* or L_q*-closest-point*.

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

$$C_q(\mathbf{X}) = \sum_{i=1}^{k} d(\mathbf{X}, \mathcal{S}_i)^q = \sum_{i=1}^{k} \|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^q \,, \tag{4}$$

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 $\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|$.

The gradient of the C_q cost function is

$$\nabla C_q(\mathbf{X}) = q \sum_{i=1}^{k} \frac{\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})}{\|\mathbf{X} - \mathcal{P}_{\mathcal{S}_i}(\mathbf{X})\|^{2-q}} \,. \tag{5}$$

Orthogonal Projection: We can write $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X})$ as

$$\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}) = \mathbf{C}_i + \mathtt{A}_i \, (\mathbf{X} - \mathbf{C}_i) \,, \tag{6}$$

where $\mathtt{A}_i$ is a matrix representing orthogonal projection 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 projection matrix $\mathtt{A}$ can be computed as $\mathtt{A} = \sum_{i=1}^{d} \mathbf{e}_i \mathbf{e}_i^{\mathsf{T}}$.

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 $\mathtt{A}_i$ is a projection matrix or a projector, and so is $\mathtt{M}_i = (\mathtt{I} - \mathtt{A}_i)$. For a projection matrix $\mathtt{M}_i^2 = \mathtt{M}_i$. Since it is a symmetric matrix we have $\mathtt{M}_i = \mathtt{M}_i^{\mathsf{T}}$. Thus, we have,

$$\mathtt{M}_i^{\mathsf{T}} \mathtt{M}_i = \mathtt{M}_i \mathtt{M}_i = \mathtt{M}_i^2 = \mathtt{M}_i \,. \tag{7}$$

With this notation, the L_q distance cost-function (4) becomes

$$\begin{aligned} C_q(\mathbf{X}) &= \sum_{i=1}^{k} \|\mathtt{M}_i(\mathbf{X} - \mathbf{C}_i)\|^q \\ &= \sum_{i=1}^{k} \left((\mathbf{X} - \mathbf{C}_i)^{\mathsf{T}} \, \mathtt{M}_i \, (\mathbf{X} - \mathbf{C}_i)\right)^{q/2} \,. \end{aligned} \tag{8}$$

A few points are to be noted here. The points $\mathbf{C}_i$ can be chosen as any arbitrary points on the subspaces $\mathcal{S}_i$, and the matrices $\mathtt{M}_i$ do not depend on this choice. The expression (8) for the subspace distance differs from the form of the L_q distance to points $\mathbf{C}_i$ only in the presence of the matrix $\mathtt{M}_i$, which is a matrix of rank r_i, the codimension of the space $\mathcal{S}_i$.

3.1 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 affine subspaces of $\mathbb{R}^N$ and denote by $T(\mathcal{S}_i)$ (the tangent-space of $\mathcal{S}_i$) the set of direction vectors parallel to $\mathcal{S}_i$; otherwise stated, $T(\mathcal{S}_i)$ is the linear space obtained by translating $\mathcal{S}_i$ to the origin.

The L_q cost-function, being a sum of individual (convex) distance functions, is a convex function. Moreover, for $q > 1$, the L_q cost-function is strictly convex, except when there exists a vector that is parallel to all the subspaces. If we assume that such a vector does not exist then the cost-function has a unique global minimum, as the following lemma states.

Lemma 1 *Given a set of affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$, the L_q cost-function (4), for $q > 1$, is strictly convex and*

hence has a single minimum if and only if their tangent spaces $T(\mathcal{S}_i)$ have trivial intersection.

A proof of this lemma and comments regarding the case $q = 1$ are given in section 6.

Assumption: It will be assumed in this paper that the condition of lemma 1 holds. The tangent spaces have trivial intersection and so (4) has a single minimum.

Since for $q > 1$ the L_q cost-function is convex and differentiable, the minimum of the cost function occurs when the gradient vanishes.

For the case $q = 1$, the condition is more complex, since the minimum may occur on one of the subspaces, at a point where the cost function is not differentiable. Similar to what was shown in [34] for the classic Fermat-Weber problem, the minimum of this cost-function may be classified as follows, for the case when the subspaces $\mathcal{S}_i$ are disjoint, as stated here.

Lemma 2 *If the subspaces $\mathcal{S}_i$ are disjoint, a point $\mathbf{X}^* \in \mathbb{R}^N$ is a minimum of the cost-function (4) for $q = 1$ if and only if it satisfies one of the following conditions:*

1. *$\nabla C_1(\mathbf{X})$ vanishes at $\mathbf{X}^*$,*
2. *$\mathbf{X}^* \in \mathcal{S}_j$ and the gradient $\nabla C_1^j(\mathbf{X})$ has norm no greater than 1 and is orthogonal to $\mathcal{S}_j$, where $C_1^j(\mathbf{X}) = \sum_{i \neq j} d(\mathbf{X}, \mathcal{S}_i)$.*

The proof of the above lemma is similar to the case where each S_i is a single point, as given in [34], and is therefore omitted. It illustrates, however, the advantage of $q > 1$, since the identification of the minimum is far simpler.

3.2 Weiszfeld iteration

We wish to find the point $\mathbf{X}$ that minimizes the L_q cost-function (8). There is no closed-form solution to this problem when $q \neq 2$. The basic idea of the Weiszfeld approach is to cast it as an iteratively reweighted least-squared (IRLS) problem, and minimizing by an iterative procedure.

Consequently, given a vector of weights $\mathbf{w}$, we define a weighted L_2 cost-function

$$C_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} w_i \left\| \mathtt{M}_i(\mathbf{X} - \mathbf{C}_i) \right\|^2 . \tag{9}$$

Weiszfeld iteration consists in starting with some initial point $\mathbf{X}^0$ and defining a sequence of iterates $\mathbf{X}^t$, setting weights

$$w_i^t = \left\| \mathtt{M}_i(\mathbf{X}^t - \mathbf{C}_i) \right\|^{q-2} , \tag{10}$$

and then minimizing (9) to give the next estimate

$$\mathbf{X}^{t+1} = \underset{\mathbf{X} \in \mathbb{R}^N}{\operatorname{argmin}} \sum_{i=1}^{k} w_i^t \left\| \, \mathtt{M}_i(\mathbf{X} - \mathbf{C}_i) \, \right\|^2 . \tag{11}$$

The weights $\mathbf{w}_i^t$ are updated at each step of iteration. This is the basic Weiszfeld algorithm for subspaces, which we shall call the IRLS algorithm. Slight refinements of this algorithm will be proposed in the what follows. However, the essential result of this paper is that the algorithm will converge to the optimum from most starting points.

Before considering the issue of convergence, however, it will be shown how each step of the algorithm may be carried out.

3.3 Solving the weighted L_2 problem

Problem (11) has a very simple structure and its solution can be computed in closed form. In fact, since the weights w_i^t are fixed at each step, this is nothing more than a simple linear least-squares problem. Equation (9) can be written more compactly as $C_2^{\mathbf{w}}(\mathbf{X}) = \|\mathtt{M}\mathbf{X} - \mathbf{c}\|^2$, where $\mathtt{M}$ is formed as the stack of the matrices $\sqrt{w_i}\,\mathtt{M}_i$, and $\mathbf{c}$ is the stack of vectors $\sqrt{w_i}\,\mathtt{M}_i\mathbf{C}_i$. The solution is then given using the normal equation method as $\mathbf{X} = (\mathtt{M}^\mathsf{T}\mathtt{M})^{-1}\mathtt{M}^\mathsf{T}\mathbf{c}$. In terms of the individual $\mathtt{M}_i$ and $\mathbf{C}_i$, this solution is

$$\mathbf{X} = \left(\sum_{i=1}^{k} w_i \, \mathtt{M}_i \right)^{-1} \left(\sum_{i=1}^{k} w_i \, \mathtt{M}_i \, \mathbf{C}_i \right) , \tag{12}$$

where we have used the relation $\mathtt{M}_i^\mathsf{T}\mathtt{M}_i = \mathtt{M}_i$ from (7). Thus, the L_2-closest-point to subspaces has a closed form solution.

Numerical stability. If the current estimate $\mathbf{X}^t$ lies close to one of the subspaces $\mathcal{S}_i$, then the weight w_i given by (10) 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 (12) 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 $C_2^{\mathbf{w}} = \| \, \mathtt{M}\mathbf{X} - \mathbf{c} \, \|^2$. Let $\mathtt{M} = \mathtt{U}\mathtt{D}\mathtt{V}^\mathsf{T}$ be the SVD of matrix $\mathtt{M}$ where $\mathtt{D}$ is an invertible diagonal matrix. Then the solution $\mathbf{X}$ is given by

$$\mathbf{X} = \mathtt{V}\,\mathtt{D}^{-1}\mathtt{U}^\mathsf{T}\,\mathbf{c} \, . \tag{13}$$

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 (12). For more on this topic, see Appendix 5 of [18].

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 $q < 2$. For practical purposes, this possibility could perhaps be ignored, as numerically unlikely. It can also be avoided, as shall be seen, by a suitable choice of an initial point $\mathbf{X}^0$. However, for an investigation of convergence, as well as describing a more complete update method, it is necessary to consider the update rule when a point $\mathbf{X}^t$ lies on a subspace. This will be done in the next section.

3.4 L_q cost-function on intersections of subspaces

The cost-function (8) may be restricted to points lying on an intersection of the subspaces $\mathcal{S}_i$. We introduce some new notation. If I is a subset of the integers $1, \ldots, k$ indexing the subspaces $\mathcal{S}_i$, we denote $\cap_{i\in I}\,\mathcal{S}_i$ by S_I. The intersection may, of course, be empty. Further, define $\mathcal{S}_I^\circ$ by

$$\mathcal{S}_I^\circ = \mathcal{S}_I \setminus \bigcup_{i\notin I} \mathcal{S}_i \ ,$$

the set of points that lie in $\mathcal{S}_I$ but not the intersection of any larger set of subspaces.

For consistency, if $I = \emptyset$, then $\mathcal{S}_I$ is defined as the whole of $\mathbb{R}^N$ and $\mathcal{S}_I^\circ$ is the complement of the union of subspaces. Note that a point $\mathbf{X}$ lies in exactly one non-empty set $\mathcal{S}_I^\circ$. Since every point lies on some maximal set (maybe empty) of subspaces, $\mathcal{S}_i$, the $\mathcal{S}_I^\circ$ form a decomposition of $\mathbb{R}^N$ into disjoint sets.

If $\mathbf{X} \in \mathcal{S}_I$, then its distance to any $\mathcal{S}_j$ with $j \in I$ vanishes, and the L_q cost-function takes the form

$$C_q^I(\mathbf{X}) = \sum_{i\notin I} d(\mathbf{X}, \mathcal{S}_i)^q \ , \tag{14}$$

for points $\mathbf{X} \in \mathcal{S}_I$. This cost-function may be minimized to find the point of smallest cost lying in the subspace $\mathcal{S}_I$. The solution to this problem can also not be solved in closed form, unless $q = 2$. To solve it one makes use of a weighted L_2 cost-function

$$C_2^{I\mathbf{w}}(\mathbf{X}) = \sum_{i\notin I} w_i\, d(\mathbf{X}, \mathcal{S}_i)^2 \ . \tag{15}$$

This leads to an update procedure that can be applied at any point $\mathbf{X}$ in $\mathbb{R}^N$ as follows. If $\mathbf{X}^t \in \mathcal{S}_I^\circ$, set

$$\begin{aligned} w_i^t &= d(\mathbf{X}^t, \mathcal{S}_i)^{q-2} \ \text{ for } i \notin I, \\ \mathbf{X}^{t+1} &= \operatorname*{argmin}_{\mathbf{X}\in\mathcal{S}_I} \sum_{i\notin I} w_i^t\, d(\mathbf{X}, \mathcal{S}_i)^2 \ . \end{aligned} \tag{16}$$

This is the update rule that will be used in the rest of the paper. The mapping $\mathbf{X}^t \mapsto \mathbf{X}^{t+1}$ defined here will be referred to as an update function W defined such that $\mathbf{X}^{t+1} = W(\mathbf{X}^t)$. In the case when $\mathbf{X}^t$ does not lie on any of the subspaces $\mathcal{S}_j$, it corresponds exactly with the update rule given by (10) and (11).

Continuity: An important point for investigating the convergence of the sequence of updates is the way the update mappings fit together for the different spaces $\mathcal{S}_I^\circ$.

Lemma 3 *The update mapping W defined in (16) is continuous on $\mathbb{R}^N$.*

Proof. It is assumed (the blanket assumption in this paper) that there is no direction tangential to all the S_i. Consequently, the cost function (4) is strictly convex, and all the argmin operations in this proof yield a single point. Moreover, the function $W(\mathbf{X})$ defined above is well-defined.

Consider an arbitrary sequence of points $\mathbf{X}_j$ converging to a point $\mathbf{X}^*$. Continuity of W will follow from proving that $W(\mathbf{X}_j)$ converges to $W(\mathbf{X}^*)$.

The sequence can be broken up into subsequences, each one consisting of those points in some fixed $\mathcal{S}_J^\circ$, where J is a subset of $\{1, 2, \ldots, k\}$. It will be sufficient to show that $W(\mathbf{X}_j)$ converges to $W(\mathbf{X}^*)$ for $\mathbf{X}_j$ in each infinite subsequence separately. Thus, we assume that all $\mathbf{X}_j$ lie in some fixed $\mathcal{S}_J^\circ$ and converge to a point $\mathbf{X}^* \in \mathcal{S}_I^\circ$ for some $\mathcal{S}_I^\circ$, necessarily satisfying $J \subset I$ and hence $\mathcal{S}_I \subset \mathcal{S}_J$. If $I = J$, the required result is immediate, since the update mapping is continuous at points in S_J°.

Since $\mathbf{X}_j$ converges to $\mathcal{S}_I$ the distances $d(\mathbf{X}_j, \mathcal{S}_i)$ converge to zero for all $i \in I$. We break up the sequence further into subsequences indexed by $i \in I\setminus J$, such that a point $\mathbf{X}_j$ is assigned to subsequence i if $d(\mathbf{X}_j, \mathcal{S}_i)$ is the maximum of these distances over all $i \in I\setminus J$. Once again, each such subsequence, if infinite, must converge to $\mathbf{X}^*$, and again it is sufficient to consider each one separately.

Without loss of generality, therefore, we assume that $J \subsetneq I$, that $\mathbf{X}_j$ is a sequence of points in S_J° converging to a point $\mathbf{X}^*$ in S_I° and that there is an index (chosen without loss of generality to be 1) in $I\setminus J$, such that $d(\mathbf{X}_j, \mathcal{S}_1) \geq d(\mathbf{X}_j, \mathcal{S}_i)$ for all $i \in I\setminus J$. The update $\mathbf{Y}_j = W(\mathbf{X}_j)$ is defined by

$$\mathbf{Y}_j = \operatorname*{argmin}_{\mathbf{X}\in\mathcal{S}_J} \sum_{i\notin J} w_{ij}\, d(\mathbf{X}, \mathcal{S}_i)^2 \tag{17}$$

with $w_{ij} = d(\mathbf{X}_j, \mathcal{S}_i)^{q-2}$. Continuity is proved by showing that $\mathbf{Y}_j$ converges to $\mathbf{Y}^* = W(\mathbf{X}^*)$, given by

$$\mathbf{Y}^* = \operatorname*{argmin}_{\mathbf{X}\in\mathcal{S}_I} \sum_{i\notin I} w_i^*\, d(\mathbf{X}, \mathcal{S}_i)^2 \tag{18}$$

where $w_i^* = d(\mathbf{X}^*, \mathcal{S}_i)^{q-2}$. Define the function $f(\mathbf{X}) = \sum_{i\notin I} w_i^*\, d(\mathbf{X}, \mathcal{S}_i)^2$, and observe that f is continuous everywhere and $\mathbf{Y}^* = \operatorname{argmin}_{\mathbf{X}\in S_I} f(\mathbf{X})$.

The first step is to show that the points $\mathbf{Y}_j$ converge to the set $\mathcal{S}_I$. Observe that the weights w_{1j} increase to infinity as $\mathbf{X}_j$ approaches $\mathcal{S}_1$. At the same time, weights w_{ij} for $i \notin I$ converge to $d(\mathbf{X}^*, \mathcal{S}_i)^{q-2}$, and hence remain bounded. We define weights $w'_{ij} = w_{ij}/w_{1j}$ and observe that $w'_{ij} \geq 1$ for $i \in I \backslash J$, whereas $\lim_{j\to\infty} w'_{ij} = 0$ for $i \notin I$. In addition, weights w_{ij} may be replaced in (17) by w'_{ij} without changing $\mathbf{Y}_j$. From (17) it follows that for any $\mathbf{X}^* \in \mathcal{S}_I$,

$$\begin{aligned}\sum_{i\in I\backslash J} w'_{ij}\, d(\mathbf{Y}_j, \mathcal{S}_i)^2 + \sum_{i\notin I} w'_{ij}\, d(\mathbf{Y}_j, \mathcal{S}_i)^2 \\ \leq \sum_{i\in I\backslash J} w'_{ij}\, d(\mathbf{X}^*, \mathcal{S}_i)^2 + \sum_{i\notin I} w'_{ij}\, d(\mathbf{X}^*, \mathcal{S}_i)^2 \\ = \sum_{i\notin I} w'_{ij}\, d(\mathbf{X}^*, \mathcal{S}_i)^2\end{aligned} \tag{19}$$

since $d(\mathbf{X}^*, \mathcal{S}_i) = 0$ for $i \in I$. This sum converges to zero, since w'_{ij} converges to zero for $i \notin I$. It follows that $\sum_{i\in I\backslash J} w'_{ij}\, d(\mathbf{Y}_j, \mathcal{S}_i)^2$ converges to zero.

Next, let $\eta_j = \sum_{i\in I\backslash J} w'_{ij}\, d(\mathbf{Y}_j, \mathcal{S}_i)^2$ and define the set

$$T_j = \{\mathbf{X} \in \mathcal{S}_J \mid \sum_{i\in I\backslash J} w'_{ij}\, d(\mathbf{X}, \mathcal{S}_i)^2 = \eta_j\}\,. \tag{20}$$

Since $w'_{ij} \geq 1$ and $\eta_j \to 0$, the T_j converge to S_I in the sense that the sets get closer and closer to S_I, and furthermore, *any* point in S_I is approached arbitrarily closely by points in the sequence of sets T_j. Furthermore, since $\mathbf{Y}_j$ lies in T_j, it follows that the $\mathbf{Y}_j$ converge to the set $\mathcal{S}_I$. However, the location of $\mathbf{Y}_j$ in T_j can be more explicitly determined by observing that

$$\begin{aligned}\mathbf{Y}_j &= \operatorname*{argmin}_{\mathbf{X}\in T_j} \sum_{i\in I\backslash J} w'_{ij}\, d(\mathbf{X}, \mathcal{S}_i)^2 + \sum_{i\notin I} w'_{ij}\, d(\mathbf{X}, \mathcal{S}_i)^2 \\ &= \operatorname*{argmin}_{\mathbf{X}\in T_j} \sum_{i\notin I} w_{ij}\, d(\mathbf{X}, \mathcal{S}_i)^2\end{aligned} \tag{21}$$

since the first sum is a constant (η_j) for points $\mathbf{X} \in T_j$ and we may replace the w'_{ij} by w_{ij} in the second line. Now we define further points

$$\mathbf{Y}'_j = \operatorname*{argmin}_{\mathbf{X}\in T_j} \sum_{i\notin I} w^*_i\, d(\mathbf{X}, \mathcal{S}_i)^2 = \operatorname*{argmin}_{\mathbf{X}\in T_j} f(\mathbf{X}) \tag{22}$$

and observe that since the weights w_{ij} converge to w^*_i for each $i \notin I$, by continuity of the update with respect to the weights, $\lim_{j\to\infty} \mathbf{Y}'_j = \lim_{j\to\infty} \mathbf{Y}_j$.

Now define further points $\mathbf{Y}''_j \in T_j$ such that $\mathbf{Y}''_j$ converges to $\mathbf{Y}^*$. Then, by definition of $\mathbf{Y}'_j$ we have $f(\mathbf{Y}'_j) \leq f(\mathbf{Y}''_j)$. Therefore

$$f(\mathbf{Y}^*) = \lim_{j\to\infty} f(\mathbf{Y}''_j) \geq \lim_{j\to\infty} f(\mathbf{Y}'_j) = f(\lim_{j\to\infty} \mathbf{Y}_j)\,.$$

However, since $\mathbf{Y}^*$ and $\lim_{j\to\infty} \mathbf{Y}_j$ are in S_I, by the definition of $\mathbf{Y}^*$ it follows that $f(\lim_{j\to\infty} \mathbf{Y}_j) \geq f(\mathbf{Y}^*)$, so

$$f(\lim_{j\to\infty} \mathbf{Y}_j) = f(\mathbf{Y}^*) = \min_{\mathbf{X}\in S_I} f(\mathbf{X})\,.$$

However, by the uniqueness of the minimum (the definition of $W(\mathbf{X}^*)$) it follows finally that $\mathbf{Y}^* = \lim_{j\to\infty} \mathbf{Y}_j$. This completes the proof. □

Computing the update: When the point $\mathbf{X}^t$ lies in $\mathcal{S}_I^\circ$, the minimum in the update step (16) is taken over points $\mathbf{X}$ in $\mathcal{S}_I$. As has been seen, this is a linear least-squares problem, but in this case, the minimum is to be restricted to an affine subspace. It is no surprise that the solution to this problem my be computed directly and non-iteratively. One way to do this is to cast the problem using Lagrange multipliers, in which case a closed-form solution can be found. However, for numerical reasons, a solution using SVD is preferred, as follows.

Guided by the approach of section 3.3 the problem may be cast in the following way.

$$\textbf{Minimize}\ \ \|\mathtt{M}\mathbf{X} - \mathbf{c}\| \tag{23}$$

$$\textbf{subject to}\ \ \mathtt{M}_I(\mathbf{X} - \mathbf{C}_I) = \mathbf{0}\,, \tag{24}$$

where $\mathtt{M}_I$ is the stack of the individual $\mathtt{M}_j$ corresponding to the subspaces $\mathcal{S}_j$ intersecting in $\mathcal{S}_I$, and $\mathbf{C}_I$ is a point on $\mathcal{S}_I$.

Let the SVD of $\mathtt{M}_I$ be $\mathtt{M}_I = \mathtt{U}\mathtt{D}\mathtt{V}^\mathsf{T}$, where $\mathtt{D}$ has rank r, the codimension of the subspace $\mathcal{S}_I$. The condition $\mathtt{M}_I(\mathbf{X} - \mathbf{C}_I) = \mathbf{0}$ is equivalent (after cancelling $\mathtt{U}$) to $\mathtt{D}\mathtt{V}^\mathsf{T}\mathbf{X} = \mathtt{D}\mathtt{V}^\mathsf{T}\mathbf{C}_I$.

Now, write $\mathbf{X}' = \mathtt{V}^\mathsf{T}\mathbf{X}$ and $\mathbf{C}'_I = \mathtt{V}^\mathsf{T}\mathbf{C}_I$. Further, define $\mathtt{M}' = \mathtt{M}\mathtt{V}$, so that $\mathtt{M}\mathbf{X} = \mathtt{M}'\mathbf{X}'$. The optimization problem then becomes

$$\textbf{Minimize}\ \ \|\mathtt{M}'\mathbf{X}' - \mathbf{c}\| \tag{25}$$

$$\textbf{subject to}\ \ \mathtt{D}\mathbf{X}' = \mathtt{D}\mathbf{C}'_I\,. \tag{26}$$

The condition $\mathtt{D}\mathbf{X}' = \mathtt{D}\mathbf{C}'_I$ means that $\mathbf{X}'$ and $\mathbf{C}'_I$ agree in their first r components. Hence, $\mathbf{X}'$ can be written as $\mathbf{X}' = (\mathbf{C}'; \widetilde{\mathbf{X}})$ (the vector obtained by stacking $\mathbf{C}'$ and $\widetilde{\mathbf{X}}$), where $\mathbf{C}'$ consists of the first r components of $\mathbf{C}'_I$ and $\widetilde{\mathbf{X}}$ is at present unknown.

Let $\mathtt{M}'_L$ represent the first r columns of $\mathtt{M}'$ and $\mathtt{M}'_R$ the remaining columns. Then, the optimization problem becomes

$$\textbf{minimize}\ \ \|\mathtt{M}'_R\widetilde{\mathbf{X}} + \mathtt{M}'_L\mathbf{C}' - \mathbf{c}\|\,, \tag{27}$$

where the minimization is to take place over all choices of $\widetilde{\mathbf{X}}$ This is now a linear least-squares problem, which can be solved for $\widetilde{\mathbf{X}}$ using the SVD method given in section 3.3. Once $\widetilde{\mathbf{X}}$ is found in this way, $\mathbf{X}'$ is calculated as $\mathbf{X}' = (\mathbf{C}'; \widetilde{\mathbf{X}})$. Finally, $\mathbf{X}$ is given by $\mathbf{X} = \mathtt{V}\mathbf{X}'$, and the problem is solved.

4 Convergence

In this section, the following important questions will be answered: does the sequence of iterates given by (16) converge, and if so, what does it converge to? One hopes that the sequence will converge to the minimum of the L_q cost-function (8). It will be seen that this is *almost* true, but it is possible also for the sequence to converge to some other points, as will be seen.

As has been remarked, the algorithm that generates this sequence of iterates is an example of iterative reweighted least squares (IRLS). The following theorem shows that for a wide class of L_q minimization problems, the IRLS step results in a decrease in the cost-function.

Theorem 1 *For $i = 1, \ldots, k$ let $f_i : D \to \mathbb{R}^+$ be a positive valued function defined on an arbitrary domain D and let $\mathbf{X}^t$ and $\mathbf{X}^{t+1}$ be two points in D. Let $0 < q < n$ and define $w_i^t = f_i(\mathbf{X}^t)^{q-n}$. If*

$$\sum_{i=1}^{k} w_i f_i(\mathbf{X}^{t+1})^n \leq \sum_{i=1}^{k} w_i f_i(\mathbf{X}^t)^n \ ,$$

then

$$\sum_{i=1}^{k} f_i(\mathbf{X}^{t+1})^q \leq \sum_{i=1}^{k} f_i(\mathbf{X}^t)^q \ ,$$

with equality if and only if $f_i(\mathbf{X}^{t+1}) = f_i(\mathbf{X}^t)$ for all i.

This theorem was proved in [1]. The most important case for our purposes is where $n = 2$ and $1 \leq q < 2$. Then, the theorem states that if the transition $\mathbf{X}^t \mapsto \mathbf{X}^{t+1}$ results in a decrease of the weighted squared cost, then the L_q cost will also be decreased. In other words a sequence of IRLS iterations applied to an L_q cost-function of this very general type will always result in a decrease of cost at each iteration, unless it hits a fixed point. In particular, when (as in this paper) $\mathbf{X}^{t+1}$ is chosen as the minimizer of the weighted L_2 cost, then it will always result in a decrease in the L_q cost, unless $f_i(\mathbf{X}^{t+1}) = f_i(\mathbf{X}^t)$ for all i, in which case $\mathbf{X}^t$ already minimizes the weighted cost-function. Applied to the current subspace problem, this is true both for the case where the point $\mathbf{X}^t$ lies on a subspace $\mathcal{S}_i$ or not, according to the update rule (16).

So the Weiszfeld (IRLS) iteration always decreases the cost. It does not follow that the sequence of iterations converges, or that it converges to a point that minimizes the L_q cost. To prove convergence, we make use of the following theorem.

Theorem 2 *Let D be a compact metric space and $C : D \to \mathbb{R}$ be a continuous function defined on D. Let $W : D \to D$ be a continuous function with the property $C(W(\mathbf{x})) < C(\mathbf{x})$ unless $W(\mathbf{x}) = \mathbf{x}$.*

Let $\mathbf{x}^0 \in D$, and $\mathbf{x}^k = W^k(\mathbf{x})$ for $k = 1, 2, \ldots$. Then the sequence $(\mathbf{x}^k)$ converges to $S = \{\mathbf{x} \mid W(\mathbf{x}) = \mathbf{x}\}$, 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 $k > N$.

If in addition S is a finite or countable set, then the sequence $(\mathbf{x}^k)$ converges (to a point in S).

Informally stated, if W is a continuous update function with fixed point set S, strictly decreasing a continuous cost-function $C(\mathbf{x})$ (except at fixed points $\mathbf{x} \in S$), then the sequence of iterates generated by W will converge to S, and moreover to a single fixed point if S is countable or finite. This is a minimal set of conditions for convergence. If D is not compact, W or C is not continuous, or S is uncountable, then there exist counterexamples where convergence does not hold.

The first part of the theorem also follows from the well known Global Convergence Theorem [22, section 6.6] applied to the special case of a single valued, continuous algorithm map. Theorem 2 gives a simple but widely useful conditions for convergence of a descent algorithm. For a slightly more general statement and detailed proofs see [1].

To prove convergence of the set of iterates generated by the update rule (16) it is enough to check the conditions of Theorem 2.

1. **Continuity.** The cost-function (8) is continuous by inspection. The update function defined by (16) is continuous, according to lemma 3.
2. **Decreasing cost.** The update step always decreases the cost, unless the point $\mathbf{X}^t$ already represents a minimum of the weighted L_2 cost function, according to Theorem 1.
3. **Compactness.** The cost and update functions are defined on the whole of R^N, which is not a compact set. However, if $\mathbf{X}^0$ is a starting point and D is defined as $\{\mathbf{X} \in \mathbb{R}^N \mid C_q(\mathbf{X}) \leq C_q(\mathbf{X}^0)\}$, then D is closed, since C_q is continuous. The update function W maps D into D, since it results in non-increasing cost. Finally, since by assumption there is no direction vector that is parallel to all subspaces, C_q is bounded since $C_q(\mathbf{X})$ tends to infinity as $\mathbf{X}$ does. Hence, the set D is closed and bounded, and hence compact.

According to Theorem 2 the sequence of updates $W : \mathbf{X}^t \mapsto \mathbf{X}^{t+1}$ given by (16) will converge from any initial point $\mathbf{X}^0$ to the set of fixed points of W. It is therefore appropriate to determine this fixed-point set. The set of fixed points of W contains one point of every non-empty intersection of $\mathcal{S}_I$, namely the minimum of C_q restricted to $\mathcal{S}_I$. By assumption there is no direction vector that is parallel to subspaces therefore the L_q cost function is strictly convex. Since C_q is strictly convex, there is exactly one minimum of C_q restricted to each $\mathcal{S}_I$ including a minimum of the unrestricted C_q on the whole of $\mathbb{R}^N$, (which may be one of the

minima lying on a subspace $\mathcal{S}_I$). There are therefore a finite number of fixed points of W, and hence, according to Theorem 2 the sequence will converge to one of these points.

Fixed points of W. Determination of the fixed points of W will rely on the following fact, easily verified by calculation: if weights w_i are defined at a fixed point $\mathbf{X}^*$ of W by $w_i = d(\mathbf{X}^*, \mathcal{S}_i)^{q-2}$, then

$$\nabla C_2^{\mathbf{w}}(\mathbf{X}^*) = K\, \nabla C_q(\mathbf{X}^*)\ ,$$

where K is a constant (equal to $q/2$). Here, $\nabla C(\mathbf{X}^*)$ denotes the gradient of C evaluated at $\mathbf{X}^*$. Now, suppose that $W(\mathbf{X}^*) = \mathbf{X}^*$ and $\mathbf{X}^*$ lies on $\mathcal{S}_I$. Recall the definitions of C_q^I from (14) and $C_2^{I\mathbf{w}}$ from (15). Then

$$\begin{aligned} W(\mathbf{X}^*) = \mathbf{X}^* &\implies \mathbf{X}^* \text{ is the minimum of } C_2^{I\mathbf{w}} \text{ on } \mathcal{S}_I \\ &\implies \nabla C_2^{I\mathbf{w}}(\mathbf{X}^*) \text{ is perpendicular } (\perp) \text{ to } \mathcal{S}_I \\ &\implies \nabla C_q^I(\mathbf{X}^*) \perp \mathcal{S}_I \\ &\implies \mathbf{X}^* \text{ is the minimum of } C_q^I \text{ on } \mathcal{S}_I \\ &\implies \mathbf{X}^* \text{ is the minimum of } C_q \text{ on } \mathcal{S}_I\ . \end{aligned}$$

In the case when $I = \emptyset$ and hence $\mathcal{S}_I = \mathbb{R}^N$, the condition that $\nabla C(\mathbf{X}) \perp \mathcal{S}_I$ is to be interpreted as meaning that $\nabla C(\mathbf{X}) = 0$, and the proof holds equally well in this case.

This result completes the proof that the set of iterates defined by (16) will converge to the minimum of C_q on one of the subspaces $\mathcal{S}_I$ from an arbitrary starting point $\mathbf{X}^0$. This is not exactly what is required; we would like the set of iterates to converge to the minimum of the cost-function C_q. It is, in fact, possible for a set of iterates to converge to a point on one of the subspaces which is not the L_q minimum. If one of the iterates $\mathbf{X}^t$ lands on one of the subspaces $\mathcal{S}_i$, then one may verify from the update rule (16) that further iterations will remain on that subspace. This is the common issue with the Weiszfeld algorithm. In reality, this eventuality is not likely to happen, but it must be taken into account if the algorithm is to be proved to converge to the optimal point.

In the next section, two strategies will be discussed to make sure that the iterates converge to the L_q minimum.

4.1 Algorithm

Given a set of affine subspaces $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\} \in \mathbb{R}^N$, $k \geq 2$, we consider algorithms to find the L_q closest point. If there is no direction vector parallel to all the subspaces, the cost-function is strictly convex (for the case $q > 1$) and so has a single minimum. If the cost-function is not strictly convex, then the optimization can take place in a subspace perpendicular to this degenerate direction. Hence, we may assume a single minimum of the cost-function.

The overall algorithm in simplest terms is as follows.

Algorithm 1. *Practical algorithm (Weiszfeld-IRLS).*

1. *Start from an initial point $\mathbf{X}^0$ not lying on any of the subspaces $\mathcal{S}_i$.*
2. *Compute iterates $\mathbf{X}^{t+1}$ according to the rule (16) until convergence.*

As a practical algorithm this is usually adequate, since the likelihood of one of the iterates landing on a subspace $\mathcal{S}_i$ (in which case subsequent iterates will stay on $\mathcal{S}_i$) is slight.

Gradient descent: The algorithm may be modified to guarantee convergence to the optimum point by adding a gradient-descent step. The gradient of the C_q cost function is given by (5). If the point $\mathbf{X}$ is on one of the subspaces $\mathcal{S}_I$, then the gradient of $C_q^I(\mathbf{X})$ should be used. By a line search in the downhill gradient direction (if the gradient is non-zero), with backtracking if necessary, one can find a step resulting in a decrease in cost.

This motivates a modification to the basic algorithm, as follows. We suppose that $q > 0$ so that the gradient is defined on any $\mathcal{S}_i$.

Algorithm 2. *Weiszfeld with gradient descent.*

1. *Apply algorithm (1).*
2. *If the limit lies on a subspace $\mathcal{S}_I$, but the cost is not minimized (gradient is nonzero), apply one step of gradient descent with backtracking [1], and restart algorithm (1) from the new point.*

This algorithm must, after a finite number of restarts, converge to the L_q optimum for the following reason. Algorithm (1) will converge to the point with minimum C_q cost on a subspace $\mathcal{S}_I$. After gradient descent the cost will be less than the cost at this point. Since further iterations decrease the cost, the algorithm can never converge to the same point again. Since there are are a finite number of S_I, the algorithm will eventually reach the global minimum.

Initialization: A final version of the algorithm relies on a smarter initialization to avoid convergence to any of the subspaces $\mathcal{S}_i$. This algorithm is convenient if each of the subspaces $\mathcal{S}_i$ are disjoint; otherwise the cost of the initialization may not be justified. Observe that by starting the algorithm with a point $\mathbf{X}^0$ on one of the subspaces $\mathcal{S}_i$ successive iterations must remain on $\mathcal{S}_i$, and the algorithm will converge to the minimum on $\mathcal{S}_i$ (assuming they are disjoint).

This suggests the following algorithm.

Algorithm 3. *Weiszfeld with initialization.*

1. *By starting at a point $\mathbf{X}^0$ on each subspace $\mathcal{S}_j$ in turn find the minimum cost on each subspace $\mathcal{S}_j$, achieved at a point $\mathbf{X}_j^*$.*

[1] Gradient descent with backtracking means head in the descent downhill gradient direction. If this does not result in a decreased cost, then back up by making a smaller step, until the cost decreases.

2. *Among the $\mathbf{X}_j^*$ select the one with minimum C_q cost and test whether it represents a minimum of the C_q cost-function; if so, then stop.*
3. *If not at a minimum, perform one step of gradient descent with backtracking to find a point $\mathbf{X}^{00}$ with lesser cost.*
4. *Run algorithm 1 from the point $\mathbf{X}^{00}$ until convergence.*

Since the final run of algorithm 1 starts at a point with cost less than that at any point on any of the subspaces, the only possible convergence point is the global optimum of the cost function. Thus, this algorithm is guaranteed to find this global optimum.

Practicalities: In reality, it is unnecessary to use the initialization procedure of algorithm 3, which is given only to supply a provably convergent algorithm. Instead, one may start with an arbitrary point $\mathbf{X}^0$, as is standard in the original Weiszfeld algorithm. The likelihood of meeting one of the subspaces is small. If iterations appear to converge towards a given subspace $\mathcal{S}_I$, then one may apply a step of gradient descent as in algorithm 2.

An alternative approach that has been suggested in the literature for other problems is to use a heuristic by adding a small value ϵ to the distance function so that the weight $w_i = d_i^{q-2}(\cdot,\cdot)$ is defined (and finite) everywhere. It is not, however, clear exactly what cost-function is then minimized, and questions of convergence are obscured.

5 Gradient-descent Weiszfeld

The original Weiszfeld algorithm [34] for finding the closest point to points in $\mathbb{R}^N$ can be regarded either as an IRLS algorithm, or alternatively as a gradient-descent algorithm with specified step size. The algorithm given here, although an IRLS algorithm, is not a gradient descent algorithm, since each update, although decreasing, is not in the gradient direction.

One may propose a slightly different Weiszfeld-style algorithm in the present subspace problem that does make an update in the gradient direction at each step. Instead of using the weighted cost-function (9), equal to

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

the weighted cost

$$\widetilde{C}_2^{\mathbf{w}}(\mathbf{X}) = \sum_{i=1}^{k} w_i \left\| d(\mathbf{X}, \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)) \right\|^2 , \tag{28}$$

is used. In this case, the update step minimizes the distance to the points $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$, the closest points to $\mathbf{X}^t$ on the subspaces from the previous iteration, as shown in fig 5. This is

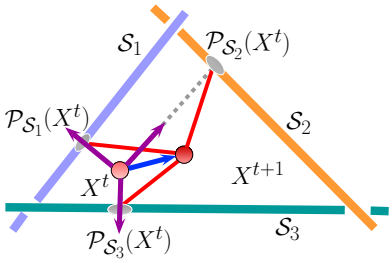


Fig. 5 *L_q-closest-point to Subspaces (Gradient-descent Approach): The above figure shows three subspaces (lines) $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$; and and a current estimate of the L_q minimum $\mathbf{X}^t$. In the gradient descent approach an updated point $\mathbf{S}^{t+1}$ is computed by keeping the projections $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$ fixed.*

then identical to the update step in the original Weiszfeld algorithm, [34] except that the points $\mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)$ change at each iteration. The weights are the same as before, in (16) but the update step is given instead by

$$\mathbf{X}^{t+1} = \frac{\sum_{i=1}^{k} w_i \mathcal{P}_{\mathcal{S}_i}(\mathbf{X}^t)}{\sum_{i=1}^{k} w_i} . \tag{29}$$

Then $\mathbf{X}^{t+1}$ is the point that minimizes the weighted L_2 cost (28). Comparison of this update formula with the gradient of C_q, given in (5) shows that the update is in the downhill gradient direction of the cost function.

The same analysis as before shows convergence of the iterates to the fixed points of W. However, there is a significant difference. As before, the update given by (29) is not defined for $\mathbf{X}^t$ on a subspace $\mathcal{S}_j$. If the update function W is extended by continuity to the subspaces $\mathcal{S}_j$, it can be verified that the update is defined by $\mathbf{X}^{t+1} = \mathbf{X}^t$ when $\mathbf{X}^t$ is on a subspace $\mathcal{S}_j$.

The consequence of this is that the fixed point set of the update function consists of the global cost minimum $\mathbf{X}^*$, plus all points on all the subspaces $\mathcal{S}_j$. This is an uncountable set. Therefore, one cannot conclude, using Theorem 2 that the sequence converges to a point, and furthermore the convergence set is large.

Nevertheless, with appropriate initialization as in algorithm 3 the sequence of iterates is guaranteed to converge to the optimum of the C_q cost-function. This is the method for our final algorithm.

Algorithm 4. *Weiszfeld gradient-descent.*

1. *Find an initial point as in algorithm 3.*
2. *Compute iterates $\mathbf{X}^{t+1}$ according to the rule (29) until convergence.*

The algorithm is guaranteed to find the global minimum of C_q since convergence to any of the subspaces cannot occur, and the only possible convergence point is the global minimum.

6 Convexity and uniqueness of solution

In this section we consider the convexity of the L_q cost-function (4). This analysis was deferred from earlier in the paper.

A function is convex if its Hessian is positive semi-definite and strictly convex if the Hessian is positive-definite. We begin by considering the distance of a point $\mathbf{X}$ from a single subspace $\mathcal{S}_i$. For simplicity, in terms of a coordinate system $(x_1, \ldots, x_N)$ for $\mathbb{R}^N$, let the subspace $\mathcal{S}_i$ be defined by $x_1 = x_2 = \ldots = x_r = 0$, where $r = N - d_i$ is the codimension of subspace $\mathcal{S}_i$. The term of the L_q cost function for that subspace is given by

$$d(\mathbf{X}, \mathcal{S})^q = \left(\sqrt{x_1^2 + x_2^2 + \ldots + x_r^2} \right)^q . \tag{30}$$

Taking the Hessian of this term, and evaluating at the point $\mathbf{X} = (d, 0, \ldots, 0)$ at distance d from S gives, by a simple computation,

$$\mathtt{H}_i = q\, d^{q-2}\, \mathrm{diag}(q-1, 1, \ldots, 1, 0, \ldots, 0) ,$$

where there are $r - 1$ entries equal to 1. This represents the general case, since any subspace and point can be represented in this way in a suitable local rectangular coordinate system. For $1 \leq q \leq 2$, the Hessian is positive semi-definite. Note that there is a difference between the cases $q = 1$ and $q > 1$.

Case $q > 1$. The case $q > 1$ is simpler; the null space of the Hessian is independent of the point $\mathbf{X}$ where the Hessian is evaluated. It is the tangent space of $\mathcal{S}_i$, namely the linear subspace $T(\mathcal{S}_i)$ of $\mathbb{R}^N$, passing through the origin, parallel to $\mathcal{S}_i$.

Now, given several subspaces $\mathcal{S}_i$, the Hessian of the L_q cost-function is the sum of the individual Hessians, and consequently, the null-space is equal to the intersection of the tangent spaces $T(\mathcal{S}_i)$. This gives the result

Lemma 4 *The cost-function (4) is strictly convex for $q > 1$ if and only if the tangent spaces $T(\mathcal{S}_i)$ have trivial intersection.*

This condition can be alternatively expressed by saying that the cost is strictly convex unless there is a direction vector lying parallel to all the subspaces $\mathcal{S}_i$. Generically, the cost-function is strictly positive-definite as long as the codimensions of the subspaces $\mathcal{S}_i$ sum to at least N, thus

$$\sum_{i=1}^{k} r_i \geq N . \tag{31}$$

Note that if $\mathbf{V}$ is such a direction and $\mathbf{X}^*$ is a global minimum of (4) then so is $\mathbf{X}^* + \lambda\mathbf{V}$, since the cost does not vary in the direction $\mathbf{V}$. Therefore, the trivial intersection of the tangent spaces $T(\mathcal{S}_i)$ in lemma 4 is a necessary and sufficient condition for there to be a single minimum of the L_q cost when $q > 1$.

This observation also suggests a procedure to apply in the contrary case, when there does exist a direction $\mathbf{V}$ parallel to all the $\mathcal{S}_i$. By selecting a hyperplane $S^\perp$ perpendicular to the direction $\mathbf{V}$, taking a slice along $S^\perp$ and replacing each $\mathcal{S}_i$ by $\mathcal{S}_i \cap S^\perp$, drops the dimension by 1. Solution of the new reduced problem yields one of a family of solutions to the original problem.

Case $q = 1$. In the case $q = 1$ the null-space of the Hessian $\mathtt{H}_i$ has one extra dimension, which varies as $\mathbf{X}$ moves. The extra dimension consists of the vector from $\mathbf{X}$ normal to the subspace $\mathcal{S}_i$. Geometrically, the null-space of the Hessian at the point $\mathbf{X}$ is the space spanned by vectors parallel to $\mathcal{S}_i$, plus the normal direction from $\mathbf{X}$ to $\mathcal{S}_i$. Otherwise stated, this is the set of all direction vectors through $\mathbf{X}$ that meet the subspace $\mathcal{S}_i$ (including at infinity).

For several subspaces $\mathcal{S}_i$ the Hessian will be non-definite at $\mathbf{X}$ (have non-trivial null-space) exactly when there exists a line through $\mathbf{X}$ that meets all the subspaces $\mathcal{S}_i$.

As an example, for four lines $\mathcal{S}_i$ in $\mathbb{R}^3$, there always exist 2 other lines that meet all four $\mathcal{S}_i$. At any point $\mathbf{X}$ on either of these two lines, the Hessian will be non-definite. If there are three lines $\mathcal{S}_i$, then the set of points $\mathbf{X}$ at which the Hessian is non-definite consists of a ruled quadric containing the three lines as generators from one of the two generator classes. A generator from the other class, passing through $\mathbf{X}$, will meet all of the three lines (see [18, 30]).

For k subspaces $\mathcal{S}_i$ of codimension r_i in general position, the dimension of the set of lines that meet all of the $\mathcal{S}_i$ is equal to

$$2(N-1) - \sum_{i=1}^{k} (r_i - 1) ,$$

because the set of all lines in $\mathbb{R}^N$ forms a family of dimension $2(N-1)$ and each subspace of codimension r_i provides $r_i - 1$ constraints on this set of lines. Thus, there will be no line that intersects all subspaces (generically) as long as

$$\sum_{i=1}^{k} r_i > 2N + k - 2 . \tag{32}$$

Counting the number of possible subspaces that intersect a given set of subspaces nontrivially is the subject of Enumerative Geometry or Schubert Calculus, involving the cohomology ring of a Grassmann manifold [20]. However, this is far beyond the scope of this paper.

The positive-definiteness of the Hessian is a sufficient, though not necessary condition for the L_1 cost-function to have a single minimum. However, there are many examples where there is not a single minimum. For instance, for lines

in the plane forming the sides of a regular polygon, the sum of distances from any point in the interior of the polygon to the lines is constant and minimum.

In summary, (32) with (31) shows that the condition for a single minimum under the L_q cost (for $q > 1$) is much weaker and much simpler than for $q = 1$.

For simplicity, we make the assumption that in either case there exists a single minimum for the cost-function.

7 Experimental Results: Triangulation

In order to show the applicability of the proposed algorithm we solve the problem of triangulation [17,32]. 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 projection of $3D$ point onto multiple images. Each image point $\mathbf{y}_i$ corresponds to a line in $3D$ space, passing through the center of the camera and intersecting the image plane at $\mathbf{y}_i$. Ideally, all the lines generated by the corresponding points in different images should intersect at a single $3D$ point and that point should 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 it is very likely that these skewed lines do not even intersect with each other in $3D$ space. The problem then is to find an optimal point of intersection of these skewed lines, for which the sum of distances from this point to all the lines is minimum. Hence we can use our algorithm to find a point in $\mathbb{R}^3$ from which the sum of distances to all the lines that is 1-dimensional affine subspaces, is minimum.

Dataset and Starting Point of Algorithms: We applied the proposed algorithm on a well know dinosaur dataset, available at `http://www.robots.ox.ac.uk/~vgg/data.html`. 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. We take the L_2-closest-point as a starting point for the algorithms.

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:

$$\text{RMS error} = \sqrt{\textstyle\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{e}_{\mathrm{j}}^2/\mathrm{n}}\,,$$

where $e_j = \sum_{i=1}^{m_j} d(\mathbf{x}_{ij}, \mathbf{x}'_{ij})/m_j$, 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 from the error that is minimized by the L_q-closest-point algorithm, that is, the distance between a point and its projection onto all of the given lines.

7.1 Convergence Behavior

We compare the proposed L_q optimization algorithms, that is, the gradient descent algorithm (section 5) and the IRLS algorithm (section 3.2), with the L_2-closest-point method and the bundle adjustment algorithm. 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 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 reprojected 3D point and its corresponding image point measurement [32]. However, in this case we are only interested in recovering the 3D structure of a scene; thus, 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 [21].

A comparison of the RMS error over all iterations of the methods is reported in fig 6. As can be seen, the L_1-closest-point method has error less than both the L_2-closest-point method and bundle adjustment. The L_2-closest-point method and bundle adjustment have roughly the same re-projection error. The main reason for smaller RMS error for the proposed algorithm is the greater robustness of L_1 methods to outliers. However, in bundle adjustment and the L_2-closest-point method, a squared error function is minimized which is comparatively less robust to outliers than an L_1 cost-function.

As expected, fig 6 shows that the L_1 optimization algorithm by using the IRLS approach converges close to the minimum quickly than the gradient descent approach. The reason for quick convergence is that the IRLS 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

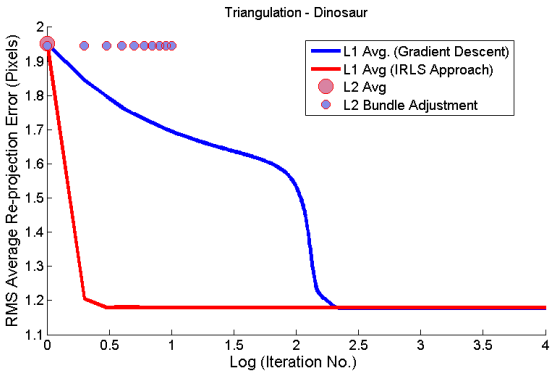


Fig. 6 *Triangulation results for Dinosaur dataset. The above figure shows re-projection error plots for bundle adjustment, the L_2-averaging method (closed form solution exists) and the proposed L_1-averaging methods. As can be seen, the L_1-averaging methods has smaller re-projection error than both the L_2-averaging and bundle adjustment methods. Moreover, the L_2-averaging and bundle adjustment methods have roughly the same error. Observe that the L_1 algorithms find the L_1-closest-points to the rays. As the graph shows this tends to minimize L_1-reprojection error as well.*

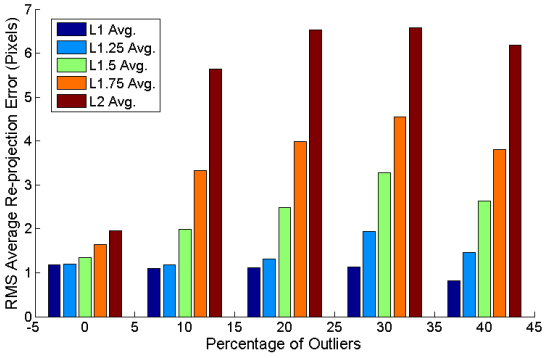


Fig. 7 *Robustness to outliers: The above figure shows re-projection errors of the L_q-averaging method for several values of q ranging from 1 to 2 with an increment of 0.25. We test the algorithms for their robustness by adding different percentage of outliers in the dinosaur dataset, ranging from 0% 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.*

update. This results in a slow convergence rate of the gradient descent approach, as shown by fig 6. In summary, both the L_q optimization approaches give superior results than the L_2 and the bundle adjustment methods. Furthermore, the IRLS approach has a higher convergence rate than the gradient descent approach.

7.2 Robustness to Outliers

In this experiment we show the robustness of the L_q method for different values of q, specifically, for q ranging from 1 to 2 with an increment of 0.25. In order to show the robustness of the methods, we add different number 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 a 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 IRLS algorithm (section 3.2) for L_q optimization because of its higher convergence rate than the gradient descent algorithm. Our experimental results show that the L_1 method is the most robust method than the rest of the methods. Note that the L_1 gives slightly better results even when no outliers are added explicitly to the dataset. For the L_q Weiszfeld algorithm for points in $\mathbb{R}^N$ the results of the L_2 averaging are better than the L_q algorithm in the absence of outliers. We conjecture that this behavior is a result of mismatches in the correspondences in the Dinosaur dataset. In summary, the L_1 method gives better results than the rest of the methods and is therefore recommended in the presence of outliers.

8 Conclusion

This paper presents 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, this paper confirms the fact that in presence of noise and outliers in data, the minimization of an L_1 cost-function gives superior results than an L_2 cost-function. Our experimental results have shown that the L_1-closest-point methods converge close to the ground truth than both the L_2-closest-point and bundle adjustment methods. Furthermore, it is also shown that the IRLS approach of the proposed algorithms converges close to the actual solution in very few iterations than the gradient descent approach.

Ease of implementation and fast iteration make the proposed algorithms attractive wherever L_q optimization is desired. A question that remained partially unanswered is, which computer vision problems can be solved using this technique?

Acknowledgements This research has been funded by National ICT Australia. National ICT Australia is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Aftab, K., Hartley, R., Trumpf, J.: Generalized Weiszfeld algorithms for Lq optimization. Submitted

2. Ameri, B., Fritsch, D.: Automatic 3d building reconstruction using plane-roof structures. ASPRS (2000)
3. Brimberg, J.: Further notes on convergence of the weiszfeld algorithm. Yugoslav Journal of Operations Research (2003)
4. Brimberg, J., Chen, R.: A note on convergence in the single facility minisum location problem. Computers & Mathematics with Applications (1998)
5. Brimberg, J., Love, R.F.: Global convergence of a generalized iterative procedure for the minisum location problem with lp distances. Operations Research (1993)
6. Chartrand, R., Yin, W.: Iteratively reweighted algorithms for compressive sensing. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2008)
7. Chi, E.C., Lange, K.: A look at the generalized heron problem through the lens of majorization-minimization. The American Mathematical Monthly (2014)
8. Daubechies, I., DeVore, R., Fornasier, M., Gunturk, S.: 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)
9. Dick, A.R., Torr, P.H., Ruffle, S.J., Cipolla, R.: Combining single view recognition and multiple view stereo for architectural scenes. In: IEEE International Conference on Computer Vision (2001)
10. Eckhardt, U.: Weber's problem and weiszfeld's algorithm in general spaces. Mathematical Programming (1980)
11. Eldar, Y., Mishali, M.: Robust recovery of signals from a structured union of subspaces. IEEE Transactions on Information Theory (2009)
12. Fletcher, P.T., Venkatasubramanian, S., Joshi, S.: The geometric median on riemannian manifolds with applications to robust atlas estimation. Neuroimage (2009)
13. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Manhattan-world stereo. In: IEEE Conference on Computer Vision and Pattern Recognition (2009)
14. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing building interiors from images. In: IEEE International Conference on Computer Vision (2009)
15. Hartley, R., Aftab, K., Trumpf, J.: L1 rotation averaging using the Weiszfeld algorithm. In: IEEE Conference on Computer Vision and Pattern Recognition (2011)
16. Hartley, R., Trumpf, J., Dai, Y., Li, H.: Rotation averaging. International Journal of Computer Vision (2013)
17. Hartley, R.I., Sturm, P.: Triangulation. Computer Vision and Image Understanding (1997)
18. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision – 2nd Edition. Cambridge University Press (2004)
19. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: the 12th International Symposium on Experimental Robotics (ISER) (2010)
20. Kleiman, S., Laksov, D.: Schubert calculus. American Mathematical Monthly (1972)
21. Lourakis, M.A., Argyros, A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. ACM Trans. Math. Software (2009)
22. Luenberger, D.G.: Linear and nonlinear programming. Springer (2003)
23. Ma, R.: Building model reconstruction from lidar data and aerial photographs. Ph.D. thesis, The Ohio State University (2004)
24. Mordukhovich, B., Nam, N.M.: Applications of variational analysis to a generalized fermat-torricelli problem. Journal of Optimization Theory and Applications (2011)
25. Mordukhovich, B.S., Nam, N.M., Salinas Jr, J.: Solving a generalized heron problem by means of convex analysis. The American Mathematical Monthly (2012)
26. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. ACM Trans. Graph. (2007)
27. Pu, S., Vosselman, G.: Knowledge based reconstruction of building models from terrestrial laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing (2009)
28. Remondino, F., El-Hakim, S.: Image-based 3d modelling: A review. The Photogrammetric Record (2006)
29. Schindler, K., Bauer, J.: A model-based method for building reconstruction. In: First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis (2003)
30. Semple, J.G., Kneebone, G.T.: Algebraic Projective Geometry. Oxford University Press (1979)
31. Taillandier, F.: Automatic building reconstruction from cadastral maps and aerial images. International Archives of Photogrammetry and Remote Sensing (2005)
32. Triggs, W., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.: Bundle adjustment for structure from motion. In: Vision Algorithms: Theory and Practice. Springer-Verlag (2000)
33. Vanegas, C.A., Aliaga, D.G., Benes, B.: Building reconstruction using manhattan-world grammars. In: IEEE Conference on Computer Vision and Pattern Recognition (2010)
34. Weiszfeld, E.: Sur le point pour lequel la somme des distances de n points donnés est minimum. Tohoku Math. Journal (1937)
35. Werner, T., Zisserman, A.: New techniques for automated architectural reconstruction from photographs. In: European Conference on Computer Vision (2003)
36. Wilczkowiak, M., Trombettoni, G., Jermann, C., Sturm, P., Boyer, E.: Scene modeling based on constraint system decomposition techniques. In: Ninth IEEE International Conference on Computer Vision (2003)
37. Yang, L.: Riemannian median and its estimation. LMS Journal of Computation and Mathematics (2010)