Intrinsic regularization of inverse problems involving non-rigid shapes

Yohai S. Devir
Intrinsic regularization of inverse problems involving non-rigid shapes

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Yohai S. Devir

Submitted to the Senate of the Technion — Israel Institute of Technology
Adar 5769 Haifa March 2009
The research thesis was done under the supervision of Prof. Yuval Rabani in the Computer Science Department.

I would like to express my deep gratitude to my advisor, Assoc. Prof. Yuval Rabani, for the guidance and support. I would like to address a special thanks to Prof. Ron Kimmel for his professional help and encouraging along the way.

I would also like to thank my family and all my friends for being with me along the way and helped me accomplish it.

The generous financial support of the Technion is gratefully acknowledged.
6 Conclusions 31
  6.1 Future research directions 31

A Derivation of the update step 33

B Derivation of triangle unfolding 35

C Derivation of other properties 39
  The Hausdorff distance gradient 39
  The area similarity gradient 40
  The volume similarity gradient 40

Abstract in Hebrew 8
List of Figures

1.1 Shadow theater example............................... 5
3.1 Stress Accuracy comparison .......................... 15
4.1 Examples of non-differentiability in geodetic distance calculations 22
5.1 Denoising experiment results ......................... 26
5.2 Convergence of the optimization process in denoising experiment 27
5.3 Stress reduction after sparse model fitting ................ 28
5.4 Configuration and inputs of bundle adjustment experiment 29
5.5 Bundle adjustment experiment results .................. 30

B.1 An example of unfolding triangles in order to split an obtuse triangle to two virtual triangles with acute angles ........ 36
Abstract

In many visions problems, both human and computer, there is a need to reconstruct an unknown three dimensional object from a partial data that is derived from that object. This problem is ill-posed since in many cases, one may find a different object from which similar data can be derived. Therefore, it is common to solve a different problem in which except for the derived data there is some other prior knowledge about the nature of the unknown object. In this work we solve the problems where beside of the derived data, we are also given that the unknown object is similar to (a bending of) a known non-rigid object denoted as the prior.

Many of the objects surrounding us are non-rigid in the sense that they have infinite variety of deformations preserving their topology and density. Among those, one may count the outer skin of living bodies, faces, cloths and others. Two objects that can be obtained from each other by means of non-rigid deformations are said to be intrinsically similar. Many intrinsic similarity measures are based on the geodesic distances between two points on the shape measuring the length of the minimal path contained in the shape connecting those two points.

In computer vision applications, shapes are approximated by a set of samples forming a triangulated mesh. There are two common ways to approximate geodesic distances on meshes, using the Dijkstra’s well known shortest paths algorithm and using the fast marching method (FMM), which lacks some of the inherent disadvantages of Dijkstra’s algorithm.

In this work, we propose to solve such reconstruction problems in the framework of optimization problems, where the optimization variables are the spatial locations of the shapes’ vertices. In order to do so we enhance the FMM algorithm so that beside of approximating the geodetic distances, it also calculate the derivatives of those distances with respect to the spatial
locations of the shape’s vertices.

Synthesis examples are given for a few shape reconstruction applications.
Abbreviations and Notations

\begin{itemize}
  \item $X$ — A shape
  \item $d_X$ — A metric in shape $X$
  \item $C$ — A correspondence
  \item $x_i$ — A shape’s sample
  \item $\pi_i$ — Coordinate vector of a shape’s sample
  \item dis — distortion
  \item $d_{\text{GH}}$ — Gromov-Hausdorff distance
  \item $d_{\text{H}}$ — Hausdorff distance
  \item $X$ — Shapes space
  \item $d_X$ — A distance in the shapes space
  \item $Y$ — Measurements space
  \item $d_Y$ — A distance in the measurement space
  \item $P$ — Measurement operator
  \item $Y$ — A measurement
  \item $d_{\text{R3}}$ — Minimal Euclidean distance
  \item $\tilde{X}$ — Unknown shape
  \item $\hat{X}$ — Reconstructed shape
  \item $X_0$ — A prior shape
  \item $x_s$ — A source vertex
  \item $t_i, T(x_i)$ — Geodesic distance between vertex $x_i$ and a known source vertex
  \item $\mathcal{F}_{\text{FMM}}$ — FMM update function
  \item $\sigma$ — stress
  \item GH — Gromov-Hausdorff
  \item FMM — Fast Marching Method
  \item MDS — Multidimensional Scaling
  \item GMDS — Generalized Multidimensional Scaling
  \item 2D — Two dimensional
  \item 3D — Three dimensional
  \item w.r.t. — with respect to
\end{itemize}
Chapter 1

Introduction

In many vision tasks, both human and computer vision, one tries to deduce the shape of an object given an observation or a measurement thereof. For example, in shadow theater (Fig. 1.1) we recognize shapes of objects from silhouettes projected on a screen. Another reconstruction task is performed by our brain almost unnoticeably when deducing the shape of a three-dimensional (3D) object from a pair of two-dimensional (2D) images of the object observed by our two eyes. In computer vision, these two tasks are known correspondingly as shape from silhouette and shape from stereo.

Figure 1.1: Shadow theater example - a shadow of a bird and a camel being cast by two hands.
Other computer vision tasks involving deducing the shape of a 3D object from other kinds of measurements include *shape from motion*, where the measurement is a sequence of still images; *shape from shading*, where the measurement is a single image of the object illuminated by a light source; and others. All these problems in which an object is reconstructed based on some measurement are known as *shape reconstruction problems*. They constitute a subset of what is called *inverse problems*.

A common aspect of such problems is that in many cases they are under-determined, in the sense that measuring different objects may yield similar measurements. For example, the essence of the shadow theater is that it is hard to distinguish between shadows cast by an animal and shadows cast by hands. If a prior knowledge about the unknown object is given (e.g. the shadow is cast by an animal in conjunction with a set of possible animals) we have better chances reconstructing the object. Such additional information is known as a *prior*.

Of particular interest are reconstruction problems involving non-rigid shapes. The world surrounding us is full with objects such as live bodies, paper products, plants, clothes etc. Every such object may be deformed to an infinite number of different postures. While bending, though, objects tends to preserve their internal geometric structure. Two objects differing by a bending are said to be *intrinsically similar*. In many cases, while we do not know the measured object, we have a prior on its intrinsic geometry. For example, in the shadow theater, though we do not know which exact posture of the hand casts the shadow, we know that the unknown object is a deformation of a hand. In other words, our prior is a shape of the hand, and the variability of the reconstructed object is the bending of the hand.

There are many approaches to specific reconstruction of shapes-from-X. Among those are the following. *Shape from shading (SFS)* [29] where a 3D shape is recovered from one or more 2D images, utilizing the variations of shading in the image. The reconstruction is usually done by minimizing some cost functional or iteratively reconstruct the object from known initialization points. Among common priors to SFS are smoothness of the surface, brightness, similarity between intensity gradients of the image extracted from the input image, symmetry or a combination of a few of them. *Bundle adjustment* [25] where a 3D shape is reconstructed given a set of 2D images of the shape from different viewpoints taken by possibly non-
calibrated cameras. **Pose transplantation** (*shape completion*) where a full shape is constructed given only the spatial locations of a few of its points. This is done, for example, when creating an animated character mimicking the motion of a human actor by placing a few markers on his body, and moving the animated character accordingly. In [1], a full model of various bodies and postures is used for this task. Here we use a single static model of a specific actor. **Shape preserving denoising** where a partially smooth shape is recovered from a shape with high frequency geometric components. Algorithms that eliminate high frequency geometric noise by smoothing a noisy surface such as [10, 16] do not incorporate knowledge about the geometry of the surface thus may distort its shape. Maintaining the intrinsic similarity between the noisy surface and the smoothed one can limit the shape’s distortion.

Due to the under-determined nature of shape-from-X applications, knowledge about the intrinsic structure of the shape may improve the result of existing methods. In particular, by using a functional measuring intrinsic similarity, the knowledge about the intrinsic structure of the shape may be incorporated in any of the many methods of solving shape from X problems that minimize some functional. Certain functionals can be minimized by solving the resulting Euler-Lagrange equations, possibly by a gradient descent process.

In order to utilize the intrinsic similarity of a shape as a prior we need to define a functional describing the distortion between two shapes. A few papers such as [14] and [19] computes the local distortion of the mesh edges and therefore unsuitable to compute a deformation that affects the global distortion of the mesh. The scale and location at which the geodesic distances are preserved can play an important role in some applications, while restriction to preserving length of edges between neighboring vertices may integrate to undesired deformations at larger scales.

One of the first attempts to measure intrinsic similarity between arbitrary surfaces was made by Elad and Kimmel in [15] who mapped the metric structure of the shapes to a low dimensional Euclidean space using multidimensional scaling (MDS) and compare the resulting shapes using extrinsic methods. Yet, low dimensional Euclidean embedding allows representing the intrinsic geometry of the shapes only approximately. This problem can be solved by using the Gromov-Hausdorff (GH) distance [18] as first suggested
by Mémoli and Sapiro [22]. In [5] Bronstein et al. proposed a method formulating the GH distance using numerical scheme similar to MDS (dubbed generalized MDS or GMDS) [6], in which one surface is embedded into another while reducing the intrinsic dissimilarity between the embedded surface and the embedding one. Since this method limits one surface to be embedded into another, it cannot be adopted for finding a shape that is embedded in \( \mathbb{R}^3 \).

In [7] Bronstein et al. show a method to avoid this limitation by deriving a combination of intrinsic and extrinsic similarities criteria. This problem can be solved as a particular case of the inverse problems considered here, where the measurement space is the set of extrinsic coordinates of the points in \( \mathbb{R}^3 \) and the distance on the measurement space is the Hausdorff distance between the coordinates of the shapes. Since their method is based on calculating the geodesic distances on a shape using Dijkstra’s algorithm, it is sensitive to the shape triangulation and is affected by the metricalation error as explained in Sec. 3.1.

In this paper we show a method less sensitive to triangulation changes and that converges to the exact measure of intrinsic dissimilarity. The numerical core of our method is based on an extension to the fast marching algorithm that enables us to calculate the derivatives of pairwise geodesic distances w.r.t. the mesh’s vertices spatial locations.

A different approach to metric change as was in [2],[24],[28] in which a conformal equivalent metric satisfying a given conditions is calculated on a given mesh. Those methods can be used for measuring the intrinsic dissimilarity between shapes by cascading transformations from one shape to some embedding and from the embedding to the second shape, such as in [21]. In contrast to our method, the embedding of the mesh in those methods is fixed, while the limit of the generated metric for short paths is not the Euclidean embedding distance.

1.1 structure of this dissertation

This thesis is organized as follows: Chapter 2 establishes the mathematical background including the definitions of shape spaces, measurement spaces and the reconstruction problems in general. Chapter 3 reviews the methods of calculating geodesic distances which are the basis for intrinsic similarity
definitions. In chapter 4 we explains the extension of the FMM algorithm for distances derivatives calculation, and the necessary modifications of the similarity functional needed for eliminating constellations which are non-differentiable or non-continuous. Chapter 5 demonstrates some applications that uses the proposed method. Finally, we provide concluding remarks and future research directions in Chapter 6.
Chapter 2

Mathematical background

2.1 Shape space

We model a shape $X$ as a metric space - a pair $(X, d_X)$ where $X$ is a 2D smooth, compact, connected and complete Riemannian manifold embedded into the Euclidean space $\mathbb{R}^3$, and $d_X : X \times X \to \mathbb{R}^+ \cup \{0\}$ is the geodesic metric. We denote by $\mathcal{X}$ the shape space in which each element is a shape.

Given two shapes $W, X \in \mathcal{X}$, we define the correspondence between them as $C \subset X \times W$ such that $\forall x \in X \exists w \in W : (x, w) \in C$ and $\forall w \in W \exists x \in X : (x, w) \in C$. We say $x \in X$ and $w \in W$ are in correspondence if $(x, w) \in C$. The distortion of a correspondence is given by

$$\text{dis} \ C = \max_{(x', w') \in C} |d_X(x, x') - d_W(w, w')|$$  \hspace{1cm} (2.1)

which measures how different the corresponding metric structures in $X$ and $W$ are.

A subset $X' \subseteq X$ is said to be an $\varepsilon$-net in (or $\varepsilon$-covering of) $X$ if

$$\max_{x \in X} \min_{x' \in X'} d_X(x, x') \leq \varepsilon.$$  \hspace{1cm} (2.2)

If a correspondence $C$ has $\text{dis} \ C \leq \varepsilon$ and the set of all points in $X$ having correspondence in $W$, $X' = \{x : \exists w, (x, w) \in C\}$ is an $\varepsilon$-net, then $X$ and $W$ are $\varepsilon$-isometric.

The similarity of two shapes in $\mathcal{X}$ is measured using a metric $d_X$. Since the elements of $\mathcal{X}$ are metric spaces we can use the Gromov-Hausdorff [18]
distance as $d_X$,

$$d_{GH} = \min_C \text{dis C}$$  \hspace{1cm} (2.3)

If $d_{GH}(X, W) \leq \varepsilon$, then $X$ and $W$ are $2\varepsilon$-isometric, and if $X$ and $W$ are $\varepsilon$-isometric, $d_{GH}(X, W) < 2\varepsilon$. In particular, $d_{GH}(X, W) = 0$ iff $X$ and $W$ are isometric.

When dealing with sampled shapes, it is important to bear in mind that most polyhedral shapes (including triangulated meshes) without boundaries are in fact rigid, i.e. do not have incongruent isometries. This was conjectured by Euler, proven for convex shapes by Cauchy [9], proven that almost all such shapes are rigid by Gluck [17], though disproved by Connelly [11]. As a result, most closed sampled shapes originated from computer models are rigid, so two deformations may be only approximate isometries. Moreover, objects that in the real world are non-rigid, such as palm’s skin, are isometric due to small wrinkles near the joints. Triangulated meshes approximating two deformations of such object with limited resolution will fail to capture and represent those wrinkles. As a result, two meshes approximating two isometric deformations of a non-rigid shape may get further from being isometric.

### 2.2 Measurement space

Let $\mathcal{Y}$ be a space, and $d_\mathcal{Y} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{0\}$ be a distance on $\mathcal{Y}$. Next, let $P$ be an operator $P : \mathcal{X} \to \mathcal{Y}$. For example, in the SFS inverse problem, $\mathcal{Y}$ is the space of gray scale images, $P$ is an operator assigning a patch on the shape to a pixel in the image whose intensity is determined by the average dot product between the surface normal at a point in this patch, and the illumination direction. In this case, $d_\mathcal{Y}$ may be the Frobenius norm. Another example, of shape completion, $\mathcal{Y} = P(\mathcal{X})$, $\mathcal{X} \in \mathcal{X}$ is a sparse set of points on $\mathcal{X}$, $d_\mathcal{Y}$ may be the Hausdorff distance,

$$d_H(Y, Z) = \max\{\max_{z \in Z} d_{\mathbb{R}^3}(z, Y), \max_{y \in Z} d_{\mathbb{R}^3}(y, Z)\},$$  \hspace{1cm} (2.4)

where $d_{\mathbb{R}^3}(x, Y) = \min_{y \in Y} \|x - y\|$ denotes the minimal Euclidean distance between a point $y \in Y$ and the point $x$. 


2.3 Reconstruction problems

We formulate the generic problem of shape reconstruction as follows. Given \( Y = P(\tilde{X}) \) where \( \tilde{X} \) is an unknown shape, recover a shape \( \hat{X} \approx \tilde{X} \) that best fits the given data

\[
\hat{X} = \arg\min_{\hat{X} \in \mathcal{X}} d_X(P(X), Y). \tag{2.5}
\]

In the general case where \( P \) is not injective and does not preserve distances, this problem is ill-posed, and it is possible to find many solutions for which \( d_Y(P(\hat{X}), Y) \) is small, while \( d_X(\tilde{X}, \hat{X}) \) is large. Moreover, in some cases, even if there is a single solution \( \hat{X} \) such that \( d_Y(P(\hat{X}), Y) \) is small, it is not always known how to find it.

Therefore, it is common practice in inverse problems to solve a different problem in which aside from \( Y = P(\tilde{X}) \), a prior shape \( X_0 \) is also given, such that \( d_X(\tilde{X}, X_0) \) is small while \( d_Y(Y, P(X_0)) \) may be large. Using the prior shape as a regularization in Eq. 2.5, we solve the following problem

\[
\hat{X} = \arg\min_{\hat{X} \in \mathcal{X}} d_X(X, X_0) + \lambda d_Y(P(X), Y), \tag{2.6}
\]

where \( \lambda > 0 \) is some parameter.

We look for a shape \( X \) which is intrinsically similar to the prior \( X_0 \) and whose measurement is similar to \( Y \). Our way of solving Problem (2.6) consists of deforming an initial shape \( X \) and evaluating the result of this deformation on \( d_X(X, X_0) \) and \( d_Y(P(X), Y) \).
Chapter 3

Geodesic distance computation

Having the surface sampled at $N$ points $\{x_i\}_{i=1}^N$, whose coordinate vectors are correspondingly denoted by $\{x_i\}_{i=1}^N$, a triangulated mesh can be defined with the surface’s samples as vertices.

Computation of geodesic distances is a key step in the calculation of intrinsic similarity. The straightforward method of approximating geodesic distances over a triangulated mesh is to consider the mesh as a weighted chordal graph $(V, E)$ for which the weight of an edge is the Euclidean distance between its ends. Next, define the geodesic distance between two vertices as the shortest path connecting those vertices. This is commonly done using Dijkstra’s algorithm [12].

Sethian [23] and Tsitsiklis [26] introduced methods for approximating weighted distance maps on regularly sampled domains with a weighted flat metric. Sethian coined the term fast marching method to his algorithm. By interpreting the geometry behind the numerical update step of these solvers, Kimmel and Sethian [20] extended the fast matching method to triangulated domains.

The geodesic distance map from a source point $x_s \in X$ is defined as $d_{x_s} : X \to \mathbb{R}^+ \cup \{0\}$ such that for each $x \in X$ we have $d_{x_s}(x) = d_X(x_s, x)$. On a triangulated surface, approximating a continuous one, the geodesic distance map can be computed by solving an eikonal equation on the surface. The eikonal equation captures the fact that the distance map can be defined.

13
by its gradient whose magnitude equals one almost everywhere.

The fast marching algorithm estimates the geodesic distances from a given source vertex \( x_s \) by simulating a front starting at the source and propagating with a constant unit velocity. The front arrives to a vertex \( x \) at time \( T(x) \) \( (T(x_s) = 0) \) indicating that the distance between \( x \) and the source vertex is \( T(x) = d_X(x_s, x) \).

The FMM algorithm is similar to Dijkstra’s algorithm, except for its update step. Unlike Dijkstra’s algorithm in which a distance is assigned to a vertex based on a single neighboring vertex and the connecting edge, the FMM algorithm assigns a distance to a vertex based on the distance values assigned to two of its adjacent neighbors and the geometry of the triangle containing the three vertices. The values assigned to the three corners define a function on the triangle whose gradient magnitude is equal to one. For a detailed description and numerical analysis of the update step and a modification for a parallel processor refer to [27].

Let \( \{a, b, c\} \) be the edges of the acute triangle \( \{x_1, x_2, x_0\} \). Given the distance values \( t_1 \) and \( t_2 \), the FMM update step estimates \( t_0 \) by

\[
t_0 = \mathcal{F}_{\text{FMM}}(t_1, t_2, x_0, x_1, x_2)
\]

\[
= \begin{cases}
  t_1 + \frac{rv}{c} + \frac{2}{c} \sqrt{c^2 - v^2} & \text{some condition holds} \\
  \min\{t_1 + b, t_2 + a\} & \text{otherwise}
\end{cases}
\]

where \( v = t_2 - t_1 \), \( r = \frac{b^2 + c^2 - a^2}{2c} \) and \( s = \sqrt{b^2 - v^2} \). It is a solution to the quadratic equation that restricts the gradient to be one.

In the case of obtuse triangles, using Eq. 3.1 may lead to solutions which do not satisfy the consistency (\( t_0 > \max\{t_1, t_2\} \)) and monotonicity (increasing \( t_1 \) or \( t_2 \) increases \( t_0 \)) conditions.

Kimmel and Sethian suggested in [20] to handle obtuse angles by splitting these angles in order to form two new acute ones (referred as virtual triangles). The splitting is done by unfolding the triangle opposing the obtuse angle to the \( x_0x_1x_2 \) plane. We denote this triangle by \( x_1x_2x_3 \) and the vertex formed by unfolding \( x_3 \) to the \( x_0x_1x_2 \) plane by \( \tilde{x}_3 \). If the line between \( x_0 \) and \( \tilde{x}_3 \) splits the obtuse angle to two acute ones, \( t_0 \) is calculated using Eq. 3.1 in triangle \( x_0x_1\tilde{x}_3 \) or \( x_0x_2\tilde{x}_3 \). If the \( x_0\tilde{x}_3 \) line splits the obtuse angle to an obtuse and an acute angle, this process is repeated recursively.
to a triangle adjacent to triangle $x_1x_2x_3$ until a vertex that splits the obtuse angle to two acute ones is found. For more details see Appendix B.

### 3.1 Dijkstra vs. FMM

Approximating geodesic distances using Dijkstra’s algorithm has two drawbacks. First, given two $\varepsilon$-isometric shapes $X$ and $Y$, the distances measured between corresponding locations on $X$ and $Y$ using FMM may defer by at most $\varepsilon + \delta$, where $\delta$ depends on the sampling and can be arbitrarily small as $N \to \infty$. Using Dijkstra, $\delta$ cannot be arbitrarily small since the graph induces a metric which is inconsistent with that of the underlying continuous surfaces. This phenomena is known as metrication error. The effect of this drawback on the similarity measure is demonstrated in Fig. 3.1.

![Figure 3.1: The accuracy of measuring the distortions of pairwise distances using Dijkstra’s algorithm and FMM. A unit square (left) sampled uniformly in various resolutions ($N$ is the number of samples) is sheered (center). The Frobenius norm of the difference between the matrix of pairwise distances in the original surface and the matrix of pairwise distances in the sheered square is denoted by $\sigma$. This norm was calculated using FMM ($\sigma_{FMM}$), Dijkstra ($\sigma_D$) and analytically ($\sigma_A$). Right: The ratios of $\sigma_{FMM}/\sigma_A$ and $\sigma_D/\sigma_A$ calculated in various resolutions.](image-url)
In particular, Dijkstra’s algorithm is sensitive to the triangulation. Distances measured on a shape with one triangulation may significantly differ from those measured on the same shape triangulated differently. Therefore, two shapes formed from the same set of samples and two different triangulations may seem dissimilar when calculating the stress using Dijkstra’s algorithm and nearly isometric when calculated using FMM.
Chapter 4

Implementation Considerations

Since $\varepsilon$-isometry is sensitive to outliers, we use the following notion of intrinsic similarity known as stress. Given two surfaces $W$ and $X$, both with $N$ samples, we define the stress $\sigma(W, X)$ as

$$\sigma(X, W) = \min_C \sum_{j,k \in C} (d_X(x_j, x_k) - d_W(w_j, w_k))^2, \quad (4.1)$$

where $C$ is a correspondence between $X$ and $W$.

Using $\sigma$ as a measure of the intrinsic dissimilarity, we solve the following version of Eq. 2.6

$$\hat{X} = \arg\min_{X \in \mathbb{X}} \sigma(X, X_0) + \lambda d_Y(P(X), Y). \quad (4.2)$$

This problem can be solved as an optimization problem where the optimization variables are the coordinates of the vertices of $X$. The initialization is the prior $X_0$ unless a better guess is available.

In order to find $\hat{X}$ of Eq. 4.2, we need to differentiate both $d_Y(P(X), Y_0)$ and $\sigma(X, X_0)$ w.r.t. the coordinates of the samples of $X$. Calculating the gradient of $d_Y(P(X), Y_0)$ for a few operators $P$ is described in the Appendix C;

In [7], the gradient of $\sigma(X, X_0)$ is differentiated from the calculation of $\sigma(X, X_0)$ using Dijkstra’s algorithm. Here, we calculate a less trivial yet
more accurate gradient that is based on FMM. Vertex-wise, the optimization has the form of

\[ \mathbf{x}_i \leftarrow \mathbf{x}_i + \alpha_i, \]

where \( \mathbf{x}_i = (\alpha_i^x, \alpha_i^y, \alpha_i^z)^T \) is the distance \( x_i \) moves in each coordinate direction.

We denote by \( \alpha \) the matrix in which \( (\alpha)_{ij} = \alpha^j_i \). Since the analysis is similar for all coordinates, we omit the superscript index for notation simplicity.

### 4.1 Gradient of the stress

Given two shapes \( X \) and \( X_0 \), both with \( N \) samples and a correspondence \( C \), the stress between \( X \) and \( X_0 \) can be written as

\[ \sigma(X, X_0) = \sum_{j,k} \left( d_X(x_j, x_k) - d_{X_0}(x_{0j}, x_{0k}) \right)^2. \]  

(4.4)

Differentiating the stress w.r.t. \( \alpha_i \), we obtain

\[ \frac{\partial \sigma(X, X_0)}{\partial \alpha_i} = 2 \sum_{j,k} \left( d_X(x_j, x_k) - d_{X_0}(x_{0j}, x_{0k}) \right) \cdot \frac{\partial d_X(x_j, x_k)}{\partial \alpha_i}. \]  

(4.5)

In order to calculate the derivatives of the stress, we need to calculated the pairwise distances in \( X \), the pairwise distances in \( X_0 \), and the derivatives of the geodesic distances in \( X \). The pairwise distances in \( X \), \( d_X(x_j, x_k) \), must be recomputed after every iteration of the optimization by running the FMM algorithm \( N \) times, one for each vertex as a source. The pairwise distances in \( X_0 \), \( d_{X_0}(x_j, x_k) \) are pre-computed only once in a similar way and can be considered as an input to the optimization problem. The derivatives of the geodesic distances in \( X \), \( \frac{\partial d_X(x_j, x_k)}{\partial \alpha_i} \), are calculated as follows.

Consider a vertex \( x_0 \), and a source vertex \( x_s \). We denote the triangle which is used for the final calculation of the distance between \( x_s \) and \( x_0 \) as \( x_0 x_1 x_2 \). Given the spatial locations of the triangle vertices \( \{ \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2 \} \). Also given the estimated distances of vertices \( x_1 \) and \( x_2 \) from the source \( (d_X(x_1, x_s), d_X(x_2, x_s)) \), and the derivatives of these estimated distances w.r.t. the movement of each vertex \( \left( \frac{\partial d_X(x_1, x_s)}{\partial \alpha_i}, \frac{\partial d_X(x_2, x_s)}{\partial \alpha_i} \right) \).
The derivatives of the distance between $x_0$ and $x_s$, denoted by $\frac{\partial d_X(x_0,x_s)}{\partial \alpha_i}$, can be calculated by differentiating Eq. 3.1 using the chain rule.

$$\frac{\partial d_X(x_0,x_s)}{\partial \alpha_i} = D_{\text{FMM}} \left( \frac{\partial d_X(x_1,x_s)}{\partial \alpha_i}, \frac{\partial d_X(x_2,x_s)}{\partial \alpha_i}, d_X(x_1,x_s), d_X(x_2,x_s), x_0, x_1, x_2 \right).$$

(4.6)

Details of deriving the update step are described in Appendix A.

If the angle $\angle x_1 x_0 x_2$ is acute, the triangle edge lengths are simply the $L_2$-norm between pairs of vertices. If the angle $\angle x_1 x_0 x_2$ is obtuse, Eq. 4.6 can be adapted by splitting $\angle x_1 x_0 x_2$ as explained in Sec. 3. The derivatives of the distances between $\hat{x}_3$ and $\{x_1, x_2, x_0\}$ w.r.t. $\alpha_i$ are calculated using the chain rule along the process of the unfolding. See Appendix B for details. Note that the derivatives of the triangle edges’ length w.r.t. $\{\pi_i\}$ are calculated only once for all source vertices.

Consequently, the derivatives of $d_X(x_0,x_s)$ w.r.t. $\alpha_i$ are calculated according to Eq. 4.6.

In order to calculate all derivatives of all pairwise distances, we propose the following modification of the FMM algorithm. At the initialization step, we also initialize $\frac{\partial d_X(x_s,x_s)}{\partial \alpha_i} = 0, \forall x_i \in X$. In every FMM iteration, we calculate the derivatives $\frac{\partial d_X(x_s,x_i)}{\partial \alpha_i}$ immediately after calculating $d_X(x_s,x_i)$ at the same triangle containing $x_i$ used for calculating the fixed value of $d_X(x_s,x_i)$.

By updating the derivatives of $\sigma(X,X_0)$ on the fly, we need to store only the derivatives that may be used for future calculations. These are the derivatives of $d_X(x,x_s)$ for vertices $x$ whose distance from $x_s$ was already set (in some FMM related papers such vertices are denoted by black or fixed vertices) and $\exists x_j \in \mathcal{N}(x)$ whose distance from $x_s$ was not set (using the same terminology, this vertex is denoted as a red vertex). Other derivatives of distances may be discarded.

For practical reasons the following adjustments are made.

4.2 Path fixing

The phenomena of vertex $x_0$ for which there are two different shortest paths connecting to the source is called a shock. The gradient of the distance function at a shock is not well defined. If $d(x_0,x_s)$ is calculated in triangle

19
$x_0x_1x_2$, increasing $t_1$ or $t_2$ may not lead to an increase of $t_0$, since $t_0$ may be updated through another path and another triangle. This means that the stress between $X$ and $Y$ as a function of the coordinations of the vertices of $X$ is not always a differentiable function. However, the stress calculated using a set of fixed paths connecting every two vertices is always a differentiable function. Therefore, we calculate a pairwise distance map as a reference and then keep the order of the updates fixed.

A shock which is formed inside a single triangle ($x_0$ is updated by $\min(t_1 + b, t_2 + a)$) is treated differently as explained in Section 4.4.

This update process has two desirable properties. Assuming that at a certain stage of an optimization process, vertex $x$ is moved away from it’s neighbor vertices such that its mean Euclidean distance to it’s neighboring vertices is much bigger than the average distance between its neighboring vertices. We denote this detail of a shape as a spike. Let also assume that this spike did not exist in the reference surface used for the stress calculation. If the paths are not fixed, vertex $x$ will affect only the distances from $x$ to the rest of the mesh, thus the projection of the gradient of the stress over the vector describing a motion of this vertex alone will be relatively small, causing this vertex to remain a spike. At the other end, all pairwise distances that used to be affected by $x$ are affected by $x$’s neighboring vertices. Therefore, every movement of these vertices will cause a relatively large change in the stress. This will cause these vertices to be kept in place preserving the spike and forming a local minima of the cost function. Fixing the paths eliminates this phenomena.

Another effect of fixing the paths is a runtime reduction, since the order of updating the vertices is known and there is no need to find the unset vertex whose distance from the source is minimal.

### 4.3 Smoothing the transition between calculations

$d_X(x_0, x_s)$ may be calculated according to triangle $x_0x_1x_2$ in one way out of a couple of possible ways. If $\angle x_1x_0x_2$ is acute, it will be updated using Eq. 3.1 in triangle $x_0x_1x_2$. If $\angle x_1x_0x_2$ is obtuse, it will be updated using Eq. 3.1 in the virtual triangle $x_0x_1\hat{x}_3$ or $x_0x_2\hat{x}_3$. Moreover, since $\angle x_1x_0\hat{x}_3$ and $\angle x_2x_0\hat{x}_3$ must be non-obtuse, $\hat{x}_3$ may not be the same vertex at different optimization steps.
As a consequence, a small perturbation in the mesh may cause a change in the calculation resulting a relatively large change in the distances calculations, and therefore a discontinuity of the stress. This happens in the following scenarios.

- **The updating triangle is changing from obtuse to acute or vice-versa.** Assuming a vertex $x_0$ is updated from triangle $x_0x_1x_2$. If $\angle x_1x_0x_2$ is about 90°, a small perturbation of $\overrightarrow{x_0}$, $\overrightarrow{x_1}$ or $\overrightarrow{x_2}$ in the updated triangle could cause the update scheme to change from a calculation in the $x_0x_1x_2$ triangle to calculation in the $x_0x_1\hat{x}_3$ and $x_0x_2\hat{x}_3$ triangles. This scenario is demonstrated in Fig. 4.1 (Left).

- **Obtuse triangle splitting is about to change.** (Fig. 4.1, middle). Assuming a vertex $x_0$ is updated from an obtuse triangle $x_0x_1x_2$ ($\angle x_1x_0x_2$ is obtuse). The line $\hat{x}_3x_0$ should split $\angle x_1x_0x_2$ to two acute angles which are smaller than $\angle x_1x_0x_2$. If due to a small perturbation $\hat{x}_3x_0$ will not split $\angle x_1x_0x_2$ as required, $\angle x_1x_0x_2$ will be split by connecting it to a different vertex, $\hat{x}_4$. In Fig. 4.1 (middle), this happens when $\angle x_2x_0\hat{x}_3$ is almost right. It should be noted that if $\angle x_1x_0x_2 \approx \pi$, both $\angle x_2x_0\hat{x}_3$ and $\angle x_1x_0\hat{x}_3$ are almost right and therefore, a small perturbation may change the virtual connectivity between three update schemes.

- **A combination of more than one change of calculation.** A small perturbation of the location of $\pi_i$ may change more than one calculation. For example, as seen in Fig. 4.1 (right), $\angle x_2x_0x_1$ is almost right, and a small movement of $\hat{x}_3$ in the $-y$ direction will cause that $\angle \hat{x}_3x_0x_1$ to be larger than $\angle x_2x_0x_1$, therefore changing the virtual triangles such that $x_0$ will be connected to $\hat{x}_4$. Note that $\angle \hat{x}_3x_0x_1$ may be still acute in such a scenario.

We denote a mesh in which a small perturbation may change the way a distance is calculated as a **nonstable mesh**. We also denote the calculation that is performed assuming that some angle’s type will be changed from an acute to obtuse or vice versa as a **speculative calculation**, and the mesh in which the said angle has changed is denoted as **speculative mesh**.

The discontinuities are treated by using a weighted average between the current scenario and the speculative scenario assuming an angle affecting
Figure 4.1: The cases in which the method of calculation is changing. In all images - updating \( x_0 \) comes from \( x_0x_1x_2 \) triangle. Left: The angle of the update triangle changes from acute to obtuse. Middle: A change of the virtual triangles. Right: A combination of the scenarios on the left and the middle.

the vertex’s distance calculation is treated as an obtuse angle. We define an angle \( \theta \) as almost obtuse if \( \cos(\theta) < L_1 \) for some small constant \( L_1 \) (we use \( L_1 = 0.02 \)). We calculate a distance using speculative scenarios if at least one of the angles affecting the current calculation is nearly obtuse. The weight we between current calculation and the speculative ones, denoted by \( h \), is calculated by

\[
h(V, L_0, L_1) = \frac{\max H}{2} \left( 1 + \sin \left( \frac{\pi}{2} \frac{2V - L_1 - L_0}{L_1 - L_0} \right) \right),
\]

where \( V = \cos(\theta) \), and \( L_0 \) is 0 for transitions in non-virtual triangles (the first scenario) and the maximum between 0 and cosine of the split triangle angle (that may be almost obtuse) when changing from one splitting of the triangle to a different one (at the second scenario). \( \max H \) is \( h(\cos(\frac{\angle \hat{x}_3x_0x_2}{2}), L_0, L_1) \) if both \( \angle \hat{x}_3x_0x_1 \) and \( \angle \hat{x}_3x_0x_2 \) are almost obtuse and 1 otherwise.

In some cases the speculative calculation has other angle which is almost obtuse and therefore the speculative distance is recursively calculated as another weighted average. In the case of a nearly straight angle, both \( \angle \hat{x}_3x_0x_1 \) and \( \angle \hat{x}_3x_0x_2 \) may change to be obtuse. Therefore, the speculative calculation is a weighted average of the speculative calculation assuming \( \angle \hat{x}_3x_0x_2 \) is
about to become obtuse and the speculative calculation assuming \( \angle \hat{x}_3x_0x_1 \) is about to become obtuse. The weight we use between this two speculative calculations is \( h(\cos(\angle \hat{x}_3x_0x_1), 0, \cos \frac{\angle \hat{x}_3x_0x_1}{2}) \).

Theoretically, since a speculative calculation may be based on another speculative calculation, there can be a \( O(2^N) \) speculative calculations where \( N \) is the number of vertices in the mesh. However, for small values of \( L_1 \) in non-pathological meshes, there are almost always less than \( 2^5 \) speculative calculations.

Derivatives of \( h \) are calculated by the chain rule using the derivatives of the edges forming the triangles affecting the weighted sum.

### 4.4 Eliminating shocks within a triangle

In case that a vertex is updated using Dijkstra scheme (the second case in Eq. 3.1), a shock may be formed when \( t_1 + b = t_2 + a \). This shock causes meshes for which the derivative of the distance function does not exist. We eliminate this problems by calculating the \( t_0 \) in those cases by

\[
t_0 = \begin{cases} 
\min\{t_1 + b, t_2 + a\} & \|(t_1 + b) - (t_2 + a)\| > \varepsilon \\
\frac{h(t_2 + a) + (1 - h)(t_1 + b)}{\max H} & \text{otherwise} 
\end{cases}
\]  

where \( \varepsilon \) is some small constant and \( h \) is calculated using Eq. 4.7 using \( V = (t_1 + b) - (t_2 + a) \), \( L_0 = -\varepsilon \), \( L_1 = \varepsilon \) and \( \max H = 1 \).

### 4.5 Limiting the paths affecting the stress

In some applications, making local patches intrinsically similar to the corresponding local patches in a second shape is more important than keeping all pairwise distances similar. However, due to the definition of the stress a relatively small distortion in a single long pairwise distance has more effect on the final result than a few relatively large distortions in short pairwise distances. Therefore we give more weight to local pairwise distances.

Ignoring completely the relatively long pairwise distances also reduces
the runtime complexity. This is done by calculating the stress as

$$\sigma(X, X_0) = \sum_{j,k} \left( d_X(x_j, x_k) - d_{X_0}(x_{0j}, x_{0k}) \right)^2.$$

(4.9)

for some constant $K$ and the correspondence $C$.

Since the paths are fixed, $K$ that is smaller than the length of the longest edge in $X_0$ may lead to infinite distances and should be avoided.

### 4.6 Convergence speedup

In general, we use gradient descent optimization with line search in order to solve Eq. 4.2. In order to speed up the convergence we use the following techniques.

**Vector extrapolation** After every twelfth iteration, we estimate the limit of the series formed by the last twelve shapes. We do this using the MPE algorithm [8]. The extrapolated limit of the series is then taken as a descent direction and a line search is done along this direction.

**Alternating minimization** Since the optimization is ill-conditioned, an alternating optimization was performed, alternating between a set of vertices whose gradient’s norm is more than 3 standard deviations from the mean norm of all vertices, and the set of the rest of the vertices. The two sets were updated every six iterations.

**multi-resolution optimization** Unless stated otherwise, the result is calculated in a coarse to fine sequence, where a coarser model is a subsampling of a finer model. After solving the problem on a coarse model, a finer model is extrapolated from the coarse result and is used as an initialization for the optimization of a finer model.
Chapter 5

Results

Most of the previous papers dealing with differentiation of the metric of a surface have used the gradient of the metric for comparing different shapes. In this section we show a few examples of using the gradient of the metric for synthesizing a shape from a measurement given a prior of a non-rigid deformation of that shape. The dogs and person’s models were taken from TOSCA database [4].

5.1 Shape-preserving denoising

In this example, two corresponding models of a palm are used. One model (denoted by $X_p$) is used as a prior while a random noise of normally distributed magnitude and uniformly distributed directions was added to every vertex of the second model (denoted by $X_n$). The cost function used in this example is

$$\text{cost}(X) = \sigma(X, X_p) + \lambda_1 d_{\text{ext}}(X, X_n) + \lambda_2 D_{\text{Vol}}(X, X_p, r_v)$$  \hspace{1cm} (5.1)

where

$$D_{\text{Vol}}(X, X', r) = \begin{cases} 
(|\text{vol}(X) - \text{vol}(X')| - r)^2 & |\text{vol}(X) - \text{vol}(X')| > r \\
0 & \text{otherwise}
\end{cases}$$  \hspace{1cm} (5.2)

and $\text{vol}(X)$ is the volume of the shape $X$. We were using $r_v = 0.05 \cdot \text{vol}(X_p)$, $\lambda_1 = 1.1 \cdot 10^{-4}$ and $\lambda_2 = 1.1 \cdot 10^{-1}$. The stress was calculated using only
pairs of points whose geodesic distance measured on the prior is at most \( K \), where \( K \) was chosen arbitrarily to 10 (for comparison, the shape’s diameter is 58).

The results are shown in Figure 5.1. The optimization was done in four resolutions of 331, 826, 1652 and 3304 vertices, with 200, 200, 200 and 500 iterations at each resolution correspondingly. The convergence is shown in Fig. 5.2.

Figure 5.1: Shape denoising. From left to right: the given prior, the unknown shape, the measurement - a noisy shape, denoising using the prior’s volume only, denoising using the prior’s intrinsic geometry and volume. Stress values are relative to the prior.

5.2 Sparse model fitting

In this example, we show how the stress can be reduced as a postprocessing of an arbitrary pair of nearly isometric shapes. Here we demonstrate it as a postprocessing for a shape completion. The inputs we use in this experiment are an initial shape \( X_0 \) (Fig. 5.3 left) and a set of 12 markers marked both on \( X_0 \) and on a target model (Fig. 5.3 second from left). The deformation was made using the algorithm of Ben-Chen and Weber [3] (Fig. 5.3 third from left), we denote its result as \( X_d \).
We minimize the following cost function
\[
    \text{cost}(X) = \sigma(X, X_0) + \lambda_1 \sum_{x_i \in X} d_{\mathbb{R}^3}(x_i, X_d)^2 + \lambda_2 D_{\text{vol}}(X, X_0, r_v). \tag{5.3}
\]
where \(d_{\mathbb{R}^3}(x_i, X_d)\) is the minimal Euclidean distance from \(x_i\) to a point on the faces of \(X_d\). We use \(r_v = 0.05 \cdot \text{vol}(X_0)\), \(\lambda_1 = 1\) and \(\lambda_2 = 10^{-5}\).

The stress between the prior and the unknown shape is \(1.11 \times 10^4\). The stress between the prior and the deformation’s result is \(1.08 \times 10^4\). After 800 iterations, the stress between the prior and the optimization’s result was \(2.4 \times 10^3\).

### 5.3 Bundle adjustment

The bundle adjustment problem is defined as follows. Let \(K\) denote the number of cameras looking at \(N\) points on the surface (we ignore the subject of occlusions and mismatches). Let \(\{s_i^{(k)}\}_{i=1}^N \subseteq \mathbb{R}^2, k \in 1, \ldots, K\) denote a set of projected coordinates obtained by projecting the points \(\{x_i\}_{i=1}^N \subseteq \mathbb{R}^3\) into a set of cameras with projection matrices \(P_k, k = 1, \ldots, K\), \(P_k (\tilde{x}_i) = \tilde{s}_i^{(k)}\), where \(\tilde{x}_i\) and \(\tilde{s}_i^{(k)}\) are homogenenous coordinates representation of \(x_i\) and \(s_i^{(k)}\) respectively.

Next, let \(\{r_i^{(k)}\}_{i=1}^N \subseteq \mathbb{R}^2, k \in 1, \ldots, K\) denote a set of noisy projected coordinates, \(r_i^{(k)} = s_i^{(k)} + n\), where \(n \sim N(0, \sigma)\) is Gaussian noise.
In this example, we try to reconstruct the unknown object $X$ given an intrinsically similar prior shape $X_p$, and correspondences functions $g^{(k)} : r_i^{(k)} \rightarrow X_p$ for $K = 2$.

The cost function we minimize is

$$
\text{cost}(X) = \sigma(X, X_p) + \lambda \sum_{k=1}^{K} \sum_{i=1}^{N} \|r_i^{(k)} - P_k(g^{(k)}(r_i^{(k)}))\|_2^2. 
$$

(5.4)

Deriving of the reconstruction error (the second term in Eq. 5.4) is explained in [25].

In this example we use $\sigma = 0.005$, the camera focal length is 10, $r_v = 0.05 \cdot \text{vol}(X_p)$, $\lambda = 1 \cdot 10^5$ and Optimization was stopped when the cost decrease between iterations was less than $10^{-8}$. Optimization was done on a single scale.

The camera arrangement and 2D images that are used as the reconstruction’s input are shown in Fig. 5.4. Results are shown in Fig. 5.3.
Figure 5.4: Left: The unknown shape, cameras location is indicated by the blue pyramids. Right: The shape vertices as viewed by the two cameras (top row) and after the noise is added (bottom row).
Figure 5.5: Unknown shape (top left), an intrinsically similar shape used as a prior (top right), Reconstruction result from noisy data in Fig. 5.4 with the intrinsic prior (bottom left - converged after 1501 iterations), and without the intrinsic prior (bottom right) - converged after 4265 iterations.
Chapter 6

Conclusions

In this thesis we have proposed a framework to solve inverse problems for which exists a prior knowledge of a shape intrinsically similar to the unknown one.

In order to solve those problems in the optimization framework, we have shown a method for differentiating the stress measured using the fast marching method. This method is based on differentiation of the FMM update step. Using the fast marching method results in consistent results but requires more adjustments since the distance function it computes is not always differentiable w.r.t. the coordinates of the shape’s vertices.

The method described in this thesis may be incorporated in any reconstruction problem which is solved by an optimization process whose optimization variables are the shape’s vertices locations. It has been demonstrated on various synthesis applications.

6.1 Future research directions

Among the future research direction we note the following.

**Weighted distances for runtime reduction** Assigning different weights to different pairwise distances in the stress computation may be used for the following changes. Run time may be reduced if only a subset of all pairwise distances will be calculated. An interesting question is, what subset of pairwise distances is sufficient in the sense that having low distortion of those
distances results in low distortion of all pairwise distances. Such selection may be based on skipping long distances, or skipping source vertices for which a distance map of a near neighboring vertex was already calculated.

**Weighted distances for removing low resolution artifacts**  Consider a two models of a straight finger and a bent one. The wrinkles that appear near the joints are not represented in those models. As a result, larger creases may appear. A possible solution is to assign lower weights to local distortions that occur in predetermined parts of the shape, such as around joints.

**Replacing additive distortion with a multiplicative one**  In the current stress definition, a distortion of $\delta$ in a long pairwise distance has much more effect on the value of the stress than a similar distortion of shorter distance. Setting the stress as some function of all multiplicative distortions will not give priority to relatively small distortions of large pathes. Moreover, when combined with assigning lower weights to distortions of vertices near joints, it may overcome the artifacts which result from the limited resolution of the mesh.
Appendix A

Derivation of the update step

Given a triangle formed by three vertices $x_A, x_B$ and $x_C$ ($A, B$ and $C$ are some indices), with corresponding edges $a, b$ and $c$, and corresponding coordinates column vectors of the vertices $\{x_A, x_B, x_C\}$.

Also given a source vertex $x_s$, the geodesic distances between $x_s$ and $x_A, x_B$, (denoted by $d_A, d_B$) and the derivatives of those distances $\left(\frac{\partial d_A}{\partial \alpha_i}, \frac{\partial d_B}{\partial \alpha_i}\right)^T$.

Next, we assume w.l.o.g. that $d_C$ is updated in the $x_A x_B x_C$ triangle.

If the triangle is not a virtual triangle, the derivatives of the edges’ length are calculated as follows.

$$
\frac{\partial}{\partial \alpha_i} \left( \begin{array}{ccc} a & b & c \end{array} \right) = \frac{\partial}{\partial \alpha_i} \left( \begin{array}{c} \|x_B - x_C\|_2 \\ \|x_A - x_C\|_2 \\ \|x_A - x_B\|_2 \end{array} \right)^T = \begin{pmatrix} (x_B - x_C) \cdot \hat{e}_j \left( \delta(B - i) - \delta(C - i) \right) \\ (x_A - x_C) \cdot \hat{e}_j \left( \delta(A - i) - \delta(C - i) \right) \\ (x_A - x_B) \cdot \hat{e}_j \left( \delta(A - i) - \delta(B - i) \right) \end{pmatrix}^T
$$

(A.1)

where $\delta(k)$ is the Kronecker delta, and $\hat{e}_j$ is a unit vector parallel to the $j^{th}$ axis. The differentiation of the length of edges in virtual triangles is described in Appendix B.

If $d_C$ is updated using Dijkstra’s update scheme, that is $d_C = \min\{d_A + b, d_B + a\}$,

$$
\frac{\partial d_C}{\partial \alpha_i} = \begin{cases} \frac{\partial d_A}{\partial \alpha_i} + \frac{\partial b}{\partial \alpha_i} & d_A + b \leq d_B + a \\ \frac{\partial d_B}{\partial \alpha_i} + \frac{\partial a}{\partial \alpha_i} & d_A + b > d_B + a \end{cases}
$$

(A.2)
From now and on we assume that $d_C$ is updated using the first case of Eq. 3.1 in which the condition holds. In order to differentiate Eq. 3.1, we first calculate the derivatives of $r$ and $s$ w.r.t. the lengths of the edges and of $d_C$ w.r.t. $d_A$ and $d_B$.

$$\frac{\partial r}{\partial \left( a, b, c \right)} = \left( \frac{-a}{c^2}, \frac{b}{c}, -\frac{b^2-c^2-a^2}{2c^2} \right), \quad (A.3)$$

$$\frac{\partial s}{\partial \left( a, b, c \right)} = \left( \frac{r}{c}, \frac{bc-rb}{bc-sc}, \frac{r}{s} \frac{b^2-c^2-a^2}{2c^2} \right), \quad (A.4)$$

$$\frac{\partial d_C}{\partial \left( d_A, d_B \right)} = \left( 1 - \frac{r}{c} + \frac{sv}{c\sqrt{c^2-v^2}}, \frac{r}{c} - \frac{sv}{c\sqrt{c^2-v^2}} \right). \quad (A.5)$$

Next, we differentiate Eq. 3.1 w.r.t. the edge length, using the chain rule and Eqs. A.3 and A.4, yielding

$$\frac{\partial d_C}{\partial \left( a, b, c \right)} = \begin{pmatrix} v \frac{\partial r}{\partial a} + \frac{\sqrt{c^2-v^2}}{c} \frac{\partial s}{\partial a} \\ v \frac{\partial r}{\partial b} + \frac{\sqrt{c^2-v^2}}{c} \frac{\partial s}{\partial b} \\ v \left( -\frac{r}{c} + \frac{\partial r}{\partial c} \right) + \frac{\sqrt{c^2-v^2}}{c} \frac{\partial s}{\partial c} + s \left( \frac{1}{\sqrt{c^2-v^2}} - \frac{c^2-v^2}{c^2} \right) \end{pmatrix}^T. \quad (A.6)$$

Combining Eqs. A.6, A.5 and A.1, we obtain

$$\frac{\partial d_C}{\partial \alpha_i} = \frac{\partial d_C}{\partial \left( d_A, d_B \right)} \frac{\partial d_C}{\partial \left( d_A \right)} \left( \frac{\partial d_A}{\partial \alpha_i} \right) + \frac{\partial d_C}{\partial \left( a, b, c \right)} \frac{\partial d_C}{\partial \alpha_i} \left( a \ b \ c \right)^T. \quad (A.7)$$
Appendix B

Derivation of triangle unfolding

First we review the triangle splitting process, according to the example described in Fig. B.1. For convenient, we use $U_0, U_1, U_2$ and $U_3$ to denote the vertices of the original triangle and the vertex that creates the two virtual triangles. $u_1, u_2$ and $u_3$ denotes the vertices of the last triangle to be unfolded. The unfolding is done according to Algorithm 1. The algorithm is based on defining a 2D coordinate system in which $U_0$ is at the origin, $U_2$ is located on the $\hat{x}$ axis and the $\hat{y}$ coordinate of $U_1$ (denoted by $(U_1)_y$) is non-negative. Iteratively, triangles are unfolded to this plane until a vertex that forms two acute triangles with $U_0, U_1$ and $U_2$ is found. The algorithm uses a function named $getNextTrig$ that given a triangle and an edge returns a the third vertex of the other triangle containing the edge. If no such vertex exists (that is, the edge is on the boundary of the mesh) it returns an empty set. We denote the $\hat{x}$ and $\hat{y}$ coordinates of a vertex $u_i$ in the above coordinates system by $u_x^i$ and $u_y^i$ respectively (or $\tilde{u}_i$ as a $2 \times 1$ column vector), and the location of vertex $u_i$ in the 3D mesh’s space by $\pi_i$.

The differentiation of the virtual triangle edges’ lengths is done in parallel to the unfolding of the triangles. This is done as follows. Derivatives the edges lengths (step 1 of Algorithm 1) w.r.t. all vertices locations is done as described in Eq. A.1.

Next, deriving the initial values of $u_x^1, u_y^1, u_x^2$ and $u_y^2$ (steps 4 and 5 of
Figure B.1: An example of unfolding triangles in order to split an obtuse triangle to two virtual triangles with acute angles. The superscripts denote different stages, the different names of a vertex denotes the variables pointing to this vertex in different iterations. $u_3$ of the first iteration crates an obtuse angle with $U_0$ and $u_1$ of the first iteration and therefore a second iteration is needed.

Algorithm 1) is as follows.

$$\frac{\partial u_2^x}{\partial \alpha_i} = \frac{\partial E_1}{\partial \alpha_i}, \quad \frac{\partial u_2^y}{\partial \alpha_i} = 0. \tag{B.1}$$

Using

$$\frac{\partial u_1^x}{\partial \left( E_0, E_1, E_2 \right)} = \left( -\frac{E_0}{E_1}, \frac{E_1^2 - E_2^2 + E_0^2}{2E_1^2}, -\frac{E_2}{E_1} \right), \tag{B.2}$$

and the chain rule we obtain

$$\frac{\partial u_1^x}{\partial \alpha_i} = \sum_{k=0}^{2} \frac{\partial u_1^x}{\partial E_k} \frac{\partial E_k}{\partial \alpha_i}, \quad \frac{\partial u_1^y}{\partial \alpha_i} = \frac{1}{u_1^x} \left( E_2 \frac{\partial E_2}{\partial \alpha_i} - u_1^x \frac{\partial u_1^x}{\partial \alpha_i} \right). \tag{B.3}$$

We define $\frac{\partial u_1^x}{\partial \alpha_i}, \frac{\partial u_1^y}{\partial \alpha_i}, \frac{\partial u_2^x}{\partial \alpha_i}$ and $\frac{\partial u_2^y}{\partial \alpha_i}$ as the derivatives of $u_1^x, u_1^y, u_2^x$ and $u_2^y$ w.r.t. the coordinates of vertex $i$. 

36
Next, deriving $\beta$ (step 12),

$$\frac{\partial \beta}{\partial \alpha_i} = \frac{\partial}{\partial \alpha_i} \frac{u_2^x - u_1^x}{\|\mathbf{u}_1 - \mathbf{u}_2\|_2} = \frac{\left(\|\mathbf{v}\|_2^2 \left(\frac{\partial u_2^x}{\partial \alpha_i} - \frac{\partial u_1^x}{\partial \alpha_i}\right)\right) + (u_2^x - u_1^x) (u_2^y - u_1^y) \left(\frac{\partial u_2^y}{\partial \alpha_i} - \frac{\partial u_1^y}{\partial \alpha_i}\right)}{\|\mathbf{u}_1 - \mathbf{u}_2\|_2} \quad (B.4)$$

We define $e_1 = \|\mathbf{v}_2 - \mathbf{v}_3\|$ and $e_3 = \|\mathbf{v}_1 - \mathbf{v}_2\|$. Deriving $e_1$, $e_2$ and $e_3$ w.r.t. $\alpha_i$ is done as described in Eq. A.1.

Deriving $x_t$ (steps 14) w.r.t. $e_1$, $e_2$ and $e_3$,

$$\frac{\partial x_t}{\partial \left( e_1, e_2, e_3 \right)} = \left( -\frac{e_1}{e_3}, \frac{e_2}{e_3}, \frac{e_1^2 - e_2^2 + e_3^2}{2e_3} \right). \quad (B.5)$$

Using Eq. B.5 and the chain rule we obtain

$$\frac{\partial x_t}{\partial \alpha_i} = \sum_{e \in \{e_1, e_2, e_3\}} \frac{\partial x_t}{\partial e} \frac{\partial e}{\partial \alpha_i} \quad (B.6)$$

Using $\frac{\partial y_t}{\partial x_t} = -\frac{x_t}{y_t}$, $\frac{\partial y_t}{\partial e_2} = \frac{e_2}{y_t}$, Eq. B.6 and the chain rule we obtain

$$\frac{\partial y_t}{\partial \alpha_i} = \frac{\partial y_t}{\partial e_2} \frac{\partial e_2}{\partial \alpha_i} + \frac{\partial y_t}{\partial x_t} \frac{\partial x_t}{\partial \alpha_i} \quad (B.7)$$

Finally, deriving $\mathbf{v}_3$ (step 16) is done as follows

$$\begin{pmatrix} \frac{\partial u_3^x}{\partial \alpha_i} \\ \frac{\partial u_3^y}{\partial \alpha_i} \end{pmatrix} = \begin{pmatrix} 1 & \cot(\beta) \\ -\cot(\beta) & 1 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} + \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} \frac{\partial x_t}{\partial \alpha_i} \\ \frac{\partial y_t}{\partial \alpha_i} \end{pmatrix} + \begin{pmatrix} \frac{\partial u_2^x}{\partial \alpha_i} \\ \frac{\partial u_2^y}{\partial \alpha_i} \end{pmatrix} \quad (B.8)$$

if the iteration step is repeated, we update the derivatives of $\tilde{u}_2$ or $\tilde{u}_1$ using $\frac{\partial \tilde{u}_1}{\partial \alpha_i} = \frac{\partial \tilde{u}_2}{\partial \alpha_i}$ if the condition in step 21 is true and $\frac{\partial \tilde{u}_2}{\partial \alpha_i} = \frac{\partial \tilde{u}_2}{\partial \alpha_i}$ otherwise.

When reaching step 18, calculating the derivatives of the distance between $U_3$ and $U_2$ (denoted by $U_3U_2$) w.r.t. $\alpha_i$ is done as follows.

$$\frac{\partial U_3U_2}{\partial \alpha_i} = \frac{U_3^x - U_2^x}{U_3U_2} \left( \frac{\partial U_2^y}{\partial \alpha_i} - \frac{\partial U_3^y}{\partial \alpha_i} \right) + \frac{U_3^y - U_2^y}{U_3U_2} \frac{\partial U_3^y}{\partial \alpha_i} \quad (B.9)$$

The derivatives of the distances between $U_3$ to $U_0$ and $U_3$ to $U_1$ w.r.t. $\alpha_i$ are done in a similar manner.
**Algorithm 1:** Splitting an obtuse angled triangle to virtual acute triangles.

**Input:** Numerical grid $\mathbf{U}$ and an acute triangle $U_1U_0U_2$

**Output:** A splitting vertex $U_3$ and the lengths of $U_0U_3$, $U_1U_3$ and $U_2U_3$ in the $U_1U_0U_2$ plane

**Initialization**

1. $E_0 \leftarrow ||\mathbf{U}_1 - \mathbf{U}_2||$; $E_1 \leftarrow ||\mathbf{U}_0 - \mathbf{U}_2||$; $E_2 \leftarrow ||\mathbf{U}_0 - \mathbf{U}_1||$
2. $u_1 \leftarrow U_1$; $u_2 \leftarrow U_2$
3. $\tilde{u}_0 \leftarrow (0, 0)^T$
4. $\tilde{u}_1 \leftarrow (E_2 \cos(\angle U_2U_0U_1), E_2 \sin(\angle U_2U_0U_1))^T$
5. $\tilde{u}_2 \leftarrow (E_1, 0)^T$
6. $\text{trig}_{\text{curr}} \leftarrow U_1U_0U_2$
7. $\gamma \leftarrow \frac{u^t_1}{u^t_1}$

**Iteration**

8. while true do

9. $u_3 \leftarrow \text{getNextTrig}(\text{trig}_{\text{curr}}, u_1u_2)$
10. if $u_3 == \emptyset$ then
    11. \hspace{1em} RETURN $\emptyset$
12. \hspace{1em} $\tilde{u}_1 \leftarrow (u^p_2, u^p_1); \beta \leftarrow \angle u_2u_1u_t$
13. \hspace{1em} $e_2 \leftarrow ||\tilde{u}_3 - \tilde{u}_1||$
14. \hspace{1em} $x_t \leftarrow e_2 \cos(\angle u_3u_1u_2)$
15. \hspace{1em} $y_t \leftarrow e_2 \sin(\angle u_3u_1u_2)$
16. \hspace{1em} $\tilde{u}_3 \leftarrow \tilde{u}_1 + \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix}$
17. if $u^x_3 > 0$ AND $\frac{u^p_1}{u^x_3} > \tan(\gamma)$ then
18. \hspace{1em} $U_3 \leftarrow u_3$
19. \hspace{1em} $\begin{pmatrix} U_3U_2 & U_3U_0 & U_3U_1 \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{(u^x_3 - u^x_2)^2 + u^p_2^2} & \sqrt{u^x_1^2 + u^p_3^2} & \sqrt{(u^x_3 - u^x_1)^2 + (u^p_3 - u^p_1)^2} \end{pmatrix}$
20. \hspace{1em} RETURN
21. if $u^x_3 < 0$ then
22. \hspace{1em} $u_1 \leftarrow u_3$
23. else
24. \hspace{1em} $u_2 \leftarrow u_3$
25. \hspace{1em} $\text{trig}_{\text{curr}} \leftarrow u_1u_2u_3$

38
Appendix C

Derivation of other properties

The Hausdorff distance gradient

In this work we differentiated a non-symmetric $L_2$ variant of the Hausdorff distance (Eq. 2.4),

$$d_H(Z, Y) = \sum_{i=1}^{N} (d_{R^3}(z_i, Y))^2. \quad (C.1)$$

where $d_{R^3}(z_i, Y)$ is the minimal Euclidean distance between $z_i$ and a point on a face of $Y$, denoted by $\hat{y}$.

$\hat{y}$ is found by checking the minimal distance to each of $Y$’s triangles relatively close to $z_i$. This set of triangles is found using an approximate nearest neighbor algorithm over $z_i$ and $Y$’s vertices.

Assuming that $\hat{y}$ is contained in a single triangle consisted from the vertices $y_1, y_2, y_3$, $d_{R^3}(z_i, \hat{y})$ is a quadratic equation depending on the coordinates of $y_1, y_2, y_3$ and $z_i$, and therefore it’s differentiation according to the coordinates of $z_i$ is straightforward. Refer to [13] for details of this quadratic equation.

In the case that $\hat{y}$ is contained in more than one triangle of $Y$, we average the gradient of $d_{R^3}(z_i, \hat{y})$ over all the triangles containing $\hat{y}$.
The area similarity gradient

Consider a surface $X$. The surface area of $X$ is defined by

$$\text{area}(X) = \sum_{x_1, x_2, x_3 \in \text{trigs}(X)} \frac{1}{2} \| (x_2 - x_1) \times (x_3 - x_1) \|_2, \quad \text{(C.2)}$$

where $\text{trigs}(X)$ is the triangulation of $X$.

The differentiation of Eq. C.2 is cumbersome but simple and therefore omitted.

Given a second surface $Y$, we define the difference between the areas of $X$ and $Y$, by $D_{\text{area}}(X, Y) = (\text{area}(X) - \text{area}(Y))$. The derivatives of the squared area difference is

$$\frac{\partial D_{\text{area}}(X, Y)^2}{\partial \alpha_i} = D_{\text{area}}(X, Y) \frac{\partial \text{area}(X)}{\partial \alpha_i} \quad \text{(C.3)}$$

We also define an area difference with tolerance $r$ as

$$D_{\text{area}}(X, Y, r) = \begin{cases} (|\text{area}(X) - \text{area}(Y)| - r)^2 & |\text{area}(X) - \text{area}(A)| > r \\ 0 & \text{otherwise} \end{cases}, \quad \text{(C.4)}$$

for which the deriving is also straightforward.

The volume similarity gradient

Consider a surface $X$. The volume of $X$ is calculated using Green’s theorem

$$\text{vol}(X) = \iiint_X dv = \int_S(X) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T N(s) ds$$

$$= \sum_{t \in \text{trigs}(X)} \text{area}(t)(N(t))_x, \quad \text{(C.5)}$$

where $S(X)$ is the surface of $X$, $N(s)$ is the normal vector at point $s$ pointing outwards, $(N(t))_x$ is the first entry of the normal to triangle $t$, area$(t)$ is the area of triangle $t$, and trigs$(X)$ is the triangulation of $X$. 

40
Since, area(t) = \( \frac{1}{2} \| (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1) \|_2 \) and \( (N(t))_x = ((\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1))_x \) the differentiation of area(t)(N(t))_x is cumbersome and simple and therefore omitted. The differentiation of Eq. C.5 is as follows.

\[
\frac{\partial \text{vol}(X)}{\partial \alpha_i} = \sum_{t \in \text{trigs}(X), x_i \in \mathcal{X}} \frac{\partial}{\partial \alpha_i} \text{vol}(t)(N(t))_x.
\] (C.6)

Given a second surface \( Y \), we define the difference between the volumes of \( X \) and \( Y \) by \( D_{\text{vol}}(X, Y) = (\text{vol}(X) - \text{vol}(Y)) \). The derivatives of the squared volume difference is

\[
\frac{\partial D_{\text{vol}}(X, Y)^2}{\partial \alpha_i} = D_{\text{vol}}(X, Y) \frac{\partial \text{vol}(X)}{\partial \alpha_i}
\] (C.7)

We also define a volume difference with tolerance \( r \) as

\[
D_{\text{vol}}(X, Y, r) = \begin{cases} 
(|\text{vol}(X) - \text{vol}(Y)| - r)^2 & \text{if } |\text{vol}(X) - \text{vol}(Y)| > r \\
0 & \text{otherwise}
\end{cases}
\]
(C.8)

for which the deriving is straightforward.
Bibliography


רגולרייזציה על-מ.GetObject Shelley בגהית

היפוך הכלולות נופיים לא קשיות

ויוחאי ש. דביר
רגולריותzia אצמציתית של בטיה
היפך: הכללון גופים לא קשיחים

ת揚גר על מחקר

לשם مليק חלך של הדרישות תכובצת והזורה
מעניט לכלים בו מועדון ומשתכר

והוא י. דבך

המותס לטכנולוגיה – מכון טכנולוגי לישראל
אדר הותס, ט"ו בחודש מאי 2009
המשחזר מששה ומגביר פרופ"י, יובל רבני בפתקול צהוב ומעטしてしまう.

ברצוני לה внיש את תודדיל השמייש לפורפ"י, יובל רבני על התודדיל והתרוממה. המדד
מק持ちים תודדיל קריפר של חלומל על התודדיל המסמך את ערודיالأור.
יריתי רכזה תודדיל למשתתפים לכל תחביל על שלוחי אזורים לאריך הצדים ואותו
ולכים צהתה.

אני מודע לעכביון על התודדיל הסיפת תודדיל בוושלהלמה.
תקציר

ישנם מקדימים רבים שבראשה האוניברסיטה ובראשי מחלוצותינו. אך הצלחתנונית
לשהאר את המנהל על האיניקט تعدامتו על סמכים ומדל שגוןurbedיקט
(ולך). עם זאת, אנו ניסו לשדר את המנהלה שהמדים ממידה של
איניקט ממידתחדולים ומדידתשקטונות עיוונית. וברקחט התמדת
של שחרת איניקט ממידתüsseldorf נтельно של האיניקט.Con בדיל
שמטרקטיביים לשאר מבית ממידה או ממידה או יותר. ביצת עליה ממידת
ишע:

מותרי התחנה של תחנה בן עיר איניקט ממידי ממידתותיה היא תחפ שביך
ויך שחרת השחרור של תחנהولد המנהל לשדר תחנה לממידתותיה. במקסם ריכם. לא
ניק תחנה מאתחראות ובחזרה מראיציוניאים שינם שאר
למדוד ממידות הצהרה. שיש обращ ממידת תופס על האיניקט 판단 ממידי חרט
אף והבחור את הצהרה. ממך אתר פלעם מיבות תחנה לממידתותיה.
לשמאר את האיניקט 판단 ממידי חרטת של السودה ו_weapon שמידי בינ
וכם היו האיניקט חמש שנים למיזוד תחנה.

ישני ימים שבחר שלוחת שחרת הס所謂ה תחנה אל קשישה. הסיווכ
יה אסף. נטוש. עמדה שסובב את החלק איניקטס ריבס שמידת שעת.
אもらえ מברת את הספרנולגיה שחרת לכבוד מיקוד תחנה.
בשם י효 מברת את חלקות ששת חמשון מחבל חמשון
של בינ אסף, מברת כי בין קצב. זה על ק. של קראס.Manיקטס למ
מונח אינסימף של עיוותים או קושיוס אופירי. חומ עמותה של תחנה איניקט.
שר יראיציוניאים שינם איניקטינו שינם
בברך클כל לשאר מברת ע"י קשישה אל קשישה שלمخם איניקט.
אותי צמות (יאירטרוס, הסיפור השולייה ואפשים ריבים, א"י על פי שלא
ידות inícioו המאיצייק שזאתון מסיס לשחרר בידות לא עד ממיתות שנה א
נמצאות (למשל, מי הנהדר בתמונה אוריהם המה של), נמח全て השארבריקט ונדמה

8
למען לצביע את תודוק האפשרויות היא ישהל יניקית יניקית גיא שבעת לשתים מים שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשתים צירוף שבעת לשתים גיא שבעת לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה לשה淋巴ymph
מאמרות קדוד והتعليقات שלהם של מחקרים נוספים מחברתיים לשני מתקנים מכנים.㋠

פצה נפשות שני מחקרים נוספים מחברתיים לשני מתקנים מכנים.㋠

פצה נפשות שני מחקרים נוספים מחברתיים לשני מתקנים מכנים.
שיני תוספ נגבה מאיצורים זרים וממשיכים בזילות האיביקיטון. מבזידה והשישה
וחירת זומת לקחה אחר, והזח דחף בשלב צלצול בטין משולשיות וירטואליות.
וחירת כיון דרי זורת, ואיזי השכינה אילзов הקקברוקים וזר הפרכסים של מאיצורים.
ולש יירטיוול לקחה אחר. יזא שרד לאיצורים בטין משולשיות וירטואליות חלינים.
המעמיסים על מחסן של פומז ממסור יזא ли or איצור בעיתון המושך.