

Computational Surface Flattening: A Voxel-based Approach

Ruth Grossmann **Nahum Kiryati***

Dept. of Electrical Engineering–Systems
Tel Aviv University
Ramat Aviv 69978, Israel

Ron Kimmel

Dept. of Computer Science
Technion–Israel Institute of Technology
Haifa 32000, Israel

Abstract

A voxel-based method for flattening a surface in 3-D space into 2-D while best preserving distances is presented. Triangulation or polyhedral approximation of the voxel data are not required. The problem is divided into two main parts: Voxel-based calculation of the minimal geodesic distances between points on the surface, and finding a configuration of points in 2-D that has Euclidean distances as close as possible to these distances. The method suggested combines an efficient voxel-based hybrid distance estimation method, that takes the continuity of the underlying surface into account, with classical multi-dimensional scaling (MDS) for finding the 2-D point configuration. The proposed algorithm is efficient, simple, and can be applied to surfaces that are not functions. Experimental results are shown.

Keywords: surface flattening, geodesic distance estimation, multidimensional scaling, voxel representation, texture mapping

*Corresponding author, e-mail: nk@eng.tau.ac.il, Fax: +972 3 640 7095.

1 Introduction

Surface flattening is the problem of mapping a surface in 3-D space into 2-D. Given a digital representation of a surface in 3-D, we wish to map each surface point into the plane, such that the distance between each pair of points in the plane is as close as possible to the corresponding geodesic distance between the points on the 3-D surface.

It is known that flattening a surface in 3-D space into a 2-D plane introduces metric distortions unless the surface has zero Gaussian curvature [7, 25]. Gaussian curvature is the product of the maximum and minimum values of the normal curvatures of the surface at a given point. Gaussian curvature is an isometric invariant, so two surfaces are isometric (i.e. preserve interpoint distances) only if they have the same Gaussian curvature. For example, a plane and a cylinder both have zero Gaussian curvature, since for the plane both principal curvatures vanish and for the cylinder one principal curvature vanishes. Therefore, a plane bent into a cylindrical shape can obviously be flattened with no distortion. On the other hand, there is no isometry of the sphere onto the plane, since their Gaussian curvatures are different. This is the dilemma of the map maker: the intrinsic geometry of the earth's surface is misrepresented by any flat map [29]. Since general surfaces are likely to have nonzero Gaussian curvatures, their flattening necessarily introduces some distortion.

The need for surface flattening arises in many scientific areas, notably in medical imaging and in computer graphics. For example, the cortical surface is highly convoluted, complicating visualization and registration of MRI or fMRI data. However, after defining appropriate cutting locations, the cortical surface can be unfolded with relatively small distortion. One of the first flattening algorithms was proposed in [27]. More efficient methods were later

developed: In [8] local metric properties were preserved in the flattening process but large global distortions sometimes occurred; In [5, 6] a global energy functional was minimized; In [1, 14, 11] angle preserving mappings (instead of distance preserving mappings) were used for flattening. Surface unfolding using level-set methods was presented in [15]. In computer graphics, there is a need for area preserving texture mapping (e.g. [3] and [23]). Surface flattening can be used to map a 2-D flat texture image onto a 3-D surface by first flattening the 3-D surface (see [32] for the case of triangulated surfaces). The mapping of the 2-D texture image onto the flattened surface is then immediate, and followed by mapping back the textured flattened surface onto the surface in 3-D space.

Following the progress in shape-from-X techniques, high quality acquisition of 3-D shape with its texture is nowadays easy to accomplish. One successful application of this technology is the 3-D photography of artifacts and natural objects for virtual museums and exhibitions. Surface flattening can serve as a useful tool for the analysis and comparison of combined shape and texture models. In particular, flattening can bring into a standard form patterns that appear on differently shaped surfaces.

This paper combines a voxel-based geodesic distance estimator with an efficient dimensionality reduction algorithm to obtain a fast, practical method for surface flattening. The flattening process is thus divided into two steps. First, the minimal geodesic distances between points on the surface are estimated. Then, a planar configuration of points that has Euclidean interpoint distances that are *globally* as close as possible to the corresponding minimal geodesic distances is determined. The method presented can be applied to surfaces that are not functions. A unique feature of our approach is that the algorithm operates directly on voxel data, which is the natural representation provided by many imaging devices,

thus avoiding the need for an intermediate triangulated representation of the surface. This may simplify system design, eliminate some representation errors and lead to computational savings.

2 Voxel-Based Geodesic Distance Estimation¹

Finding minimal geodesic distances between points on a continuous surface is a classical problem in differential geometry. However, given *digital* 3-D data, purely continuous differential methods are impractical due to the inherent need for interpolation, the vast computational cost, and the risk of convergence to a local minimum in the solution space.

The common practice is to transform the digital (voxel-based) surface representation into a triangulated surface representation (see [22] for an efficient triangulation method) prior to distance calculation. In [24] the distances between a given source vertex on the surface and all other surface vertices are computed in $O(N^2 \log N)$ time, where N is the number of edges on the triangulated surface. In [31] an algorithm that is simpler to implement is given, but the algorithm runs in exponential time. In [20] and [19] the fast marching method on triangulated domains is introduced. The method computes the minimal distances between a given source vertex on the surface and all other surface vertices in $O(N \log N)$ time, where N is the number of triangles that represent the surface.

An algorithm for geodesic distance estimation on surfaces in 3-D, that uses the voxel representation without a need to first triangulate or interpolate the surface, was presented in [21].

¹Parts of this section, including Figs. 1-3, are reprinted from *Pattern Recognition* **26**: N. Kiryati and G. Székely, “Estimating Shortest Paths and Minimal Distances on Digitized Three-Dimensional Surfaces”, pp. 1623-1637, ©1993, with permission from Elsevier Science.

The method is based on a high precision length estimator for continuous 3-D curves that have been digitized and are given as a 3-D chain code [18], and on the representation of the digital surface as a sparse graph. The vertices and edges of the graph respectively correspond to voxels and to 3-D digital neighborhood relations between voxels. The shortest path between two points on the surface is associated with the path that has the shortest length estimate and that length estimate corresponds to the geodesic distance between the two points. The computational complexity of the method is the same as that of finding shortest paths on sparse graphs, i.e. $O(N \log N)$, where N is the number of surface voxels. Subsection 2.1 is a brief overview of 3-D length estimation. The application of 3-D length estimation to the estimation of minimal distances on surfaces is outlined in subsection 2.2.

2.1 3-D length estimation

The 3-D length estimation problem is to estimate the length of an underlying continuous 3-D curve given its chain code. The estimator described in [18] is based on link classification in the 26-directional chain code representation of the curve. It is of the general form

$$\hat{L} = \Psi_1 N_1 + \Psi_2 N_2 + \Psi_3 N_3, \quad (1)$$

where N_1 is the number of direct links in the chain code, i.e., links that are parallel to one of the three main axes, and N_2 and N_3 are respectively the number of minor and major diagonal links in the chain code (see Figure 1). Ψ_1 , Ψ_2 and Ψ_3 are weights, and \hat{L} is the estimated length.

Assuming that the digitization is sufficiently dense so that the curve is reasonably straight

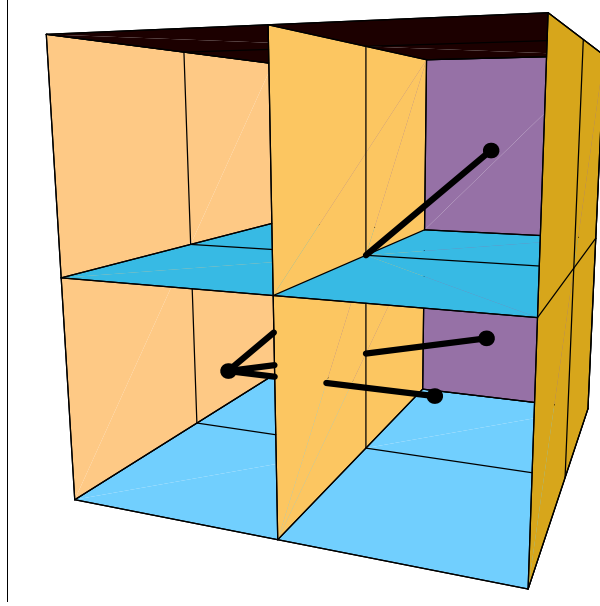


Figure 1: Link types in a 26-directional 3-D chain code: a direct link (parallel to one of the main axes), a minor diagonal link and a major diagonal link.

within one or two voxels, the naive selection of weights $\Psi_1 = 1$, $\Psi_2 = \sqrt{2}$ and $\Psi_3 = \sqrt{3}$ would lead to consistent overestimation of the length. Extending planar perimeter estimation theory to 3-D, in [18] the frequencies of appearance of direct, minor and major diagonal links in the 3-D chain code of an infinite straight line at any given orientation were determined. Assuming uniform distribution of orientations, the weights were then set to obtain an *unbiased* estimator that achieves the least *RMS* error possible. That estimator is

$$\hat{L} = 0.9016N_1 + 1.289N_2 + 1.615N_3. \quad (2)$$

For infinite straight lines, the estimation error is lower bounded by -9.84% and upper bounded by 4.59%, and the *RMS* error is 2.88%. The unbiasedness of the estimator implies that if the direction of the tangent varies along the curve, local estimation errors cancel out and much lower total estimation errors are obtained. Figure 2 shows the average, maximum and

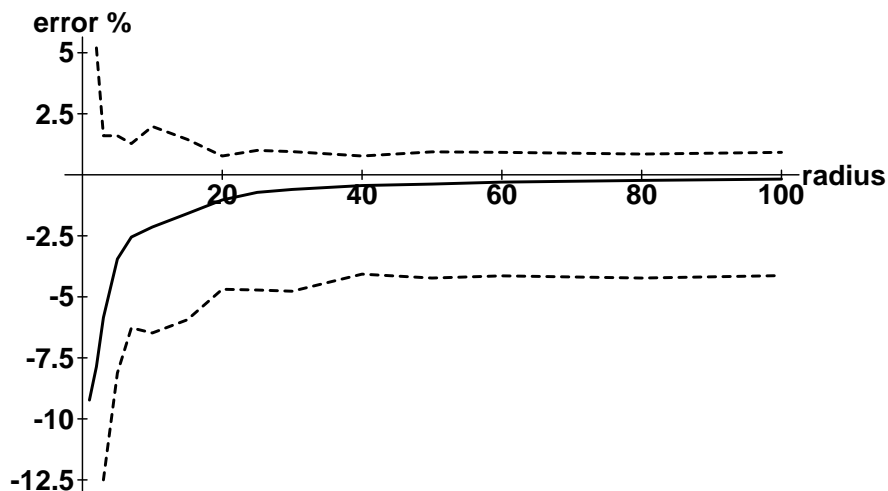


Figure 2: Perimeter estimation of circles in 3-D using the unbiased, minimum RMS error, 26-directional link classification estimator. The solid curve shows the average error (of 100 experiments) as a function of radius. The dashed curves show the minimum and maximum errors encountered.

minimum errors as a function of radius in perimeter estimation of 100 randomly oriented and translated circles in 3-D space [18]. As expected, the average error is very small and approaches zero as the radius increases.

Sophisticated and more accurate voxel-based 3-D length estimators have recently been presented in [17]. They rely on the cube-quantization method and are thus inherently better behaved in the mathematical sense [16]. These advanced 3-D length estimators have not been used in this research, since their incorporation in a scheme for geodesic distance estimation is not straightforward.

2.2 Finding minimal distances on a surface

Suppose that the surface of an object is given in digitized form as a set of voxels in a three dimensional array. Assume that a voxel belongs to the digitized surface if it is traversed

by the underlying continuous surface and if at least one of its direct (i.e., 6-directional) neighbors is a “background” voxel.

A path that runs on the surface and connects two surface points can be represented in digital form by the set of surface voxels that the path traverses. This set of voxels, the “digital path”, can be represented by a 6-directional chain code. By allowing full diagonal connectivity a 26-directional chain code representation is obtained.

If it is assumed that the digitization grid is fine with respect to the curvature of a path, such that the path is roughly linear within most of the voxels, its length can be estimated from the digital representation using a three dimensional length estimator. This is the basis for estimating minimal distances and shortest paths on digitized three dimensional surfaces: the shortest path between two points is assumed to be associated with the digital path whose length estimate is the shortest, and that estimate is taken as an estimate of the minimal distance. In this hybrid discrete-continuous technique, interpolation of the digitized surface data is implicit in the design of the length estimator (hence computationally costless) rather than an explicit algorithmic stage.

The digital surface is viewed as a three dimensional graph, in which each vertex corresponds to a surface voxel. Given any specific definition of surface connectivity, pairs of vertices that correspond to pairs of neighboring surface voxels are connected by arcs in the graph. 26-directional connectivity is defined, hence every surface voxel is connected by an arc to every surface voxel in its (26) neighborhood². If each arc is assigned a cost according to the weight

²In the digitization of surfaces that fold onto themselves, as in MRI brain data, it might happen that two neighboring surface voxels correspond to parts of the underlying continuous surface that are actually distant from each other. In the sequel we assume that the digital surface data used as input to the flattening algorithm is topologically correct, implying that this exception has been eliminated by preprocessing. This is a common practice in the analysis of MRI data.

of its link type (direct, minor diagonal or major diagonal) in the 3-D length estimator, then estimating minimal distances and shortest paths on a continuous surface given in digitized form reduces to finding the shortest path in a graph.

Algorithms for finding minimal distances and shortest paths in graphs are well known; for an overview see [13]. Here all arcs have positive weights, so in principle the algorithm of Moore and Dijkstra can be applied. Let N denote the number of vertices in the graph, i.e., the number of surface voxels; then the shortest paths between a source vertex to all other vertices can be found in $O(N^2)$ time. It is however clear that surface graphs are sparse: the number of arcs emanating from any vertex is upper bounded by 26. (In practice, since it is assumed that the digitization is fine, a surface is roughly planar within most small neighborhoods, so the number of arcs emanating from a vertex is about 8). The computational complexity can thus be reduced, and the minimal distance from a vertex to all others can be estimated in $O(N \log M)$ time, where M is the total number of arcs in the graph, and is proportional to N [13]. This means that the minimal distances can be found in $O(N \log N)$ time. The key to computational efficiency in the implementation is the use of a priority queue (heap) data structure [28]. The minimal distances between *every* pair of vertices can obviously be determined in $O(N^2 \log N)$ time by N applications of the algorithm. Observe however, that the shortest paths and minimal distances in the graph are just an *approximation* to the shortest paths and minimal distances on the continuous surface. The quality of the approximation reflects the quality of the 3-D length estimator [18].

The operation of the algorithm is demonstrated in Fig. 3. The surface resembles a hilly terrain, with a river-like obstacle that divides it into two regions connected by a narrow “bridge”. The black path is the estimated shortest path from the source voxel to the desti-

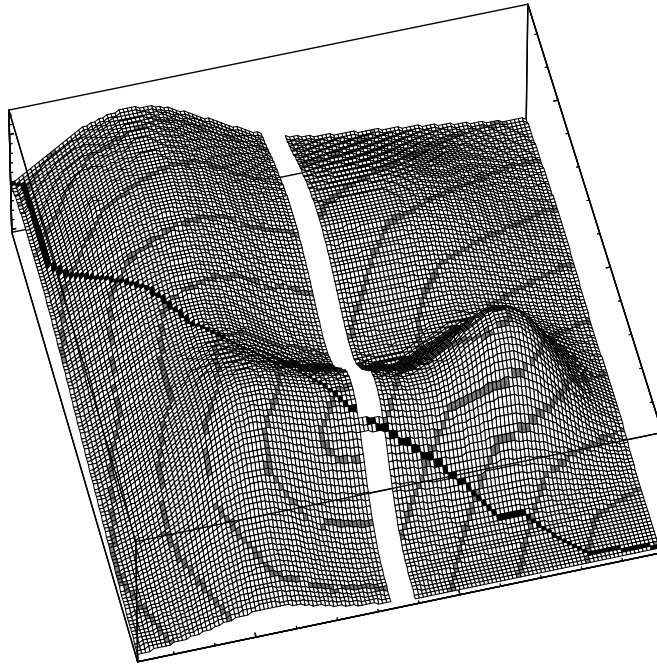


Figure 3: Finding the shortest path between two voxels using the algorithm of [21]. The surface is a hilly terrain, cut into two regions connected by a narrow “bridge” through which the path from the source (bottom-right) to the destination (top-left) must pass. The black path is the estimated shortest path from the source voxel to the destination voxel; the grey contours represent equal estimated distance from the source. Observe the diffraction-like pattern of the equal estimated distance contours beyond the bridge.

nation voxel; the grey contours represent equal estimated distance from the source. Observe the diffraction pattern beyond the narrow bridge.

Having obtained estimates of the minimal geodesic distances between points of the surface in 3-D space, we proceed to finding a corresponding configuration of points in 2-D with inter-point Euclidean distances that are as close as possible to the respective geodesic distances.

3 Flattening by Multidimensional Scaling

A *dissimilarity matrix* is a matrix that stores measurements of dissimilarity among pairs of objects. Multidimensional scaling (MDS) is a common name for a collection of data analytic

techniques [2] for finding a configuration of points in a low-dimensional Euclidean space that will represent the given dissimilarity information by the interpoint Euclidean distances.

Since usually no configuration of points can precisely preserve the information, an objective function is defined and the task is formulated as a minimization problem. So given a set of items $\{\bar{\mathbf{x}}_k\}$ with dissimilarities $\delta(k, l)$ between items $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{x}}_l$, the goal is to find p -dimensional data vectors $\{\hat{\mathbf{x}}_k\}$, $\hat{\mathbf{x}}_k \in \mathfrak{R}^p$ that have Euclidean distances $\{d(k, l)\}$ that approximate $\{\delta(k, l)\}$ well.

Many variants of MDS exist, differing in the objective function and optimization algorithms. These can be divided into two basic classes: metric and non-metric MDS. In metric MDS the original distance (or dissimilarity) matrix is approximated. Non-metric MDS deals with ordinal data or data in which only the order of the distances (dissimilarities) needs to be preserved. There is increasing interest in the development and use of statistical dimensionality reduction techniques for image analysis [26, 30].

MDS is the second step in our surface flattening approach. Once the minimal geodesic distances between points on the surface have been estimated, they are represented as a dissimilarity matrix. The flattened 2-D point configuration that best preserves these distances is then obtained via a direct and simple metric MDS method, known as *Classical Scaling*. It provides an analytic solution and is fast to compute. Classical scaling minimizes an objective function known as *Strain*.

Let \star denote the element by element product of two matrices, i.e., if $\mathbf{A} = (a_{ij})$ and $\mathbf{B} = (b_{ij})$, then $\mathbf{A} \star \mathbf{B} = (a_{ij}b_{ij})$. Given a dissimilarity matrix Δ (a symmetric matrix with nonnegative elements and zeroes on the diagonal) that, here, contains the estimated geodesic distances

between the surface points, the *Strain* minimization problem is in this case

$$\min_{\mathbf{X}} \|\mathbf{D}(\mathbf{X}) \star \mathbf{D}(\mathbf{X}) - \Delta \star \Delta\|_F^2$$

where \mathbf{X} is the $n \times 2$ matrix of point coordinates in \mathfrak{R}^2 , $\mathbf{D}(\mathbf{X})$ is a $n \times n$ matrix of Euclidean distances between the points in \mathfrak{R}^2 and $\|\cdot\|_F$ denotes the Frobenius norm, i.e. $\|\mathbf{A}\|_F^2 = \text{trace}(\mathbf{A}'\mathbf{A})$.

The *double centering operator* [4] is defined as:

$$\mathbf{T}(\mathbf{A}) = -\frac{1}{2}\mathbf{J}\mathbf{A}\mathbf{J}.$$

Here \mathbf{A} is a square $n \times n$ matrix and $\mathbf{J} = \mathbf{I} - \frac{1}{n}\bar{\mathbf{e}}\bar{\mathbf{e}}'$, where $\bar{\mathbf{e}} = (1, \dots, 1)'$ is an n -vector of ones and \mathbf{I} is the $n \times n$ identity matrix. The double centering operator is a linear operator on square matrices. Note that $\mathbf{T}(\mathbf{A})$ is symmetric if \mathbf{A} is symmetric.

The classical scaling algorithm consists of the following steps:

- Apply double centering to $\Delta \star \Delta$: $\mathbf{B} = \mathbf{T}(\Delta \star \Delta)$
- Compute the first two eigenvalues (λ_1, λ_2) and eigenvectors $[\bar{\mathbf{v}}_1 | \bar{\mathbf{v}}_2]$ of \mathbf{B} . Create $\Lambda = \text{diag}(\lambda_1, \lambda_2)$ and $\mathbf{Q} = [\bar{\mathbf{v}}_1 | \bar{\mathbf{v}}_2]$.
- The output (flattened) coordinate matrix is given by $\mathbf{X} = \mathbf{Q}\Lambda^{1/2}$.

In this method, *partial* eigendecomposition of an $n \times n$ symmetric matrix is required. Had complete eigendecomposition been necessary, the symmetric QR algorithm could have been used, requiring $O(n^3)$ time. Since only the two largest eigenvalues and their corresponding eigenvectors are required, much faster algorithms can be applied, such as bisection, the power method and Rayleigh quotient iteration and orthogonalization with Ritz acceleration (see [12] for details).

4 Interpolation

Surfaces in 3-D space are often represented by tens of thousands of voxels. Although the interpoint geodesic distances between the surface voxels can be computed quite efficiently, applying the MDS algorithm to such a large amount of data may be inconvenient. We therefore first select a sample of surface voxels (say 1000) and apply the flattening procedure to this subset. Note that the minimal distance between each pair of points in this sample is still calculated using the complete surface model. After flattening the sample, the flattened position of the remaining surface voxels is obtained by interpolation. Many different interpolation algorithms can be employed; radial function interpolation was used in this research.

Formally, given the flattened representation $\{(\xi_i^1, \xi_i^2) \mid i = 1, \dots, m\}$ of a subset of the n surface points $\{(x_i, y_i, z_i) \mid i = 1, \dots, m\}$ we would like to map the remaining surface points from 3-D to 2-D. In particular, we would like to find functions $f_1 : \mathfrak{R}^3 \mapsto \mathfrak{R}$ and $f_2 : \mathfrak{R}^3 \mapsto \mathfrak{R}$ that for $k = 1, 2$ satisfy $f_k(x_i, y_i, z_i) = \xi_i^k$ and interpolate the data well.

Radial functions are a convenient and simple tool for global interpolation of scattered multi-

variate data [9]. Generally, given scattered data of the form $\{(\bar{\mathbf{x}}_i, F_i) : \bar{\mathbf{x}}_i \in \mathfrak{R}^d, F_i \in \mathfrak{R}, i = 1, \dots, n\}$, the radial function interpolation problem is to find an interpolant of the form:

$$S(\bar{\mathbf{x}}) = \sum_{i=1}^m b_i g(\|\bar{\mathbf{x}} - \bar{\mathbf{x}}_i\|^2) + p_s, \quad \bar{\mathbf{x}} \in \mathfrak{R}^d; \quad (3)$$

$$\text{s.t.} \quad p_s \in \Pi_s \quad (4)$$

$$\sum_{i=1}^n b_i q(\bar{\mathbf{x}}_i) = 0, \quad q \in \Pi_s \quad (5)$$

$$S(\bar{\mathbf{x}}_i) = F_i, \quad i = 1, \dots, n \quad (6)$$

where g is a univariate function defined on \mathfrak{R}^+ , Π_s is the space of all algebraic polynomials of degree less than s on \mathfrak{R}^d , and $\|\cdot\|$ is the Euclidean norm on \mathfrak{R}^d .

Condition (5), that the coefficient be orthogonal to the polynomial space, guarantees uniqueness of solution for appropriate $g(\cdot)$ and s . Condition (6) is the requirement that the solution be clamped to the data points $(\bar{\mathbf{x}}_i, F_i)$.

Various classes of functions $g(\cdot)$ have been considered [9, 10]. The function $g(t) = \sqrt{t}$ with $s = 2$ is known to have a unique solution for any set of distinct $\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_m \in \mathfrak{R}^3$. For this choice of g and s the problem reduces to finding radial functions f_k ($k = 1, 2$) of the form

$$f_k(x, y, z) = \sum_{i=1}^m b_i^k \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + a_0^k + a_1^k x + a_2^k y + a_3^k z \quad (7)$$

$$\text{s.t.} \quad \sum_{i=1}^m b_i^k \cdot 1 = 0, \quad \sum_{i=1}^m b_i^k x_i = 0, \quad \sum_{i=1}^m b_i^k y_i = 0, \quad \sum_{i=1}^m b_i^k z_i = 0 \quad (8)$$

$$f_k(x_i, y_i, z_i) = \xi_i^k \quad (9)$$

The coefficients can be found by solving the following linear system (for $k = 1, 2$)

$$\begin{bmatrix} \mathbf{A} & \mathbf{E} \\ \mathbf{E}' & \mathbf{0} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{b}}^k \\ \bar{\mathbf{a}}^k \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\xi}}^k \\ \bar{\mathbf{0}} \end{bmatrix}$$

where

- $\mathbf{A}(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$
- $\mathbf{E}(i, 1) = 1, \mathbf{E}(i, 2) = x_i, \mathbf{E}(i, 3) = y_i, \mathbf{E}(i, 4) = z_i$
- $\bar{\mathbf{b}}^k = (b_1^k, \dots, b_m^k)'$
- $\bar{\mathbf{a}}^k = (a_0^k, \dots, a_3^k)'$
- $\bar{\boldsymbol{\xi}}^k = (\xi_1^k, \dots, \xi_m^k)'$

The solution of this linear system defines a radial function interpolant of the form (7), that satisfies (8) and (9).

The radial function interpolant $f_k(x, y, z)$ (for $k = 1, 2$) is optimal in the sense that it minimizes the roughness measure

$$\int_{\mathbb{R}^3} \left\| \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} + 2 \frac{\partial^2 f}{\partial x \partial y} + 2 \frac{\partial^2 f}{\partial x \partial z} + 2 \frac{\partial^2 f}{\partial y \partial z} \right\|^2 dx dy dz$$

while passing through the observed data points.

The quality of the interpolation depends on the curvature of the surface and on the sampling method. Sampling adapted to the curvature of the surface, or uniform sampling of the surface, will produce the best results but are not always easy to obtain. However, if the

curvature of the surface is not too high, other sampling methods may still lead to good results. For example, when dealing with depth images $z(x, y)$ of smooth surfaces, uniform sampling in the (x, y) plane is usually adequate.

5 Experimental Results

Figure 4 shows a synthetically created surface in 3-D space. The surface can be regarded as a curled rectangular planar sheet of paper, with a uniform chessboard pattern. Note that, due to the curling, this surface is not a function.

Since the surface is a curled plane, it can, ideally, be flattened without any distortion. Applying the algorithm suggested in this paper to the surface yielded the flattened result shown in Fig. 5. The slight distortion is due to the small errors in the estimation of geodesic distances that are used as input to the MDS procedure. Note that as the voxel resolution increases, the distortion, albeit small, will not reach zero. This follows from the local way we measure length (and distance) as an (optimally) weighted sum of chain-code links, and from the fact that the structure of a digital straight line does not change as resolution increases.

Fig. 6 shows Euclidean distances on the flattened surface $\{d(k, l)\}$ as a function of the corresponding estimated geodesic distances $\{\delta(k, l)\}$ on the curled surface in 3-D space. Since the original surface can ideally be flattened without distortion, had error-free geodesic distance estimates been available, $d(k, l)$ would have been equal to $\delta(k, l)$ for all pairs (k, l) , and all points in the graph would have been on the diagonal line. However, due to the slight errors in geodesic distance estimation, distances on the flattened surface cannot be identical to the geodesic distance estimates, leading to the small spread of points near the line. In

quantitative terms, let $\epsilon(k, l)$ denote the absolute value of the discrepancy between $d(k, l)$ and $\delta(k, l)$,

$$\epsilon(k, l) \equiv |d(k, l) - \delta(k, l)|$$

and let $\epsilon_r(k, l)$ denote the absolute value of the relative error,

$$\epsilon_r(k, l) \equiv \left| \frac{d(k, l)}{\delta(k, l)} - 1 \right| .$$

In this example, the average $\bar{\epsilon}$ of the absolute discrepancy $|\epsilon(k, l)|$ over all surface point pairs is 0.423 (voxel size units), less than half a voxel. For comparison, $\bar{\delta}$, the average geodesic distance between points on this surface, is 35.4 . $\bar{\epsilon}_r$, the average absolute value of the relative error, is 2.13%.

Since spatial quantization of a continuous surface is not a space invariant operation, changing the orientation of the curled plane in space, as shown in Fig. 7, generally results in a different voxel-representation of the surface. Thus, the geodesic distance between any two points on the continuous surface, which is obviously independent of orientation, will be estimated slightly differently following digitization and voxel-representation of the surface. This, in turn, leads to a slightly different flattening, as shown in Fig. 8. See also Fig. 9. In this case, the average absolute discrepancy is $\bar{\epsilon} = 0.428$ and the average absolute value of the relative error is $\bar{\epsilon}_r = 2.06\%$.

One application of the suggested surface flattening algorithm is texture mapping. Fig. 10 (left) is a depth image of a human face, in which depth is represented as brightness. Following flattening, Fig. 10 (right) shows Euclidean distances on the flattened surface as a function of the corresponding estimated geodesic distances on the surface of the 3-D model. The larger



Figure 4: A synthetic surface in 3-D space obtained by curling a planar rectangular sheet. Note that this surface is not a function.



Figure 5: The flattened surface obtained using the method presented in this paper.

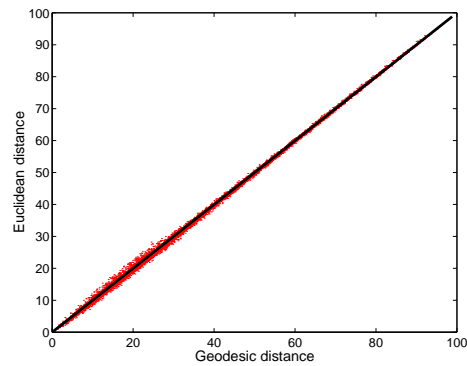


Figure 6: Euclidean distances on the flattened surface vs. the corresponding estimated geodesic distances on the 3-D surface.

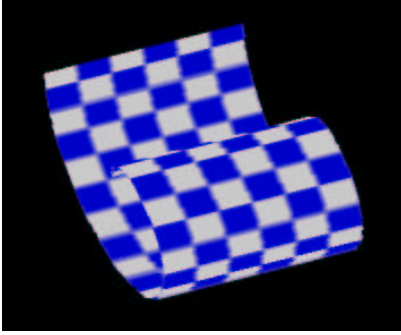


Figure 7: The same surface as in Fig. 4, but at a different spatial orientation. Since the world coordinate system is fixed, the voxel-representation of the surface is totally changed.

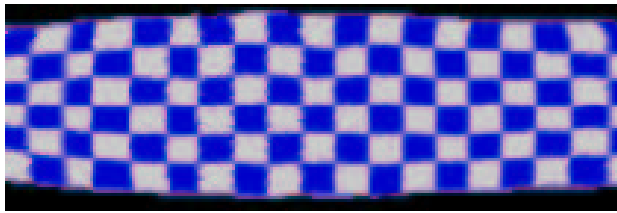


Figure 8: The flattened surface obtained is slightly different than the one shown in Fig. 5.

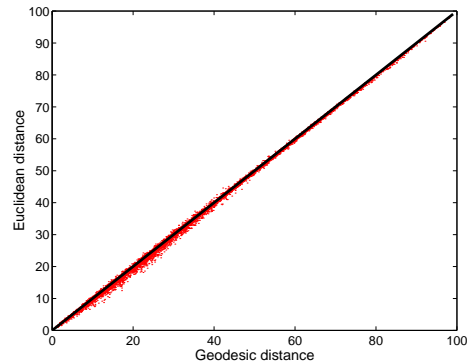


Figure 9: Euclidean distances on the flattened surface vs. the corresponding estimated geodesic distances on the re-oriented 3-D surface. Compare with Fig. 6.

scattering in this case around the diagonal line is due to the fact that this surface cannot be flattened without some distortion. Here, the average absolute discrepancy is $\bar{\epsilon} = 9.04$, and should be compared with the average geodesic interpoint distance $\bar{\delta} = 141.9$. The average absolute value of the relative error is $\bar{\epsilon}_r = 7.66\%$.

Fig. 11 (left) shows a brick wall texture. Overlaying this texture onto the flattened face surface and mapping the surface back to 3-D with its texture (using the known transformation) yields the texture-mapped face shown in Fig. 11 (right). Additional texture mapping examples are shown in Figs. 12-13.

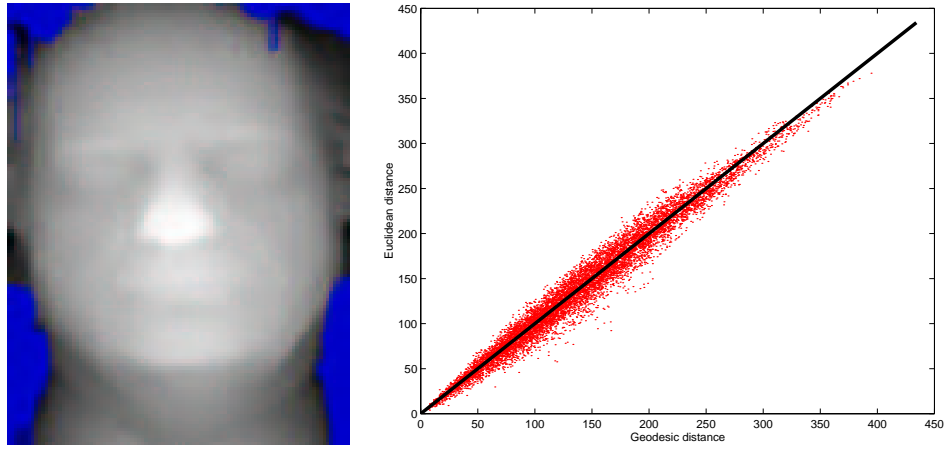


Figure 10: *Left:* A depth image of a human face. *Right:* Euclidean distance on flattened surface vs. estimated geodesic distance on 3D surface of the face.

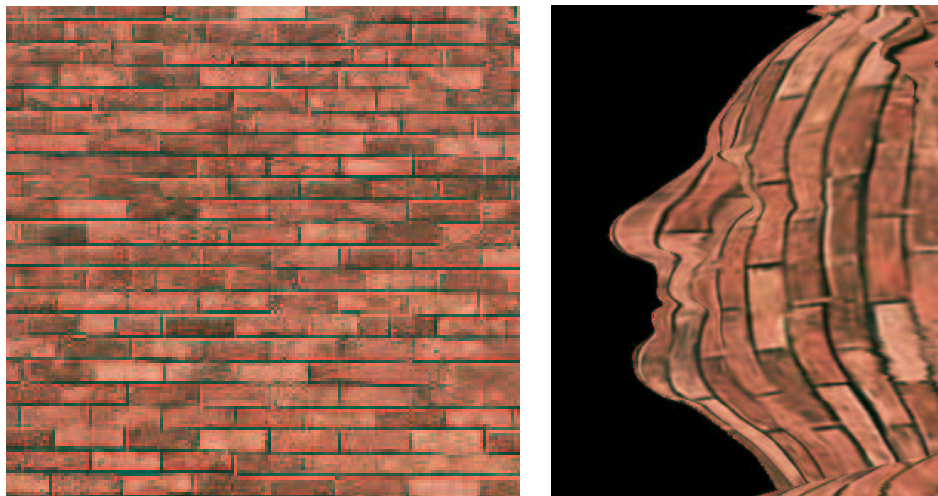


Figure 11: *Left:* A brick-wall texture. *Right:* The brick texture mapped onto the face.

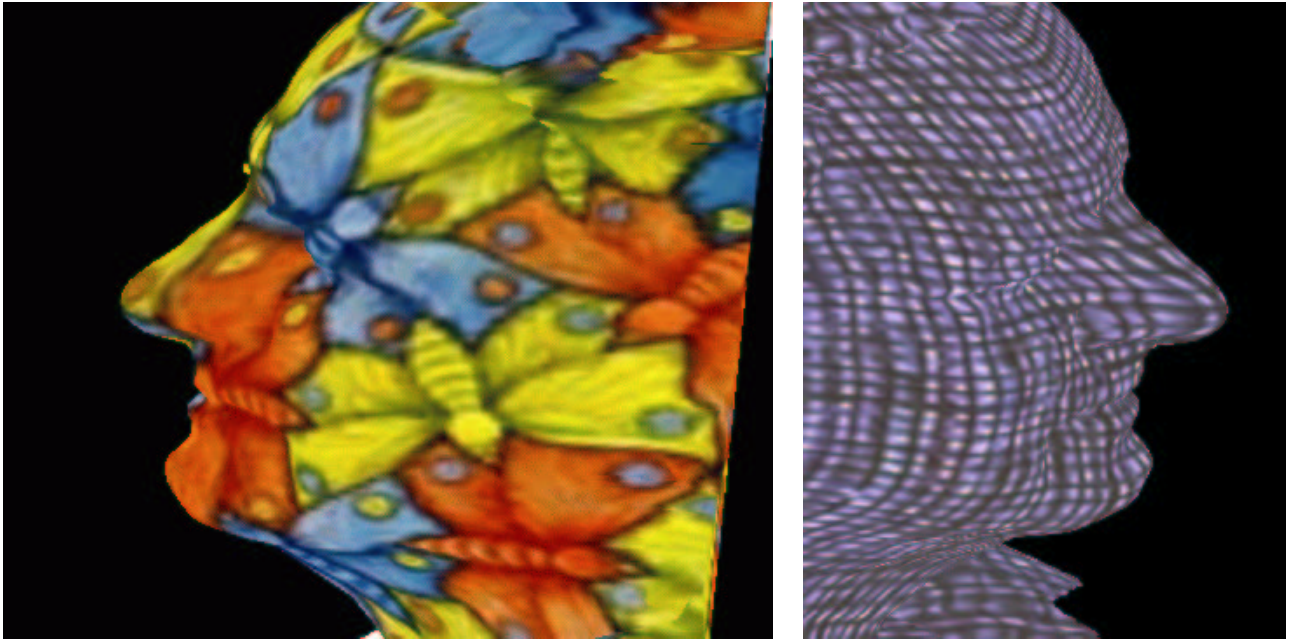


Figure 12: *Left:* An Escher butterfly texture mapped onto the face. *Right:* A cloth texture mapped onto the face.

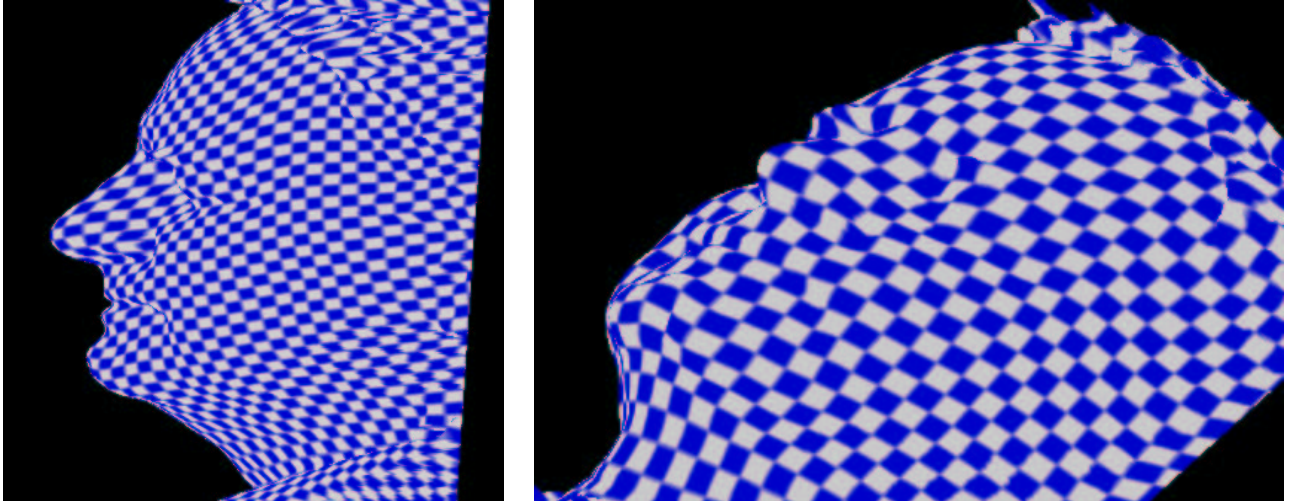


Figure 13: Chessboard texture mapped onto the face.

6 Conclusions

We presented and demonstrated an efficient and practical surface flattening method. Beyond its computational efficiency and ease of implementation, the suggested approach has the following unique characteristics:

- The algorithm operates directly on voxel data. An intermediate triangulated representation of the surface is not necessary. Note that voxel data is the natural output format of common medical imaging systems.
- The solution is optimal in a mathematically well defined sense (the strain metric, applied to global preservation of the estimated geodesic distances). This means that global distances are preserved almost perfectly on developable surfaces, and are approximated as well as possible on non-developable surfaces.
- The approach is essentially analytic; it does not call for iterative search in a solution space, hence there is no risk of convergence to local minima.

We believe that this unique combination of features makes the proposed algorithm attractive for the analysis and visualization of medical imaging data and remote sensing data.

Acknowledgments

We thank Haim Shvaytser, Gabriele Lohmann and Nira Dyn for the interesting discussions. The friendly and helpful advice of Gil Zigelman and Sharon Ganot is appreciated. We are grateful to the National Research Council of Canada for the 3-D digital head model. This research was supported in part by the Adams Super-Center for Brain Studies at Tel Aviv

University, and by the German-Israeli Foundation for Scientific Research and Development (GIF).

References

- [1] S. Angenent, S. Haker, A. Tanenbaum and R. Kikinis, “Conformal Geometry and Brain Flattening”, *Proc. 2nd Intern. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI’99)*, Cambridge, England, pp. 269-278, 1999.
- [2] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer, 1997.
- [3] C. Bennis, J.M. Vezien and G. Iglesias, “Piecewise Surface Flattening for Non-Distorted Texture Mapping”, *Computer Graphics*, Vol. 25, pp. 237-247, 1991.
- [4] F. Critchley, “On Certain Linear Mappings Between Inner-Product and Squared-Distance Matrices”, *Linear Algebra and its Applications*, Vol. 105, pp. 91-107, 1988.
- [5] A.M. Dale, B. Fischl and M.I. Sereno, “Cortical Surface-Based Analysis: 1. Segmentation and Surface Reconstruction”, *NeuroImage*, Vol. 9, pp. 179-194, 1999.
- [6] A.M. Dale, B. Fischl and M.I. Sereno, “Cortical Surface-Based Analysis: 2. Cortical Surface Based Analysis”, *NeuroImage*, Vol. 9, pp. 195-207, 1999.
- [7] M.P Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.
- [8] H.A. Drury, D.C. Van Essen, C.A. Anderson, C.W. Lee, T.A. Coogan and J.W. Lewis, “Computerized Mappings of the Cerebral Cortex: A Multiresolution Flattening Method

- and a Surface-Based Coordinate System”, *J. of Cognitive Neuroscience*, Vol. 8, pp. 1-28, 1996.
- [9] N. Dyn, “Interpolation of Scattered Data by Radial Functions”, in *Topics in Multivariate Approximations* (C.K. Chui, L. L. Schumaker, and F.I. Utreras, eds.), pp. 47-62, Academic Press, 1987.
- [10] N. Dyn, “Interpolation and Approximation by Radial and Related Functions”, *Approximation Theory VI*, Vol. 1, pp. 211-234, 1989.
- [11] J. Gomes and O. Faugeras, “Segmentation of the Inner and Outer Surfaces of the Cortex in Man and Monkey: an Approach Based on Partial Differential Equations”, *Proc. Human Brain Mapping Conf.*, Dusseldorf, Germany, June, 1999.
- [12] G.H Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1989.
- [13] M. Gondran and M. Minoux, *Graphs and Algorithms*, Wiley, Chichester, 1984 (Ch. 2).
- [14] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro and M. Halle, “Conformal Surface Parameterization for Texture Mapping”, *IEEE Trans. on Visualization and Computer Graphics*, Vol. 6, No. 2, 2000.
- [15] G. Hermosillo, O. Faugeras and J. Gomes, “Unfolding the Cerebral Cortex Using Level Set Methods”, *Proc. Scale-Space’99, Lecture Notes in Computer Science*, Vol. 1682, pp. 58-69, 1999.
- [16] A. Jonas and N. Kiryati, “Digital Representation Schemes for 3-D Curves”, *Pattern Recognition*, Vol. 30, pp. 1803-1816, 1997.

- [17] A. Jonas and N. Kiryati, “Length Estimation in 3-D using Cube Quantization”, *J. of Mathematical Imaging and Vision*, Vol. 8, pp. 215-238, 1998.
- [18] N. Kiryati and O. Kübler, “Chain Code Probabilities and Optimal Length Estimators for Digitized Three-Dimensional Curves”, *Pattern Recognition*, Vol. 28, pp. 361-372, 1995.
- [19] R. Kimmel and J.A. Sethian, “Computing Geodesic Paths on Manifolds”, *Proceedings of the National Academy of Sciences*, Vol. 95, pp. 8431-8435, 1998.
- [20] R. Kimmel and J.A. Sethian, “Fast Voronoi Diagrams and Offsets on Triangulated Surfaces”, in *Curve and Surface Design: Saint-Malo 1999*, Vanderbilt University Press, 1999.
- [21] N. Kiryati and G. Székely, “Estimating Shortest Paths and Minimal Distances on Digitized Three-Dimensional Surfaces”, *Pattern Recognition*, Vol. 26, pp. 1623-1637, 1993.
- [22] W.E. Lorensen and H.E. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, *SIGGRAPH’87, ACM Computer Graphics*, Vol. 21, pp. 163-169, 1987.
- [23] S.D. Ma and H. Lin, “Optimal Texture Mapping”, *Eurographics*, pp. 421-428, 1988.
- [24] J.S.B. Mitchell, D.M. Mount and C.H. Papadimitriou, “The Discrete Geodesic Problem”, *SIAM J. Comput.*, Vol. 16, pp. 647-668, 1987.
- [25] B. O’Neill, *Elementary Differential Geometry*, Academic Press, 1966.
- [26] S.T. Roweis and L.K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding”, *Science*, Vol. 290, pp. 2323-2326, 2000.

- [27] E.L. Schwartz, A. Shaw and E. Wolfson, “A Numerical Solution to the Generalized Mapmaker’s Problem: Flattening Nonconvex Polyhedral Surfaces”, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 11, pp. 1005-1008, 1989.
- [28] R. Sedgewick, *Algorithms*, Addison-Wesley, Reading, Massachusetts, 1988 (Chs. 11,31).
- [29] J.P. Snyder, *Flattening the Earth: Two Thousand Years of Map Projections*, The University of Chicago Press, 1993.
- [30] J.B. Tenenbaum, V. de Silva and J.C. Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction”, *Science*, Vol. 290, pp. 2319-2323, 2000.
- [31] E. Wolfson and E.L. Schwartz, “Computing Minimal Distances on Polyhedral Surfaces”, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 11, pp. 1001-1005, 1989.
- [32] G. Zigelman, R. Kimmel and N. Kiryati, “Texture Mapping using Surface Flattening via Multi-Dimensional Scaling”, to appear.