International Journal on Artificial Intelligence Tools
Vol. 22, No. 6 (2013) 1360020 (25 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218213013600208

A NEW ALGORITHM FOR COMPUTING RIEMANNIAN GEODESIC DISTANCE IN RECTANGULAR 2-D AND 3-D GRIDS

OLA NILSSON
Department of Science and Technology, Linköping University, Campus Norrköping
Norrköping, SE-601 74, Sweden
per.ola.nilsson@gmail.com

MARTIN REIMERS
Department of Informatics, University of Oslo, P.O box 1080, Blindern
Oslo, 0316, Norway
martinre@ifi.uio.no

KEN MUSETH
Department of Science and Technology, Linköping University
Campus Norrköping, Norrköping, SE-601 74, Sweden
ken.museth@itn.liu.se

ANDERS BRUN*
Centre for Image Analysis, Uppsala University, Box 337
Uppsala, SE-751 05, Sweden
anders@cb.uu.se

Received 31 January 2013
Accepted 6 July 2013
Published 20 December 2013

We present a novel way to efficiently compute Riemannian geodesic distance over a two- or three-dimensional domain. It is based on a previously presented method for computation of geodesic distances on surface meshes. Our method is adapted for rectangular grids, equipped with a variable anisotropic metric tensor. Processing and visualization of such tensor fields is common in certain applications, for instance structure tensor fields in image analysis and diffusion tensor fields in medical imaging.

The included benchmark study shows that our method provides significantly better results in anisotropic regions in 2-D and 3-D and is faster than current stat-of-the-art solvers in 2-D grids. Additionally, our method is straightforward to code; the test implementation is less than 150 lines of C++ code. The paper is an extension of a previously presented conference paper and includes new sections on 3-D grids in particular.

Keywords: Geodesic distance; the eikonal equation; manifold.

*Corresponding author

1. Introduction

The computation of distances in manifolds is important in both academic and industrial applications, e.g. computational geometry,[2] seismology, optics, computer vision,[3] computer graphics and image analysis.[5,4] Another cross disciplinary example is the versatile level set method[18] that propagates interfaces embedded in scalar distance fields. Often the computations are performed on rectangular grids.

Geodesic distance in a manifold Ω is defined as the metric $d : \Omega \times \Omega \to \mathbb{R}$ that measures the length of the minimal continuous path $\boldsymbol{\gamma}$ connecting two points $\boldsymbol{a}$ and $\boldsymbol{b}$ in Ω, i.e.

$$d(\boldsymbol{a}, \boldsymbol{b}) = \min_{\boldsymbol{\gamma} \subset \Omega} \int_{\boldsymbol{\gamma}} ds \,. \tag{1}$$

Different metrics on Ω can be considered by the choice of distance element ds. In this article we focus on the computation of two- and three-dimensional *distance maps* $\phi(\boldsymbol{x})$ with respect to a source set $S \subset \Omega$, such that

$$\phi(\boldsymbol{x}) \overset{\text{def}}{=} d(\boldsymbol{x}, S) \,, \qquad \boldsymbol{x} \in \Omega \,. \tag{2}$$

These distance functions, or distance transforms, holds distances from all points $\boldsymbol{x}$ in the domain Ω to the source S. We investigate two factors that impact the resulting maps: (a) the type of distance element ds, and (b) the shape of S.

A minimizing path $\boldsymbol{\gamma}$ in (1) is called a *geodesic* and can, because of its special minimizing nature, be used for path planning, interpolation, or more generally as the solution to minimal cost problems. Geodesics can either be explicitly traced or extracted from a distance function.

We begin by exemplifying different distance functions by showing some corresponding geodesics with respect to the topographic map in Figure 1 over $\Omega \subset \mathbb{R}^2$. In $\mathbb{R}^2$, the length of the differential element in Eq. (1) is $ds = \sqrt{d\boldsymbol{\gamma}^T d\boldsymbol{\gamma}} = \sqrt{dx^2 + dy^2}$ and the resulting geodesics are straight lines. The dotted black path is a Euclidean geodesic and hence a straight line between the two end points. To extend this, a positive weight function can be inserted into the integral in Eq. (1): $\int_{\boldsymbol{\gamma}} w \, ds$, so that $ds = w(\boldsymbol{\gamma})\sqrt{d\boldsymbol{\gamma}^T d\boldsymbol{\gamma}}$. In this way, a position dependent weight can model obstacles and non-uniform regions. The dashed red line in Figure 1(a) shows a geodesic, which has a weight function proportional to the magnitude of the gradient of the height function, i.e. a path that avoids steep regions.

Even if a non-uniform weight function is inserted into Eq. (1) the result is locally *isotropic*: the distance element ds depends only on position. *Anisotropic* distance, where the distance element also depends on direction, can be illustrated using the topographic map shown in Figure 1(a). The solid green path is a geodesic with respect to the distance travelled on the terrain surface M defined by the mapping $\boldsymbol{f}(x, y) = (x, y, h(x, y))$ for $(x, y) \in \Omega$. This metric also depends on direction; the differential distance ds depends on the terrain steepness in the direction of travel and is hence anisotropic.

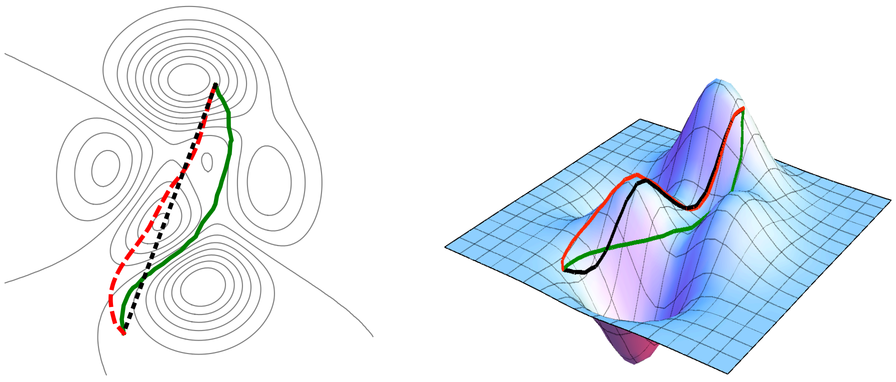


Fig. 1. (Color online) (a) A topographic map corresponding to a height function $h(x, y)$ defined in $\Omega \subset \mathbb{R}^2$, with geodesic paths corresponding to different measures of distance. Dotted black — the Euclidean straight line geodesic for $ds = ||d\boldsymbol{\gamma}||_2$. Dashed red — a weighted geodesic with $ds = \|\nabla h(\boldsymbol{x})\|_2 \, ||d\boldsymbol{\gamma}||_2$. Solid green — using an anisotropic distance element $ds = \sqrt{d\boldsymbol{\gamma}^T \boldsymbol{G}\, d\boldsymbol{\gamma}}$ where distance in Ω equals distance on the surface $(x, y, h(x, y))$ with metric tensor $\boldsymbol{G}$. (b) The corresponding surface and geodesics embedded in three dimensions.

More generally, consider a surface M parameterized over Ω by a bijective mapping $\boldsymbol{f} = (f_1, f_2, \ldots)$ so that $\boldsymbol{f}(\boldsymbol{x}) \in M$ for each $\boldsymbol{x} \in \Omega$. In this case, the distance element is given by

$$ds = \sqrt{d\boldsymbol{\gamma}^T \boldsymbol{G}\, d\boldsymbol{\gamma}} \tag{3}$$

where the *metric tensor* $\boldsymbol{G}$ has elements

$$g_{ij} = \frac{\partial \boldsymbol{f}}{\partial x_i} \cdot \frac{\partial \boldsymbol{f}}{\partial x_j}\,. \tag{4}$$

Such a parametric surface gives raise to a *Riemannian metric* (1) on the parameter domain Ω, and this metric is anisotropic in general. See Figure 2 for an illustration of a parametric surface and a corresponding distance function. In the particular case where the metric is given by an explicit surface parameterization, and in general for surfaces embedded in 3-D, one can consider distance functions on the surface M itself. Such a distance function can be approximated by e.g. triangulating the surface and then running a suitable algorithm on the triangulation.

In this paper,[a] we consider the more general situation when only the metric is provided and no parameterization is known. For this setting, we present a method that efficiently compute geodesic distances to a source set S in a rectangular grid. The technique is based on previous work that has been proposed for triangulated domains in 2-D,[20] which we extend to rectangular grids in 2-D and 3-D equipped

[a]This is an extended version of a previously published conference paper[15] dealing only with the 2-D case. The present article is based in part on unpublished chapters of a PhD thesis.[13]

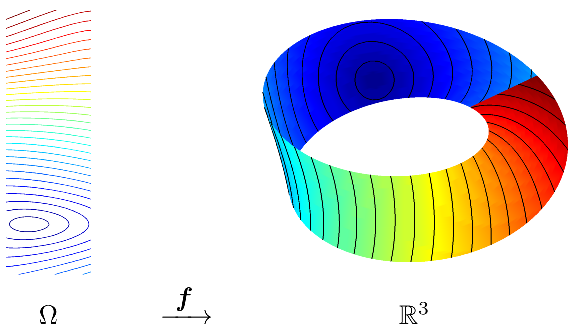


Fig. 2. (Color online) Iso-curves or "wave fronts" of a distance map in the Möbius band. The distances can either be found in the plane, Ω, by considering the metric tensor of the parametrization, or in $\mathbb{R}^3$ if the parameterization $\boldsymbol{f}$ is known. In this case, the distances are computed in Ω; the extent of the 2-D domain and the parameterization determine where the distance map "meets itself" in 3-D.

with a variable metric tensor in each sample point. Examples of this kind include data sets from diffusion tensor MRI,[17] structure tensor data from images,[11] and in general metrics computed from measured data. In Figure 3 we present a toy example, where a distance function is used to study wave propagation in an experiment

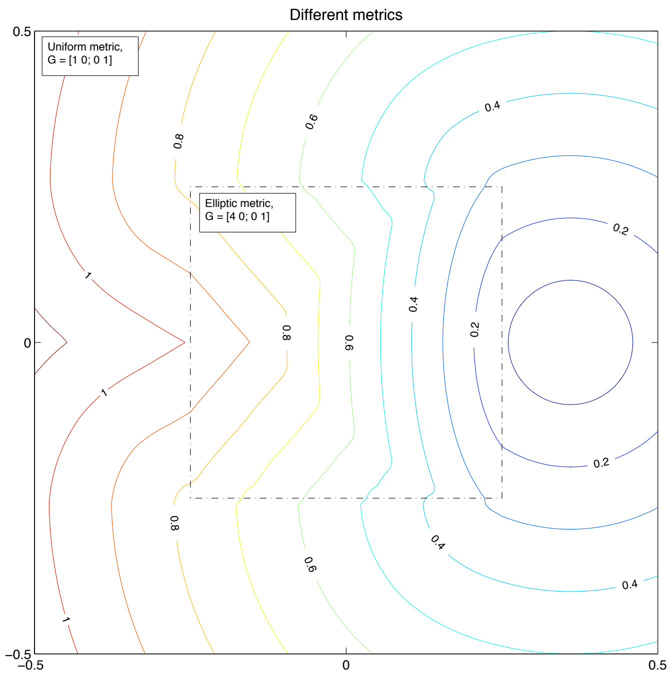


Fig. 3. (Color online) A toy example in a 2-D rectangular grid, including both isotropic (outside dashed rectangle) and anisotropic (inside dashed rectangle) media.

including both isotropic and anisotropic media in a 2-D rectangular grid. Later in this paper, in a series of numerical experiments, we show that our novel method is accurate and fast to compute in comparison with previously reported methods. The method we propose is also relatively simple to understand and implement, consisting of only around 150 lines of C++ code.

In section 2, we review previous work in isotropic and anisotropic distance computations. Section 3 then presents a special discretization based on the integral in Eq. (1) and the dynamic programming principle. In section 4, we explain how to adopt this method for 2-D grids equipped with an anisotropic metric. We first give an intuitive unifying geometric construction that brings together two different approaches by considering the shape of the source S of the distance function. Then we show how the discretization can be used to compute distances in a 2-D grid. Section 5 then extends our method to 3-D grids. Section 6 provides an extensive benchmark study in 2-D, where our new method is compared to other methods. Section 7 then presents a more limited benchmark study for 3-D. The algorithm lends itself well to data sets with sampled metrics and some applications are discussed in section 8. After the conclusions in section 9 we have also included information about the test surfaces used in our experiments in Appendix A and B, to encourage others to reproduce and compare our results.

2. Previous Work

Considerable attention has been devoted to the computation of distances in an anisotropic setting. Early work include shading from shape[30,22] and the salience distance transforms.[21] In optics the problem is typically solved using a shooting type of algorithm, where a method based on solving ODEs and tracking individual rays of light is used.[19] This type of Lagrangian approach is not efficient when it comes to computing a continuous approximation over the rays' embedding domain, which is the case for a distance map.

A distance function as defined in (2) with $ds = ||d\boldsymbol{\gamma}||$ is the *viscosity solution* of the *eikonal equation*

$$\begin{aligned} \|\nabla\phi(\boldsymbol{x})\|_2 &= 1\,, \qquad \boldsymbol{x} \in \Omega, \\ \phi(\boldsymbol{x}) &= 0\,, \qquad \boldsymbol{x} \in S, \end{aligned} \tag{5}$$

see e.g. Ref. 1. The eikonal equation is a nonlinear Hamilton-Jacobi PDE. A weighted version of (5) results by exchanging the right hand side with an inverse positive weight function $1/w$, corresponding to $ds = w(\boldsymbol{\gamma})||d\boldsymbol{\gamma}||$.

Much of the previous work on anisotropic distances is based on the discretization in Ref. 28. It focuses on a Hamilton-Jacobi equation on the more general form

$$H(\nabla\phi, \boldsymbol{x}) = r(\boldsymbol{x})\,, \tag{6}$$

where the model equation

$$H(\phi_x, \phi_y) = \sqrt{a\phi_x^2 + 2b\phi_x\phi_y + c\phi_y^2} = \sqrt{(\nabla\phi)\boldsymbol{B}(\nabla\phi)^T}\,, \qquad \boldsymbol{B} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \tag{7}$$

provides the connection to our work. Approximating distances on a grid using this discretization involves a Godunov-type approximation of the gradient in Eq. (6). The solution (per grid point) is selected from a set containing solutions to 8 quadratic equations in 2-D. For 3-D, the size of this set (and the number of equations to solve) increases to 26. In the original paper, an iterative sweeping update scheme was used to find the global solution in $O(k\,n)$ time where n is the number of grid points in the discretization and k the number of sweeps. This algorithm updates all grid points the same number of times, which is sub-optimal. A different iterative update scheme was proposed in Ref. 7 based on the same discretization. It keeps an unsorted list holding the expanding solution-front and updates nodes on a need-to basis. Even though the run time is not bounded the method is fast in practice, and also amenable to parallelization.

If the distance value at each node only depends on neighboring nodes with smaller values it is possible to construct an update scheme that finds all distances in the domain in one single pass. This causality property is the basis for the well known *fast marching method* (FMM),[23] see also Ref. 29, which approximates isotropic distances on a regular grid in n steps with $O(n \log n)$ time complexity. In Ref. 2, this is extended to parametric three-dimensional manifolds. Two recent one-pass methods are proposed in Refs. 9 and 10; the former uses a reaction-diffusion setting and the latter is based on control theory.

The causality property of the FMM holds only for certain metrics and discretizations, e.g. isotropic distance approximated on regular grids with linear interpolants, and typically not for point source interpolants (see Figure 6) or when the metric is anisotropic. Because of this, a process called *unfolding* was introduced in Ref. 8 to apply the FMM to geodesic distance on triangle meshes. The idea behind unfolding is to insert virtual dependencies between grid points to allow for a one-pass scheme. A related one-pass approach targeting a broader range of Hamilton-Jacobi problems was introduced in Ref. 25. This ordered upwind method is not dependent on the underlying triangulation and therefore avoids unfolding obtuse triangles.

In Ref. 16, a modified version of the FMM scheme was proposed that is more accurate in certain cases. Finally, Ref. 27 finds the exact (isotropic) distance over a triangle mesh in $O(n^2 \log n)$ time.

Our work is based on a method that computes distance maps on triangulated domains proposed in Ref. 20, using a dynamic programming approach and a scheme similar to the one in Ref. 16. We extend this method with a new way to handle more general metrics and 3-D domains combined with line and point sources.

3. Geodesic Distance in Surface Meshes

Finding a distance map on a triangulated domain fulfills the criteria for a dynamic programming problem (DPP) in fixed boundary form.[25] This means that the problem can be broken down into a set of coupled local equations. In this case, the solution to each of these sub-equations is a distance value. This is called an *update* and is

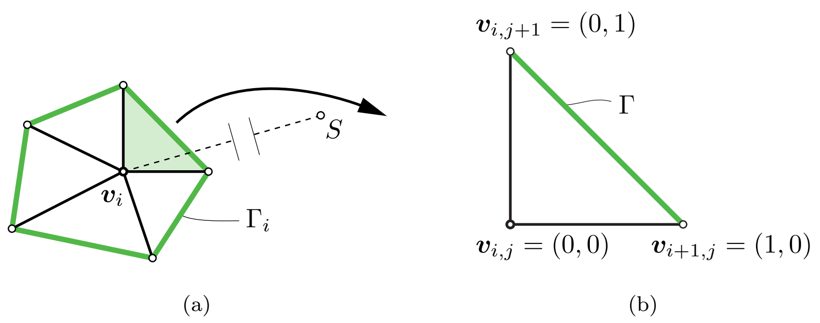

S
$\boldsymbol{v}_i$
Γ_i
(a)
$\boldsymbol{v}_{i,j+1} = (0,1)$
Γ
$\boldsymbol{v}_{i,j} = (0,0)$
$\boldsymbol{v}_{i+1,j} = (1,0)$
(b)

Fig. 4. (Color online) (a) The neighborhood around vertex $\boldsymbol{v}_i$ is split into triangles, which each are considered in turn. (b) The distance $d(\boldsymbol{v}_i, S)$ can be decomposed into the distance to the neighborhood boundary (dashed) plus the distance from the boundary to the source (dotted).

performed per vertex over the mesh. If the distance value at each vertex in the mesh is updated in a strictly decreasing manner, then distances will converge globally.[20]

In Ref. 20 the update was constructed by defining a neighborhood around each vertex $\boldsymbol{v}_i$ (see Figure 4), and requiring that the following nonlinear functional hold

$$\phi(\boldsymbol{v}_i) = \min_{\boldsymbol{v}^* \in \Gamma_i} d(\boldsymbol{v}_i, \boldsymbol{v}^*) + \phi(\boldsymbol{v}^*)\,. \tag{8}$$

This stipulates that ϕ obeys local optimality, i.e. that the solution at $\boldsymbol{v}_i$ is such that the minimizer $\boldsymbol{v}^*$ on the local boundary Γ_i yields the least possible sum of distances. This strategy is valid for any compact manifold.

Inside a triangle the distance $d(\boldsymbol{v}_i, \boldsymbol{v}^*)$ is $\|\boldsymbol{v}_i - \boldsymbol{v}^*\|_2$. We use an *edge interpolant* $f(\boldsymbol{v}) \approx \phi(\boldsymbol{v})$ to approximate the unknown distance from a point on an edge to the source S, interpolating the distances at the edge end-points. As shown in Figure 4, the one-ring can be used as neighborhood. When split into its constituting triangles, Figure 4(b), this gives rise to the following discrete DPP:

$$\begin{aligned} \phi_i &= \min\nolimits_{[\boldsymbol{v}_j, \boldsymbol{v}_k] \subset \Gamma_i} \Phi_{ijk}\,, && \text{for } i = 1, \ldots, n\,. \\ \Phi_{ijk} &= \min\nolimits_{\boldsymbol{v}^* \in [\boldsymbol{v}_j, \boldsymbol{v}_k]} \|\boldsymbol{v}_i - \boldsymbol{v}^*\|_2 + f(\boldsymbol{v}^*) && \\ \phi_i &= 0\,, && \text{for } i \text{ with } \boldsymbol{v}_i \in S\,. \end{aligned} \tag{9}$$

It can be shown that the solution to this system is exact for exact edge interpolants. In practice, one can express the functional in terms of a parameter $t \in [0,1]$, as

$$\Phi_{ijk} = \min_{t \in [0,1]} \|\boldsymbol{v}_i - ((1-t)\boldsymbol{v}_j + t\boldsymbol{v}_k)\|_2 + f((1-t)\boldsymbol{v}_j + t\boldsymbol{v}_k)\,. \tag{10}$$

Early work[6] used a linear edge interpolant $f = f_{linear}$,

$$f_{linear}((1-t)\boldsymbol{v}_j + t\boldsymbol{v}_k) = (1-t)\phi_j + t\phi_k\,, \tag{11}$$

which we will refer to as the *line source interpolant*. This interpolant has Euclidean precision for a linear source S and $\Omega \subset \mathbb{R}^2$, meaning that $f_{linear}(\boldsymbol{v}) = d(\boldsymbol{v}, S)$ in this case. For non-planar meshes, the scheme has first order accuracy.[6] Another linear scheme is used in the popular fast marching method.[23]

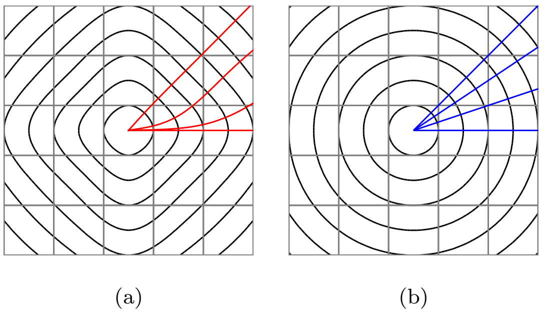


Fig. 5. Iso-lines of distance maps in $\mathbb{R}^2$ from a point source with overlaid geodesics. Computed with: (a) FMM, and (b) the method in Ref. 20. In $\mathbb{R}^2$, the wave fronts from a single point should be perfect circles and the geodesics straight lines.

In some applications it is important to measure the distance to a point and in this case, a linear interpolant can be a poor choice, see Figure 5(a). Because of this, Ref. 20 introduced a nonlinear edge interpolant that is exact for a single point source in the plane, Figure 5(b). This *point source interpolant* reads

$$f_{point}((1-t)\boldsymbol{v}_j + t\boldsymbol{v}_k) = \sqrt{(1-t)\phi_j^2 - t(1-t)\phi_{jk}^2 + t\phi_k^2} \tag{12}$$

where $\phi_{jk} = \|\boldsymbol{v}_j - \boldsymbol{v}_k\|_2$. It gives the same local update as the geometric solution in Ref. 16 but was employed together with a different global update scheme.

3.0.1. *Global update scheme*

Using the linear interpolant, a one-pass algorithm for the global solution of (9) can be found using the fact that values at nodes always depend solely on nodes with smaller values.[23] This is true for linear sources producing linear "wave fronts." However for point sources this causality relationship does not hold as shown in Figure 6.

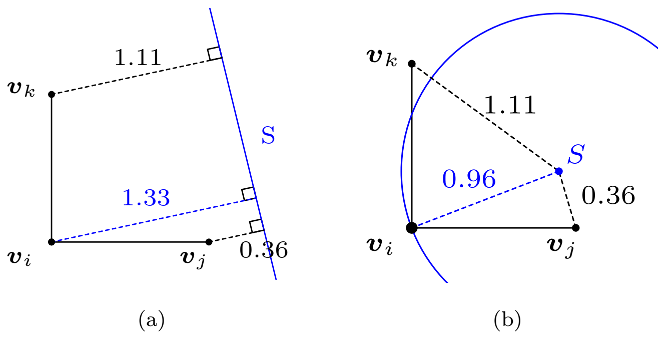


Fig. 6. Different shapes of the source S give different distance maps. With a linear source the "wave fronts" are linear (a) and the value at a node is guaranteed to be larger than nodes that are closer to the source. For a point source, the wave fronts are circular (b) and the FMM causality relationship does not hold.

The method described in Ref. 20 only deals with point sources and cannot depend on such a causality relationship. For this reason, a label-correcting update scheme was introduced that efficiently finds the global solution of (9). This algorithm fits nicely into the anisotropic setting developed in section 4.3 and we describe it briefly here.

Let S be the set of source vertex indices for which distances are known. Then, let C be the candidate set that keeps track of the expanding front of the solution and where $\min C$ retrieves the index of the point in the candidate set with the smallest distance value. The algorithm first initializes C and then proceeds to remove, update, and add points to the candidate set:

Algorithm 3.1: SOLVE(S)

```
$C \leftarrow S$
while $C \neq \emptyset$
do
    $i \leftarrow$ extract($\min C$)
    for $j \in$ neighbors($i$)
    do
        $\phi'_j \leftarrow$ update($j$)
        if $\phi'_j < \phi_j$
        then
            $\phi_j \leftarrow \phi'_j$
            $C \leftarrow$ add($j$)
```

Termination is effective when the candidate set is empty. At this stage, there are no grid points that can be updated to a smaller value and the principle of optimality from dynamic programming states that the solution is optimal. If the candidate set C is a sorted data structure such as a heap, then extract($\min C$) can be performed efficiently in $O(\log n)$ time. From an implementation point of view, this scheme has a complexity similar to the popular FMM.[20] Compared to the algorithm used in Ref. 7 this approach performs fewer redundant updates, at the cost of sorting the candidate set.

4. A Novel Method for 2-D Grids

In this section, we first show how the geometric construction allows previous work to handle configurable sources in 2-D. This is explained in detail, and pseudo-code is given for the update subroutine. The same construction is then used for anisotropic domains in 2-D.

4.1. *A geometric construction*

As noted in Refs. 20 and 16, one can find the minimum in (9) corresponding to a triangle $T_{ijk} = [\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k]$ with the point source interpolant (12) by a planar geometric construction as illustrated in Figure 7. The idea is to map T_{ijk} isometrically to a triangle $T'_{ijk} = [\boldsymbol{v}'_i, \boldsymbol{v}'_j, \boldsymbol{v}'_k]$ in $\mathbb{R}^2$ such that $\boldsymbol{v}'_i = (0,0)$, $\boldsymbol{v}'_j = (d(\boldsymbol{v}_i, \boldsymbol{v}_j), 0)$ and $\boldsymbol{v}'_k$ so that $\|\boldsymbol{v}'_i - \boldsymbol{v}'_k\| = d(\boldsymbol{v}_i, \boldsymbol{v}_k)$ and $\left\|\boldsymbol{v}'_j - \boldsymbol{v}'_k\right\| = d(\boldsymbol{v}_j, \boldsymbol{v}_k)$. Then, if the triangle inequality $d(\boldsymbol{v}_j, \boldsymbol{v}_k) \leq \phi_j + \phi_k$ holds, one can uniquely determine the position of

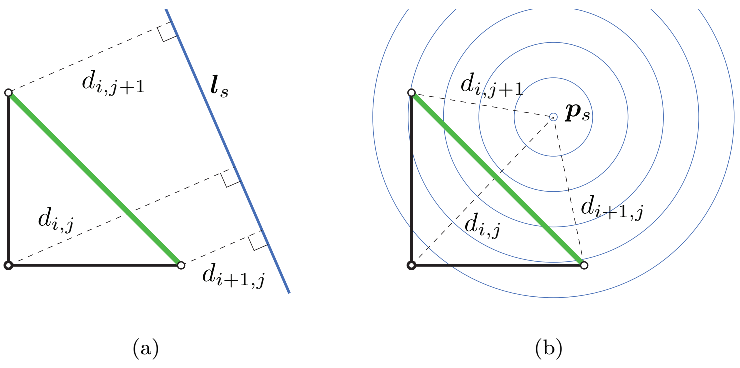


Fig. 7. (Color online) A tentative update value at the ith node in triangle T'_{ijk} is sought. The update is denoted Φ_{ijk}. It is the closest distance from $\boldsymbol{v}'_i$ to the virtual source $\boldsymbol{s}'$ (dashed blue). Using vertex positions and distance values for $\boldsymbol{v}'_j$ and $\boldsymbol{v}'_k$ (i.e. ϕ_j and ϕ_k) we make a geometric construction that places $\boldsymbol{s}'$ in the plane of the triangle; then Φ_{ijk} can be *measured.* (a) The interpretation of a linear source. (b) A point source; located at the circle intersection (red) *outside* Γ.

a virtual point source $\boldsymbol{s}' \in \mathbb{R}^2$ such that $\|\boldsymbol{v}'_j - \boldsymbol{s}'\|_2 = \phi_j$ and $\|\boldsymbol{v}'_k - \boldsymbol{s}'\|_2 = \phi_k$ as in Figure 7(b). Then the minimum of (9) is given by $\Phi_{ijk} = \|\boldsymbol{v}'_i - \boldsymbol{s}'\|_2$ in case the segment $[\boldsymbol{v}'_i, \boldsymbol{s}']$ crosses the edge $[\boldsymbol{v}'_j, \boldsymbol{v}'_k]$, otherwise

$$\Phi_{ijk} = \min\{d(\boldsymbol{v}_i, \boldsymbol{v}_j) + \phi_j,\ d(\boldsymbol{v}_i, \boldsymbol{v}_k) + \phi_k\}\,. \tag{13}$$

The latter solution is also chosen in case the triangle inequality does not hold.

A similar geometric construction is possible for the linear edge interpolant in (11), as follows. A virtual line source S' in $\mathbb{R}^2$ can be determined such that $d(\boldsymbol{v}'_j, S') = \phi_j$ and $d(\boldsymbol{v}'_k, S') = \phi_k$. Then the projection of $\boldsymbol{v}'_i$ on S' yields the closest point $\boldsymbol{s}'$ on S' as illustrated in Figure 7(a). If the geodesic path crosses the edge in question, i.e. the segment $[\boldsymbol{v}'_i, \boldsymbol{s}']$ crosses the edge $[\boldsymbol{v}'_j, \boldsymbol{v}'_k]$ then $\Phi_{ijk} = d(\boldsymbol{v}'_i, \boldsymbol{s}')$, else (22) is chosen. We summarize the geometric update in pseudo code:

Algorithm 4.1: Update$(\boldsymbol{v}_i, \phi, f)$

```
for each $[\boldsymbol{v}_j, \boldsymbol{v}_k] \in \Gamma_i$
  do
    Place $T'_{ijk}$ in $\mathbb{R}^2$ using inter-vertex distances        (1)
    Use $\phi_j, \phi_k$ and $f$ to also place $\boldsymbol{s}'$        (2)
    if $\boldsymbol{s}'$ can be placed and $[\boldsymbol{v}'_i, \boldsymbol{s}']$ crosses $[\boldsymbol{v}'_j, \boldsymbol{v}'_k]$
      then $\Phi_{ijk} \leftarrow \|\boldsymbol{v}'_i - \boldsymbol{s}'\|_2$
      else
        /* Dijkstra type update */
        $\Phi_{ijk} \leftarrow \min\{\|\boldsymbol{v}'_i - \boldsymbol{v}'_j\|_2 + \phi_j,\ \|\boldsymbol{v}'_i - \boldsymbol{v}'_k\|_2 + \phi_k\}$
    $\phi_i \leftarrow \min\{\phi_i,\ \Phi_{ijk}\}$
return $(\phi_i)$
```

This subroutine is the core of the proposed method. It is a loop around the one-ring of vertex $\boldsymbol{v}_i$ where each triangle T_{ijk} is considered as shown in Figure 7. The minimal ϕ_i is then returned as the update value. Together with Algorithm 3.1 this constitutes the main body of the code and is straightforward to implement.

For clarity we also note that with a linear interpolant $\boldsymbol{s}'$ can always be placed (statement 2) and the pseudo code can, in this case, be slightly simplified.

We can also formalize the tentative update value as the solution to a general equation system. Each tentative update value Φ_{ijk} as depicted in Figure 7 is computed as follows. For the linear interpolant the line source S' is determined by the expression $n_x x + n_y y + p = 0$, and from projection we get the following equation system

$$\begin{cases} \hat{n} \cdot \boldsymbol{v}'_i + p = \Phi_{ijk}\,, \\ \hat{n} \cdot \boldsymbol{v}'_j + p = \phi_j\,, \\ \hat{n} \cdot \boldsymbol{v}'_k + p = \phi_k\,, \\ \|\hat{n}\|_2 = 1\,. \end{cases} \tag{14}$$

This allows to solve for the update value Φ_{ijk}.

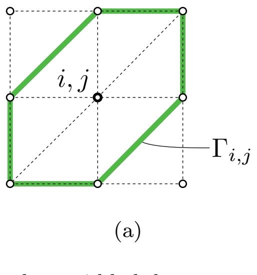
Using instead the point source interpolant gives a nonlinear equation system to solve. Let the position of the point source be $\boldsymbol{s}' = (x, y)$. The equation of the circle then gives us the next system of coupled equations

$$\begin{cases} (\boldsymbol{v}'_j - \boldsymbol{s}')^2 = \phi_j^2 \\ (\boldsymbol{v}'_k - \boldsymbol{s}')^2 = \phi_k^2, \end{cases} \tag{15}$$

which gives the means to determine $\boldsymbol{s}'$. Then, the update is given by $\Phi_{ijk} = \|\boldsymbol{v}'_i - \boldsymbol{s}'\|_2$.

4.2. *Gridded data*

Distance computations are important in many applications for which data is naturally given on a rectilinear grid; e.g. image processing. Our method adopts to a grid in the following straightforward manner. Let the neighborhood of each grid point be triangulated as shown in Figure 8(a) creating a mesh of vertices (grid points) connected by edges. This merely dictates the connectivity of the grid points. The triangulation can be done implicitly in the solver without a penalty overhead or explicitly as a pre-processing step. The solution is then found for the vertices in



(a)

Fig. 8. When applying the method to gridded data, a tessellation is needed. Here is a regular triangulation of a rectilinear grid around point i, j.

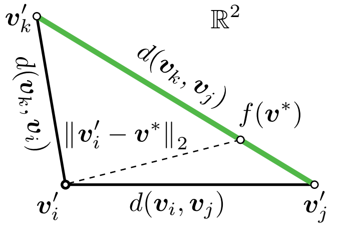


Fig. 9. A linear approximation of the isometric embedding using triangle edge lengths. Now the DPP in Eq. (9) can be used both with line and point interpolants (see Figure 7).

the mesh, i.e. the grid points. No post processing or interpolation is needed in the conversion.

Note that when the mesh is constructed from an underlying rectilinear uniform grid the expression for the update (using the linear or point interpolant) can be simplified.

4.3. *Weighted and anisotropic updates*

Geodesic distance in a manifold is locally measured using the differential element ds from Eq. (3). Combining this with Eq. (1) we get the following continuous form

$$d(\boldsymbol{a}, \boldsymbol{b}) = \min_{\gamma \subset \Omega} \int_\gamma \sqrt{\dot{\boldsymbol{\gamma}}(t)^T \boldsymbol{G}(\boldsymbol{\gamma}(t))\, \dot{\boldsymbol{\gamma}}(t)}\, dt, \tag{16}$$

where γ is any C^1 curve connecting $\boldsymbol{a}$ and $\boldsymbol{b}$. We use the DPP Eq. (8) as before, but in the general case we have to also approximate the length of a geodesic path inside each triangle in Γ_i with respect to the chosen non-Euclidean metric. We do this by a local planar embedding similar to the one in the previous section, and then approximate local distance in Euclidean space. Our approach is based on the following properties: (1) The Euclidean distance computation in section 4 is convergent under grid refinement,[20] and (2) A triangle is uniquely determined by its edge lengths (up to isometry).

To approximate geodesic distance with respect to a general metric inside a triangle T_{ijk} in Ω, we use a geometric planar construction similar to the one in the previous section: map T_{ijk} to a triangle $T'_{ijk} = [\boldsymbol{v}'_i, \boldsymbol{v}'_j, \boldsymbol{v}'_k]$ in $\mathbb{R}^2$ so that each edge $[\boldsymbol{v}'_p, \boldsymbol{v}'_q]$ of T'_{ijk} has length equal to an approximation $\tilde{d}(\boldsymbol{v}_p, \boldsymbol{v}_q)$ of the true geodesic distance $d(\boldsymbol{v}_p, \boldsymbol{v}_q)$ as shown in Figure 9.

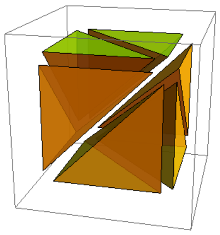
We use

$$d(\boldsymbol{a}, \boldsymbol{b}) \approx \tilde{d}(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{(\boldsymbol{b} - \boldsymbol{a})^T \boldsymbol{G}(\boldsymbol{a}, \boldsymbol{b})(\boldsymbol{b} - \boldsymbol{a})}\,, \tag{17}$$

where

$$\boldsymbol{G}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{G}(\boldsymbol{a}) + \boldsymbol{G}(\boldsymbol{b})}{2}\,, \tag{18}$$

and $\boldsymbol{G}(\boldsymbol{x})$ is metric tensor evaluated at $\boldsymbol{x}$. As in the previous section we place $\boldsymbol{v}'_i$ in the origin, $\boldsymbol{v}'_j$ on the positive x-axis and $\boldsymbol{v}'_k$ above the x-axis. The embedded triangle



(a)

Fig. 10. (Color online) In 3-D, a grid can be split into tileable tetrahedra. The grid points are located at the cube corners.

is, in theory, realizable by construction since it is positioned using the geodesic edge lengths, which are guaranteed to fulfill the triangle inequality; this is required for any metric. In practice, due to discretization and round-off errors, this requirement can fail. In these cases, we use the "Dijkstra" update in Eq. (22) instead. In the code, this corresponds to one extra check at Statement (1) in Algorithm 4.1.

After a simplex has been embedded in Euclidean space, the method proceeds exactly as previously described for isotropic distances. That is, an update value is found from the DPP in Eq. (9), using the geometric construction in $\mathbb{R}^2$ with a virtual source $\boldsymbol{s}'$; see Figures 7 and 9.

The distance measure in (17) could also be directly inserted into Eq. (9) to find the local update. With the metric taken from Eq. (18), however, the minimization would involve finding roots of a fourth or fifth order polynomial depending on the interpolant; this is feasible but impractical. Another alternative is to take $\boldsymbol{G}$ constant and equal to the average of the metric at the vertices of T_{ijk}, i.e. $\boldsymbol{G} = (\boldsymbol{G}(\boldsymbol{v}_i) + \boldsymbol{G}(\boldsymbol{v}_j) + \boldsymbol{G}(\boldsymbol{v}_k)/3$, as proposed in Ref. 20 for scalar weighted metrics. The geometric construction is still applicable for this approximation. In our 2-D experiments, we have also tested this approximation and named it "Reimers."

5. A Novel Method for 3-D Grids

As suggested above, it is possible to extend our method to 3-D. In 3-D we start by assuming a tetrahedral mesh, which can be implicitly constructed from an underlying grid if needed. One such tessellation strategy is shown in Figure 10(a).

Following the notation in Ref. 13, we denote the local *boundary* for vertex i with Γ_i. It is shaded green in Figure 11. The boundary in 3-D is the set of all triangles T_{ijk} in the generalized one-ring of $\boldsymbol{v}_i$. A particular update, given inside tetrahedra T_{ijkl}, has the face T_{jkl} as its corresponding part of Γ_i. In 3-D, the DPP from Ref. 13 reads

$$\phi_i = \min_{T_{jkl} \subset \Gamma_i} \Phi_{ijkl}\,, \qquad \text{for } i = 1, \dots, n\,. \tag{19a}$$

$$\Phi_{ijkl} = \min_{\boldsymbol{v}^* \in T_{jkl}} \|\boldsymbol{v}_i - \boldsymbol{v}^*\|_2 + f(\boldsymbol{v}^*, \phi) \tag{19b}$$

$$\phi_i = 0\,, \qquad \text{for } i \text{ with } \boldsymbol{v}_i \in S\,. \tag{19c}$$

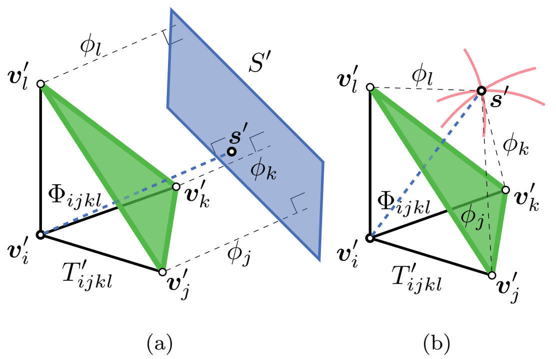


Fig. 11. (Color online) The minimizer is the closest distance from $\boldsymbol{v}_i$ to the source (dashed blue). (a) Using linear interpolation, i.e. approximating a planar source. (b) Using a point source interpolant and the sphere intersection solution. In 3-D the boundary Γ of the tetrahedron T'_{ijkl} is the triangle T'_{jkl} (shaded green). A valid update is required to cross Γ.

We begin by expressing the interpolants proposed in Refs. 20 and 6 in three dimensions. First, write a point on a triangular face of Γ_i in barycentric coordinates, as $\boldsymbol{v}^* = u\boldsymbol{v}_j + v\boldsymbol{v}_k + w\boldsymbol{v}_l$ with $u, v, w \geq 0$ and $u + v + w = 1$. The three-dimensional counterpart of the linear interpolant from Ref. 6 then becomes

$$f_{linear}(u, v, w, \phi) = u\phi_j + v\phi_k + w\phi_l\,, \tag{20}$$

where ϕ_i, ϕ_j, and ϕ_k are the current distance values at node i, j, and k. Similarly, a 3-D equivalent of the point source interpolant from Ref. 20 can be derived to be

$$\begin{aligned} f_{point}(u, v, w, \phi) &= \|(u\boldsymbol{v}_j + v\boldsymbol{v}_k + w\boldsymbol{v}_l) - \boldsymbol{s}\|_2 \\ &\;\;\vdots \\ &= \sqrt{\phi_j^2 u + \phi_k^2 v + \phi_l^2 w - \phi_{jk}^2 uv - \phi_{kl}^2 vw - \phi_{jl}^2 uw}\,, \\ &\quad \text{for } u, v, w \in [0, 1], \text{ and } u + v + w = 1\,. \end{aligned} \tag{21}$$

This expression also makes use of the known inter vertex distances, $\phi_{ij} = \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2$.

The DPP in Eq. (19) finds the position $\boldsymbol{v}^*$ that gives the shortest path from $\boldsymbol{v}_i$ to $\boldsymbol{s}$ by minimization. The length of this geodesic, $d(\boldsymbol{v}_i, \boldsymbol{s})$, is the update value. For this to be a valid update, we require the geodesic to be contained in the set formed by the two touching tetrahedra T_{ijkl} and T_{sjkl}. In Euclidean space, this is equivalent to ensuring that the geodesic crosses the face T_{jkl} — the associated subset of the boundary Γ_i.

5.1. *The geometric construction in 3-D*

Analogous to the two-dimensional case, the update value, Eq. (19b), can also be found in a geometric setting. This is done by projecting the vertex in question on the source set S giving a position $\boldsymbol{s}$ and then measuring $d(\boldsymbol{v}_i, \boldsymbol{s})$. We do this in an isometric space. Let the tetrahedra T_{ijkl} be embedded as $T'_{ijkl} \in \mathbb{R}^3$. This can be achieved by first placing $\boldsymbol{v}'_i$ in $(0, 0)$ and letting $\boldsymbol{v}'_j = (d(\boldsymbol{v}_i, \boldsymbol{v}_j), 0)$. Then, place $\boldsymbol{v}'_k$ such that $\|\boldsymbol{v}'_i - \boldsymbol{v}'_k\|_2 = d(\boldsymbol{v}_i, \boldsymbol{v}_k)$ and $\left\|\boldsymbol{v}'_j - \boldsymbol{v}'_k\right\|_2 = d(\boldsymbol{v}_j, \boldsymbol{v}_k)$. If T'_{ijk} is a valid triangle, then place $\boldsymbol{v}'_l$ such that $\|\boldsymbol{v}'_i - \boldsymbol{v}'_l\|_2 = d(\boldsymbol{v}_i, \boldsymbol{v}_k)$, $\left\|\boldsymbol{v}'_j - \boldsymbol{v}'_l\right\|_2 = d(\boldsymbol{v}_j, \boldsymbol{v}_l)$, and $\|\boldsymbol{v}'_k - \boldsymbol{v}'_l\|_2 = d(\boldsymbol{v}_k, \boldsymbol{v}_l)$. If the construction of any of the constituting triangles fail[b] or T'_{ijkl} does not form a valid tetrahedron, a Dijkstra update,

$$\Phi_{ijkl} = \min \{d(\boldsymbol{v}_i, \boldsymbol{v}_j) + \phi_j,\ d(\boldsymbol{v}_i, \boldsymbol{v}_k) + \phi_k,\ d(\boldsymbol{v}_i, \boldsymbol{v}_l) + \phi_l\}, \tag{22}$$

is used. Otherwise we can proceed and place $\boldsymbol{s}'$.

We consider two types of boundary conditions, or shapes of S. When using the linear interpolant (Eq. (20)), the corresponding source is a plane S'. The projection of $\boldsymbol{v}'_i$ on S' is shown in Figure 11(a). This gives the position $\boldsymbol{s}'$ that subsequently is used to measure $d(\boldsymbol{v}_i, \boldsymbol{s}')$. As previously mentioned, it is required that $\boldsymbol{v}^*$ lies inside the triangular face T'_{jkl}.

For the point source interpolant (Eq. (21)), the boundary is a single point $S' = \boldsymbol{s}'$. Its position is given by a three-sphere intersection, as illustrated in Figure 11(b).

6. Experimental Results in 2-D

We choose to benchmark our method with metrics that have a corresponding known analytic distance function. For future reference and benchmarks by other researchers, a detailed description of the test manifolds has been included in Appendix A.

For the point source interpolant, distances can be found in closed form for all the tested metrics. Validation for the line source interpolant, it is less straightforward. In Euclidean space the distance to polygons are readily given in closed form, but in anisotropic domains we resort to numerical minimization to find sufficiently accurate distances. The line source interpolant furthermore assumes an infinite line as a source but the boundary condition effectively only model *line segments*. This results in regions in the domain where the closest point on the boundary is the line segment itself, and regions that are closest to one of its end points. With a line source interpolant the distance to the former can be exactly computed, but the latter can only by approximated.

We compare our method with the state-of-the-art anisotropic eikonal Hamilton-Jacobi solver of Jeong and Whitaker,[7] which to our knowledge is the fastest solver using the discretization introduced in Ref. 28. We also compare our work with the newly introduced fast marching solver of Lenglet *et al.*,[10] which is publicly available.[12] The GPU version of the algorithm by Jeong and Whitaker has been reported to be significantly faster, but requires specialized hardware. To make a fair comparison, we have not included this or any other parallell or GPU based code in our benchmark.

[b]The edge lengths must pass the triangle inequality.

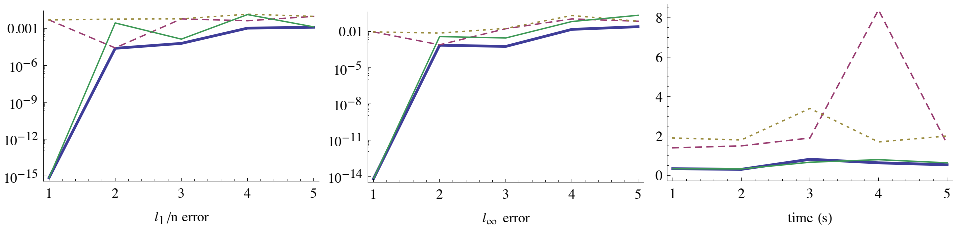


Fig. 12. (Color online) Error and run time of our method (thick blue), Jeong and Whitaker (dashed red), Lenglet *et al.*(dotted brown) and Reimers (solid green) for five different test cases: 1 — plane S_p, 2 — plane S_l, 3 — sphere S_p, 4 — sphere S_l, 5 — Poincare S_p.

The test grid is 400×400 grid points with spacing $h = 0.005$. For the unit disk domains, i.e. the sphere and Poincaré metrics, we have restrained computations outside a radius of 0.98 to avoid degenerate tensors at the border. For the point source interpolant, we place the sources exactly on grid points. To produce meaningful results, line sources are not placed exactly on grid points. Due to limitations in the code (Ref. 12) this restricts the accuracy of the method of Lenglet *et al.* for some of the test cases. We report the mean of the absolute error, that is l_1/n, the max error l_∞, and the time for the computations. The findings are listed in Figure 12 and the corresponding error distribution is shown in Figure 14. To show how the method behaves under refinement we have also performed a numerical convergence study; this is reported in Figure 13.

As expected, our method and Reimers' provide Euclidean precision over domains with unit metric using the point source interpolant. This is directly evident from Figure 14. Our method additionally gives the same kind of accuracy for the line source interpolant over the plane, but only for limited regions as explained above, see also Figure 14. Similar accuracy, for this test case, is inherently given by the method by Jeong and Whitaker.

The numbers and figures show that our method consistently performs better than competing work. Compared to the methods that are based on the Hamilton-Jacobi discretizations in Ref. 28, our results are orders of magnitudes better and yet has

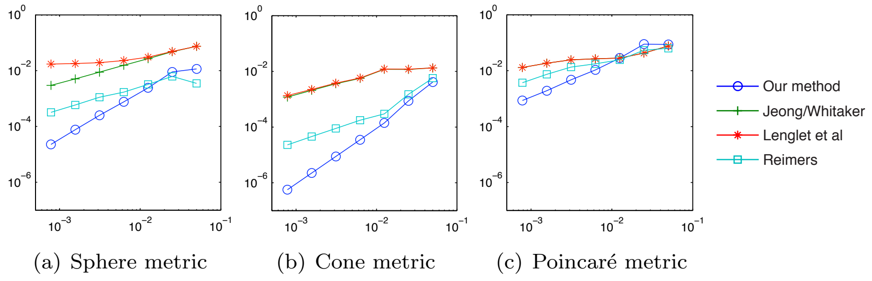


Fig. 13. Convergence under regular refinement using different metrics and the point source interpolant. Plots show l_1/n as a function of step size h.

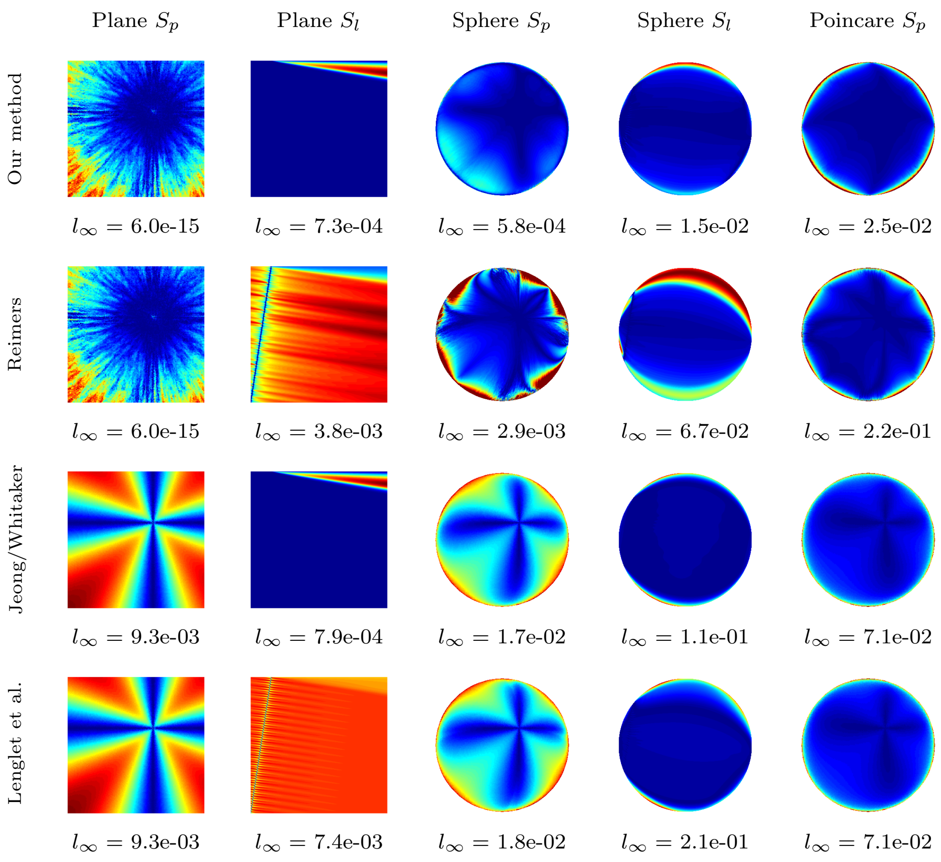


Fig. 14. (Color online) Error distribution plots. Color shows absolute error (blue to red) and is linearly scaled to fall in $[0, l_\infty]$. For more statistics, see Table 12 and Figure 13. (Limitations in the Lenglet code[12] prohibits accurate placement of linear sources.)

lower run times. When compared to the more related method in Ref. 20 our results also show favourable behaviour, and as seen in Figure 14 our error distributions are smoother and seem to be more predictable, especially close to the source.

The numerical study indicates convergent behaviour of our method. The rate of convergence is on par or slightly better than the competing methods as seen in Figure 13. Interestingly, the convergence rate seems to be higher for the cone metric, which is a "developable" manifold.

7. Experimental Results in 3-D

To verify the behavior of the 3-D version of the algorithm we have performed a limited convergence study using a proof of concept implementation. We partition the positive quadrant of the unit sphere into a regular grid and equip it with a metric tensor field, $\boldsymbol{G}_i$, at each grid point. Distances are then computed with our method to a single

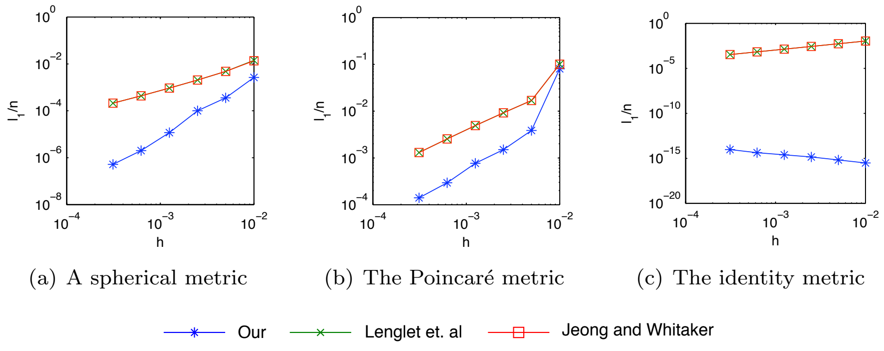


Fig. 15. We compute distances over part of the unit sphere and plot the mean of absolute error, l_1/n, against the grid step size h for different metrics.

point using the point source interpolant. We compare the results with a state of the art implementation[7] of the classical Hamilton-Jacobi discretization as originally proposed in Ref. 28. We also compare to the fast marching type of method in Ref. 10.

The results, shown in Figure 15 together with the results from Figure 13, indicate that our method behaves similarly in 2-D and 3-D. For Euclidean space, our method is exact to numerical precision. Because of this the results are dominated by round-off errors and the l_1/n-norm actually grows under grid refinement, as demonstrated in 15(c). In anisotropic spaces, our method consistently has a smaller error than competing work and seems to perform predictably. The proof of concept implementation is furthermore relying on a complete tetrahedral tessellation. The extra burden of this explicit connectivity information in 3-D makes the method less efficient, especially since the test is run on a regular grid. For the examples shown here, our method is slower than competing work. We believe this can be helped by an implicit tessellation strategy that would significantly lower the memory footprint of the method and thus streamline processing.

8. Applications

In this section we briefly outline some applications that need accurate geodesic distances in anisotropic metric settings.

8.1. *Finding geodesics*

A common goal of distance computations is to extract geodesics from the distance map, as is done in Figure 1 to find the optimal path between two points. The typical way of doing so includes computing the gradient of the distance field and tracing the path of steepest descent. In this and other scenarios, the smoothness of the distance function is important. In Figure 16, we compare the smoothness of our distance maps with competing work.

We also recapitulate a way of finding geodesics without depending on the differentiability of the approximated distance map as first proposed in Ref. 20, *cf.* Figure 1.

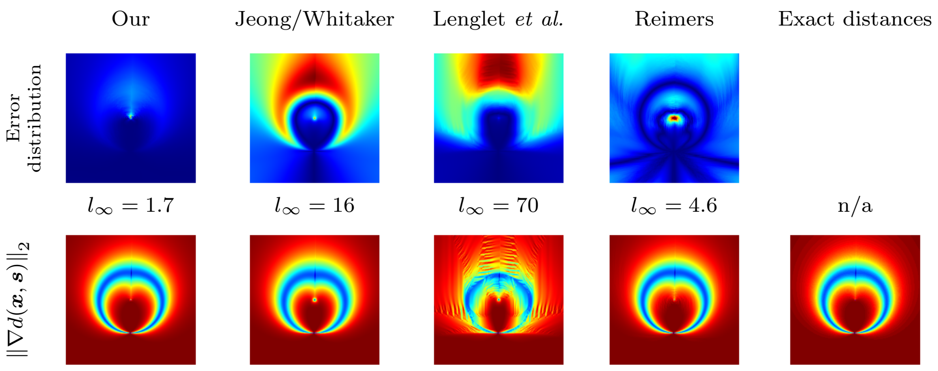


Fig. 16. (Color online) Visualizing distance map smoothness under a cone metric with slope $s = 5$. Top row shows color coding of the absolute error (from blue to red) of the distance map. Smoother transition in color indicate smoother maps. Bottom row shows derivative smoothness by color coding the norm of the gradient of the respective map.

The DPP in Eq. (8) can be applied "backwards" and gives a natural way of computing the geodesic paths. Starting from a point $\boldsymbol{p}$ the geodesic can be found by applying the following Algorithm.

Algorithm 8.1: TraceGeodesic($\boldsymbol{p}, \phi, f$)

```
$i \leftarrow 0, \boldsymbol{\gamma}_i \leftarrow \boldsymbol{p}$
while $\boldsymbol{\gamma}_i \neq S$
  do { $\boldsymbol{\gamma}_{i+1} = \text{FindNextMinimizer}(\boldsymbol{\gamma}_i, \phi, f)$
       $i \leftarrow i + 1$
return $(\gamma)$
```

As illustrated in Figure 17, the subroutine FindNextMinimizer finds the next node $\boldsymbol{\gamma}_{i+1}$ on the geodesic path as seen from the current node $\boldsymbol{\gamma}_i$. This is achieved by finding the argument that minimizes the following expression

$$\boldsymbol{\gamma}_{i+1} = \arg\min_{\boldsymbol{x} \in \Gamma_i} \|\boldsymbol{x} - \boldsymbol{\gamma}_i\| + f(\boldsymbol{x}). \tag{23}$$

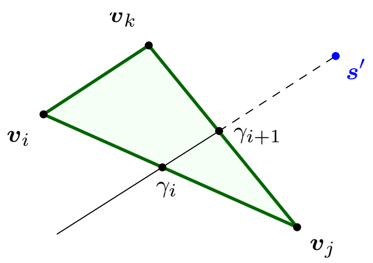


Fig. 17. The geodesic path γ is the path that minimizes the distance integral in Eq. (1). We approximate the geodesic by successively finding points $\boldsymbol{\gamma}_i$ that minimize the travel path locally. This is done by placing $\boldsymbol{s}'$ and locating $\boldsymbol{\gamma}_{i+1}$ as the intersection of the segments $[\boldsymbol{\gamma}_i, \boldsymbol{s}']$ and $[\boldsymbol{v}_j, \boldsymbol{v}_k]$.

Here the interpolant f and the norm $\|\cdot\|$ can be chosen and combined as described earlier in section 4. To emphasize the similarity with Algorithm 4.1, we also outline the pseudo-code for Eq. (23).

Algorithm 8.2: FindNextMinimizer($\boldsymbol{\gamma}_i, \phi, f$)

```
for each $[\boldsymbol{v}_j, \boldsymbol{v}_k] \in \Gamma_i$
  do
    Place $\boldsymbol{\gamma}_i, \boldsymbol{v}_j, \boldsymbol{v}_k$ in a local coordinate system in $\mathbb{R}^2$
    Use $\phi_j, \phi_k, f$ to also place $\boldsymbol{s}'$
    if $\boldsymbol{s}'$ can be placed and $[\boldsymbol{\gamma}_i', \boldsymbol{s}']$ crosses $[\boldsymbol{v}_j', \boldsymbol{v}_k']$
      then $\Phi_{ijk} \leftarrow \|\boldsymbol{v}_i - \boldsymbol{s}'\|_2$
      else
        /* Dijkstra type update */
        $\Phi_{ijk} \leftarrow \min\{\|\boldsymbol{v}_i' - \boldsymbol{v}_j'\|_2 + \phi_j,\ \|\boldsymbol{v}_i' - \boldsymbol{v}_k'\|_2 + \phi_k\}$
        $\boldsymbol{s}' \leftarrow \arg\min_{\boldsymbol{v} \in \{\boldsymbol{v}_j, \boldsymbol{v}_k\}} \phi(\boldsymbol{v})$
    if $\Phi_{ijk} < \phi_i$
      then
        $\phi_i \leftarrow \Phi_{ijk}$
        $\boldsymbol{\gamma}_{i+1} \leftarrow$ intersection$([\boldsymbol{\gamma}_i, \boldsymbol{s}'], [\boldsymbol{v}_j, \boldsymbol{v}_k])$
return $(\boldsymbol{\gamma}_{i+1})$
```

If the current node on the geodesic coincides with a vertex then Γ_i is taken to be the one-ring of $\boldsymbol{\gamma}_i$, else Γ_i equals the boundary of the current triangle. For the Dijkstra type update, we can't place the virtual source as normal. Instead we approximate the direction of the solution and place $\boldsymbol{\gamma}_{i+1}$ in the node $\{\boldsymbol{v}_j, \boldsymbol{v}_k\}$ with the smallest distance value.

8.1.1. *Anisotropic partitioning and sampling*

An interesting application using anisotropic distances is Voronoi partitioning and anisotropic sampling. A Voronoi cell of a set of indexed samples amounts to

$$\mathrm{Vor}(\boldsymbol{x}_i) = \{\boldsymbol{x} \mid d(\boldsymbol{x}_i, \boldsymbol{x}) \le d(\boldsymbol{x}_j, \boldsymbol{x}), \quad \text{for } i \neq j\}\,. \tag{24}$$

Previous work on anisotropic partitioning has used a definition of geodesic distance that can give rise to disconnected Voronoi cells, called *orphans* in Refs. 4 and 5, see Figure 18(a). The problem stems from the approximations made where distance is measured pointwise without considering the intermediate path. This is not consistent with the principle of local optimality and the contradiction gives rise to physically unrealizable solutions. Using instead Eq. (8) together with Algorithm 3.1 gives distances that are guaranteed to be monotonically increasing. This yields a partitioning where each cell is connected set by construction, see Figure 18(b). The result of partitioning an image using this technique can be seen in Figure 19. Applications for this sampling include packing of tensor glyphs, visualization of diffusion tensor MRI, and adaptive sampling of images.

8.1.2. *Distance to objects*

In many image analysis applications, it is important to compute distance to an object as opposed to just a point. In this setting a linear wave front assumption is

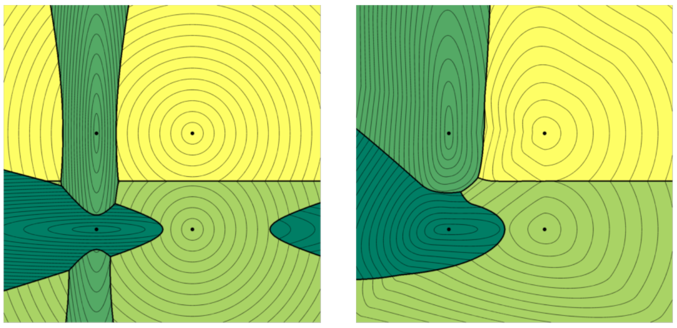


Fig. 18. A Voronoi partitioning. (a) Disconnected Voronoi cells, called *orphans*, can appear when using non-physical definition of geodesic distance. (b) Using a physically realizable interpretation of geodesic distance orphans are guaranteed not to appear.

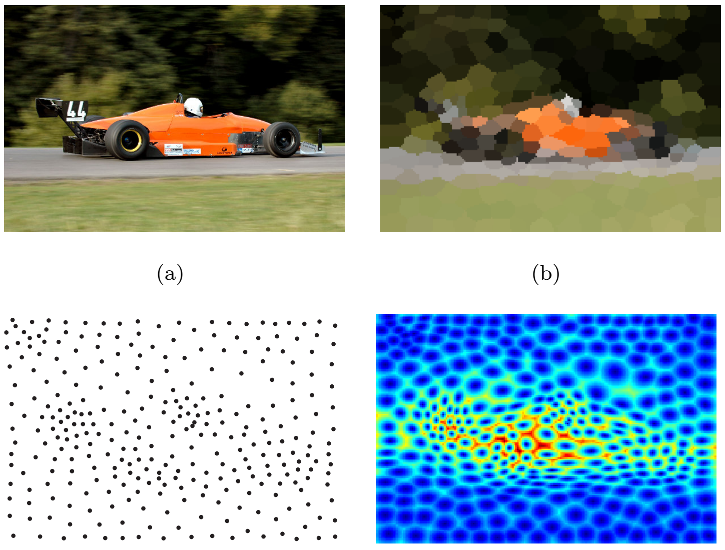
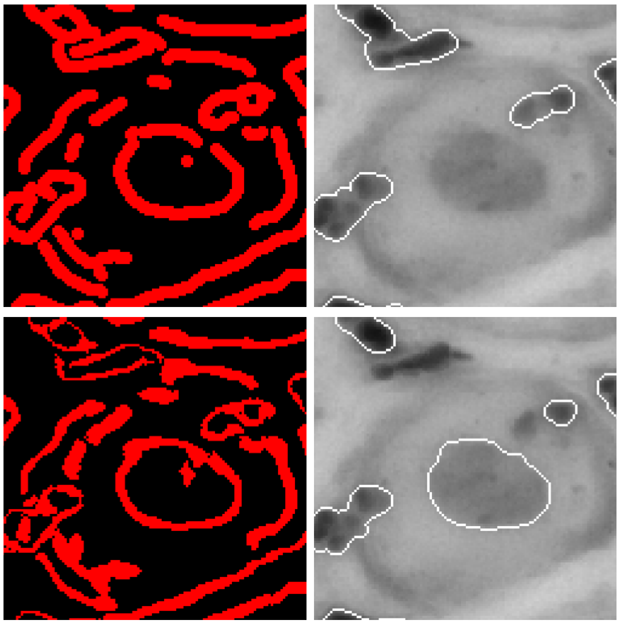


Fig. 19. (Color online) (a) A picture of a fast moving car. A low-passed structure tensor field defines a metric for the image, which then was partitioned into 300 geodesic Voronoi cells using manifold Loyd relaxation.[5] (b) Each cell is colored with its respective mean color. Anisotropic cells align with lines and features in the image. Bottom row shows sample placement (c) and the color coded distance map (d).

in many aspects a better candidate than the point source, see for instance Figure 14. Figure 20(a) shows a more realistic example, that could benefit from our method, anisotropic curve closing for cell nuclei segmentation.[11] The applications of so-called distance transforms of this kind, for binary objects, are frequent in image analysis.[26,24]



(a)

Fig. 20. Anisotropic curve closing used to perform cell nuclei segmentation. Top row shows result using isotropic distances, bottom row anisotropic distances. Image courtesy of Patrik Malm.

This means that our method enables new anisotropic versions of dilation, erosion, skeletonization, computation of form factors, watershed segmentation and much more.

9. Conclusion

We have presented a novel metric dependent solver that converges to geodesic distances under grid-refinement in 2-D and 3-D. It is relatively simple to implement, less than 150 lines of C++ code. The proposed method gives results that converge to geodesic distances under grid-refinement and performs in a predictable manner, as seen in Figure 13 and Table 12 for 2-D and Figure 15 for 3-D. The accuracy is orders of magnitude better than competing methods and has a better error distribution; this is particularly evident in Figures 16 and 14. It appears to have better convergence properties than two current state-of-the-art algorithms.

Our main motivation for publishing these results is to present a solver that we have found to be easy to implement and adapt. It is a solver that is different from many current approaches, in particular FMM.

Acknowledgments

The main ideas of this paper were first explored in the technical report “Distance maps in an arbitrary metric tensor field, An iterative solver for 2-D charts”.[14] We are grateful for comments on the manuscript by Cris Luengo, Gunilla Borgefors and Michael Bang Nielsen.

References

1. Martino Bardi and Italo Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations* (Birkhauser, 1997).
2. Alexander M. Bronstein, Michael M. Bronstein and Ron Kimmel, Weighted distance maps computation on parametric three-dimensional manifolds. *Journal of Computational Physics* **225**(1) (2007) 771–784.
3. A. R. Bruss, The eikonal equation: Some results applicable to computer vision, in B. K. P. Horn and M. J. Brooks (eds.), *Shape from Shading* (MIT Press, Cambridge, MA, 1989), pp. 69–87.
4. Qiang Du and Desheng Wang, Anisotropic centroidal voronoi tessellations and their applications. *SIAM J. Sci. Comput.* **26**(3) (2005) 737–761.
5. Louis Feng, Ingrid Hotz, Bernd Hamann, and Kenneth Joy, Anisotropic noise samples, *IEEE Transactions on Visualization and Computer Graphics* **14**(2) (2008) 342–354.
6. Roberto Gonzalez and Edmundo Rofman, On deterministic control problems: An approximation procedure for the optimal cost I. The stationary problem, *SIAM Journal on Control and Optimization* **23**(2) (1985) 242–266.
7. Won-Ki Jeong, P. Thomas Fletcher, Ron Tao and Ross T. Whitaker, Interactive visualization of volumetric white matter connectivity in DT-MRI using a parallel-hardware Hamilton-Jacobi solver, in *IEEE Transactions on Visualization and Computer Graphics* (*Proc. of IEEE Visualization 2007*) (2007), pp. 1480–1487.
8. Ron Kimmel and James A. Sethian, Computing geodesic paths on manifolds, 1998.
9. Ender Konukoglu, Maxime Sermesant, Olivier Clatz, Jean-Marc Peyrat, Hervé Delingette and Nicholas Ayache, A recursive anisotropic fast marching approach to reaction diffusion equation: Application to tumor growth modeling, in *IPMI* (2007), pp. 687–699.
10. Christophe Lenglet, Emmanuel Prados, Jean-Philippe Pons, Rachid Deriche and Olivier Faugeras, Brain connectivity mapping using Riemannian geometry, control theory and PDEs, *SIAM Journal on Imaging Sciences* (2008).
11. Patrik Malm and Anders Brun, Closing curves with Riemannian dilation: Application to segmentation in automated cervical cancer screening, in *Proc. of 5th Int. Symp. on Visual Computing* (Las Vegas, Nevada, USA, 2009).
12. Gabriel Peyré, Toolbox Fast Marching, http://www.mathworks.com/matlabcentral/fileexchange/6110, 22 August 2008.
13. Ola Nilsson, *Level-set methods and geodesic distance functions*, PhD thesis, Linköping University Linköping University, Department of Science and Technology, The Institute of Technology, 2009.
14. Ola Nilsson and Anders Brun, Distance maps in an arbitrary metric tensor field: An iterative solver for 2-d charts, Technical report, ITN, Linköping University, March 2008. Presented at the Swedish Symposium on Image Analysis.
15. Ola Nilsson, Martin Reimers, Ken Museth and Anders Brun, A novel algorithm for computing Riemannian geodesic distance in rectangular 2D grids, in George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Charless Fowlkes, Sen Wang, Min-Hyung Choi, Stephan Mantler, Jürgen P. Schulze, Daniel Acevedo, Klaus Mueller and Michael E. Papka (eds.) *ISVC* (2), Vol. 7432 of *Lecture Notes in Computer Science* (Springer, 2012), pp. 265–274.

16. Marcin Novotni and Reinhard Klein, Computing geodesic distances on triangular meshes, in *Proc. of the 10th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision* (*WSCG'2002*), February 2002.
17. Lauren O'Donnell, Steven Haker and Carl-Fredrik Westin, New approaches to estimation of white matter connectivity in diffusion tensor MRI: Elliptic PDEs and geodesics in a tensor-warped space, *Fifth Int. Conf. on Medical Image Computing and Computer-Assisted Intervention* (*MICCAI'02*), 2002.
18. Stanley Osher and James A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, *Journal of Computational Physics* **79** (1988) 12–49.
19. Claudio G. Parazzoli, Benjamin E. C. Koltenbah, Robert B. Greegor, Tai A. Lam and Minas H. Tanielian, Eikonal equation for a general anisotropic or chiral medium: Application to a negative-graded index-of-refraction lens with an anisotropic material, *J. Opt. Soc. Am. B* **23**(3) (2006) 439–450.
20. Martin Reimers, *Topics in Mesh Based Modelling*, PhD thesis, Univ. of Oslo, September 2004.
21. Paul L. Rosin and Geoff A. W. West, Salience distance transforms, *Graph. Models Image Process.* **57**(6) (1995) 483–521.
22. Elisabeth Rouy and Agnes Tourin, A viscosity solutions approach to shape-from-shading, *SIAM Journal on Numerical Analysis* **29**(3) (June 1992) 867–884.
23. James A. Sethian, A fast marching level set method for monotonically advancing fronts, *Proc. Nat. Acad. Sci.* **93**(4) (1996) 1591–1595.
24. James A. Sethian, *Level Set Methods and Fast Marching Methods* (Cambridge Univ. Press, June 1999).
25. James A. Sethian and Alex Vladimirsky, Ordered upwind methods for static Hamilton-Jacobi equations: Theory & applications, *SIAM J. Numerical Analysis* **41**(1) (2003) 325–363.
26. Milan Sonka, Vaclav Hlavac and Roger Boyle, *Image Processing: Analysis and Machine Vision* (Thomson-Engineering, September 1998).
27. Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler and Hugues Hoppe, Fast exact and approximate geodesics on meshes, in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (ACM Press, New York, NY, USA, 2005), pp. 553–560.
28. Yen-Hsi Richard Tsai, Li-Tien Cheng, Stanley Osher and Hong-Kai Zhao, Fast sweeping algorithms for a class of hamilton-jacobi equations, *SIAM Journal on numerical analysis* **41**(2) (2004) 673–694.
29. John N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, *IEEE Transactions on Automatic Control* **40**(9) (1995) 1528–1538.
30. Piet W. Verbeek and Ben J. H. Verwer, Shading from shape, the eikonal equation solved by grey-weighted distance transform, *Pattern Recogn. Lett.* **11**(10) (1990) 681–690.

Appendix A. Test Metrics and Analytic Distances in 2-D

A spherical shell: $\{x, y : x^2 + y^2 \le 0.98^2\}$, $S_p = \{0.25, 0.25\}$,
$S_l = \{[(-1,-1), (-.7, 1)]\}$

$$g_{ij}(x,y) = \frac{1}{1-x^2-y^2}\begin{bmatrix} 1-y^2 & xy \\ yx & 1-x^2 \end{bmatrix}, \quad \text{and}$$

$$d(\mathbf{a},\mathbf{b}) = \cos^{-1}(\mathbf{a}^T\mathbf{b} + \sqrt{(1-\mathbf{a}^T\mathbf{a})(1-\mathbf{b}^T\mathbf{b})})\,.$$

A flat square: $(x, y) \in [-1, 1] \times [-1, 1]$, $S_p = \{0.25, 0.25\}$, $S_l = \{[(-1, -1), (-.7, 1)]\}$

$$g_{ij} = \delta_{ij}\,, \qquad d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2\ .$$

The Poincaré disk model: $\{x, y : x^2 + y^2 \leq 0.98^2\}$, $S_p = \{0.25, 0.25\}$

$$g_{ij}(x, y) = \frac{\delta_{ij}}{(1 - x^2 - y^2)^2}\,, \qquad d(\mathbf{a}, \mathbf{b}) = \frac{1}{2}\cosh^{-1}\left(1 + \frac{2\,\|\mathbf{a} - \mathbf{b}\|_2^2}{(1 - \|\mathbf{a}\|_2^2)(1 - \|\mathbf{b}\|_2^2)}\right).$$

Cone metric: A cone $z = s\sqrt{x^2 + y^2}$ with slope s, $\{(x, y) \neq (0, 0)\}$

$$g_{ij}(x, y) = \delta_{ij} + \frac{s^2}{x^2 + y^2}\begin{bmatrix} x^2 & xy \\ yx & y^2 \end{bmatrix}$$

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{\left[\|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2 - 2\|\mathbf{a}\|_2\|\mathbf{b}\|_2 \cos\left(\frac{\angle(\mathbf{a}, \mathbf{b})}{\sqrt{(1 + s^2)}}\right)\right](1 + s^2)}$$

Appendix B. Test Metrics and Analytic Distances in 3-D

The Poincaré Metric: $\{x, y, z : x^2 + y^2 + z^2 \leq 0.98^2\}$

$$\boldsymbol{G} = \boldsymbol{I}/(1 - x^2 - y^2 - z^2)^2\,,$$

with distance function

$$d(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{2}\cosh^{-1}\left(1 + \frac{2\,\|\boldsymbol{a} - \boldsymbol{b}\|_2^2}{(1 - \|\boldsymbol{a}\|_2^2)(1 - \|\boldsymbol{b}\|_2^2)}\right).$$

The 3-sphere metric: $\{x, y, z : x^2 + y^2 + z^2 \leq 0.98^2\}$

$$\boldsymbol{G} = \boldsymbol{J}^T\boldsymbol{J}$$

$$\boldsymbol{J} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \tau x & \tau y & \tau z \end{bmatrix}$$

$$\tau = -1/\sqrt{1 - x^2 - y^2 - z^2}\,,$$

with distance function

$$d(\boldsymbol{a}, \boldsymbol{b}) = \cos^{-1}\left(\boldsymbol{a}^T\boldsymbol{b} + \sqrt{(1 - \boldsymbol{a}^T\boldsymbol{a})(1 - \boldsymbol{b}^T\boldsymbol{b})}\right).$$