Eurographics Symposium on Geometry Processing 2013
Yaron Lipman and Richard Hao Zhang
(Guest Editors)

Volume 32 (2013), Number 5

Practical Anisotropic Geodesy

Marcel Campen Martin Heistermann Leif Kobbelt

RWTH Aachen University, Germany

Figure 1: *Visualization of a geodesic distance field with respect to an* anisotropic metric *(shown by tensor ellipses on the left; field source on top). Next to a reference solution, we show the results and approximation errors of various adapted known algorithms applied for the distance computation. Our novel* Short-Term Vector Dijkstra *on the far right shows the best results.*

Abstract

The computation of intrinsic, geodesic distances and geodesic paths on surfaces is a fundamental low-level building block in countless Computer Graphics and Geometry Processing applications. This demand led to the development of numerous algorithms – some for the exact, others for the approximative computation, some focussing on speed, others providing strict guarantees. Most of these methods are designed for computing distances according to the standard Riemannian metric induced by the surface's embedding in Euclidean space. Generalization to other, especially anisotropic*, metrics – which more recently gained interest in several application areas – is not rarely hampered by fundamental problems. We explore and discuss possibilities for the generalization and extension of well-known methods to the anisotropic case, evaluate their relative performance in terms of accuracy and speed, and propose a novel algorithm, the* Short-Term Vector Dijkstra*. This algorithm is strikingly simple to implement and proves to provide practical accuracy at a higher speed than generalized previous methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

1. Introduction

Measuring lengths or distances is one of the most fundamental operations in the analysis, processing, and synthesis of geometry. While this task can be as simple as applying the Pythagorean formula in free Euclidean space, it quickly gets more complex as we move to a non-Euclidean, intrinsic setting on manifolds, and even more if these measurements are to be taken with respect to a non-standard metric.

We consider a 2-manifold M (possibly with boundary) equipped with smoothly varying norms $\|\cdot\|_{g_x}$ on the tangent spaces T_xM. As these norms allow us to infinitesimally measure distances on M, for a continuously differentiable curve $\zeta : [0,1] \to M$ we can define its total length through integration as $\ell(\zeta) = \int_0^1 \|\zeta'(t)\|_{g_{\zeta(t)}} \, \mathrm{d}t$. With this we can define the *intrinsic metric* g measuring geodesic distances between two points $p, q \in M$ as the infimum over the lengths of all curves ζ on M connecting p with q, i.e.

$$g(p,q) = \inf_{\zeta} \{\ell(\zeta) : \zeta(0) = p, \zeta(1) = q\},$$

and in this way obtain a so-called *length metric space* (M,g).

Note that in the special but common case that $\|\cdot\|_{g_x}$ is induced by some inner product $\langle\cdot,\cdot\rangle_x$ on T_xM for each $x \in M$, i.e. $\|v\|_{g_x} = \sqrt{\langle v,v\rangle_x}$, g is a Riemannian metric. If M is further embedded in Euclidean space – as is the case in most geometry processing scenarios – the Euclidean dot product implies a natural inner product via its restriction to T_xM. We call the corresponding norm *standard norm*, denoted $\|\cdot\|_x$, and the induced Riemannian metric *standard metric*.

Relative to the standard metric and norm we characterize a metric g and its underlying norm based on their quotient $r(v,x) = \|v\|_{g_x}/\|v\|_x$ (for $v \neq 0$) as follows:

- If $r(v,x) \equiv r(x)$, i.e. it is independent of v, we call the metric g *isotropic* as there is no directional dependency.
- If $r(v,x) \equiv r(x) \not\equiv 1$, we call the metric g *weighted* by the weight field $r(x)$.
- If $r(v,x) \not\equiv r(x)$, i.e. there is some directional dependency, we call the metric g *anisotropic*.

The value $\gamma(x) = \max_v r(v,x)/\min_v r(v,x)$ defines the local degree of anisotropy or simply *local anisotropy* and $\gamma = \max_x \gamma(x)$ is the *maximum anisotropy*.

While the standard norm realizes the intuitive notion of geodesic distance and is most common in applications, more general anisotropic norms that incorporate directional dependencies gained increasing interest in recent years. Applications range from Meshing [BPC08, CBK12] and Segmentation [BPC08, SJC09], over Path Planning [RR90, LMS99] and Shape Matching [SJC09], to (mainly in 3D) Medical Imaging [PWKB02, PWT05, BCLC09, BC11]. The anisotropy can be related to principal curvature directions, terrain steepness, surface vector fields, or MRI diffusion tensors, to name some examples (cf. Section 3).

Such applications typically operate in a discrete setting, e.g. on triangle meshes approximating manifolds. Numerous methods for approximate computation of (mainly isotropic) distances in such settings have been proposed. Their accuracy and efficiency typically depends on the quality of the triangulation: intuitively, the "rounder" the individual triangles, i.e. the closer to equilateral, the better. While working with rather nice triangulations is quite common in the digital geometry processing field (leveraged by powerful remeshing techniques), a problem emerges when anisotropic distances are dealt with: the notion of "roundness" is metric dependent! This means a perfectly round triangle which is equilateral in the standard metric can be a "cap" or "needle" far from round when viewed under another, anisotropic metric.

Unfortunately, when naïvely adapting traditional methods to non-standard metrics, they expect element roundness with respect to these metrics, while the meshes typically used in applications are optimized with respect to the standard metric – as often favored by the other processing steps. Hence, in practice anisotropic distance computations quickly arrive at inacceptably low accuracy with increasing degree of anisotropy, as illustrated in Figure 1 for $\gamma = 20$.

1.1. Contribution

In this paper we show how (and how well) known distance computation methods proposed for the isotropic standard scenario can deal with general metrics, analyze the inherent issues, and discuss the results that can be achieved. For metrics with a high degree of anisotropy it turns out: typically either the runtime gets high or the accuracy low. This is problematic for practical applications in Computer Graphics and Geometry Processing as they rely on distance computations as a fundamental operation that is used many times.

Improving upon this, we propose *Short-Term Vector Dijkstra*: an algorithm that can intuitively be understood as Dijkstra's classical shortest path algorithm equipped with a vector-valued short-term memory. It is fast and easy to implement (hardly more complex than Dijkstra's method itself) while providing a practical level of accuracy even for metrics with a high degree of anisotropy.

2. Related Work

Exact Distances Sophisticated methods using window propagation or sequence trees [MMP87, CH90, SSK*05, XW09] allow for the computation of exact geodesic distances on triangulated surfaces. Note that this exactness is with respect to the piecewise-linear surface specified by the mesh M. In geometry processing scenarios where M itself is an approximation of a (piecewise) smooth manifold, the expense of employing such exact algorithms can be futile depending on the application. This holds even more when we turn our focus to non-standard metrics which are also specified only approximately, e.g. discretely per mesh element.

Graph Approximation Dijkstra's classical shortest path algorithm computes shortest paths and distances in graphs. By choosing an appropriate graph and suitable edge weights we can use the corresponding weighted graph distance to approximate distances on M. In the simplest case this graph is the 1-skeleton, i.e. the edge graph, of the mesh M, where the edges are weighted by their length. Higher accuracy, and actually an arbitrary balance between speed and accuracy, can be achieved by constructing a graph with additional Steiner vertices on M's edges and edges across M's faces [Lan99, LMS97, KS00]. The addition of edges between non-adjacent but nearby vertices [CBK12] allows for faster computations and (at comparable graph size) higher accuracy, but on the downside no arbitrary balancing is possible.

Consistent Approximation In contrast to these Dijkstra-based approaches, so-called *Fast Marching* methods compute and propagate distances not only along edges of a graph, but also "continuously" across the faces of a triangle mesh [KS98, SV00, Tsi95]. By appropriate choices of the per-face propagation rules the approximation can be made consistent – in the sense that the results could be driven towards the exact solution by refining the

mesh. For this, M needs to be an acute triangulation, no obtuse inner angles are allowed. As this is hardly ever the case for unstructured meshes in practice, techniques that add additional virtual edges/triangles to be considered during the computation have been presented as a remedy [KS98, SV04, YSS*12]. Alternative non-linear propagation rules have been proposed [NK02, TWZZ07], which can typically increase accuracy in practice – although at the expense of losing consistency [WDB*08].

Non-Propagative A very different approach has been presented by Crane et al. [CWW13]. An approximation to the intrinsic distance field of a source is computed by means of solving two global linear systems instead of explicitly propagating distances from the source over the surface. An interesting property is leveraged by the information about sources appearing only on the right hand side of the system: after a pre-factorization, distance fields for different sources can be computed very efficiently (basically in linear time).

3. Anisotropic Metrics

We consider a discretized setting where, on a triangle mesh M, an (anisotropic) norm $\|\cdot\|_g$ is specified in a sampled manner. The samples can be given per vertex, edge, or face of M, denoted as $\|\cdot\|_{g_v}$, $\|\cdot\|_{g_e}$, or $\|\cdot\|_{g_f}$, respectively – where necessary, we can approximate one form from another via averaging/interpolation.

Looking at the application scenarios of anisotropic metrics which appeared in the literature so far, most often Riemannian metrics are dealt with. In such case, the corresponding norms can conveniently be expressed through a tensor field G as $\|v\|_{g_x} = \sqrt{v^T G_x v}$. Popular examples include:

- **Curvature tensor:** using the shape operator, tensors whose eigenvectors are aligned with directions of minimal and maximal curvature of M and whose eigenvalues are related to the magnitude of minimal and maximal curvature can be constructed. Figure 1 exemplarily visualizes such curvature-related tensors using ellipses.
- **Vector field tensor:** using the vectors of a tangent vector field as first eigenvector and given two (global) coefficients to be used as eigenvalues, tensors "aligned" with the field can be constructed, e.g. to guide geodesic curves accordingly [CBK12].
- **Diffusion tensor:** the characteristics of the water diffusion process in biological tissue can be estimated from magnetic resonance imaging (MRI) acquisitions and be expressed as a tensor field. Note that this is typically applied in 3D volumes. While we focus on 2-manifolds here, we show in Section 6.2 that our novel method is as well applicable to volumetric meshes or grids.

While such tensor based metrics are widely used, it bears noting that also more general metrics, based on non-elliptic norms, are of interest. Examples are terrain steepness profiles [LMS99], curvature (variation) minimizing metrics [YSS*12], or high angular resolution diffusion imaging (HARDI) metrics [PWT05]. We will hence keep the exposition general instead of restricting to Riemannian metrics.

4. Generic Adaptation

Before elaborating on the possibilities for individual adaptation of the available algorithms to a non-standard norm $\|\cdot\|_g$ in Section 5, we discuss a generic way (with certain limitations) in the following.

4.1. Discrete Metric

Typical implementations of the abovementioned distance computation algorithms use the vertex coordinates to derive metric dependent properties like edge lengths, angles, and areas. In this way computations implicitly rely on the standard metric induced by M's embedding in Euclidean space.

By not taking any extrinsic vertex coordinates into account, but instead relying on intrinsic edge lengths, computed according to $\|\cdot\|_g$ as $\ell_g(e) = \|\vec{e}\|_{g_e}$, most distance computation algorithms can directly be adapted to non-standard norms (an exception is [CBK12], which relies on (relative) tangent plane orientation information). To that end one, where required, computes angles and areas based on the intrinsic edge lengths. The intrinsic area A of a triangle with edge lengths a, b, c can be computed using Heron's formula

$$A = \frac{1}{4}\sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}$$

and the inner angle α opposing the edge with length a using the half-angle theorem

$$\tan\frac{\alpha}{2} = \sqrt{\frac{(a-b+c)(a+b-c)}{(a+b+c)(-a+b+c)}}.$$

While being extremely simple, this generic strategy has a few disadvantages:

Loss of fidelity The metric information is injected solely via the intrinsic edge lengths, which specifiy the so-called *discrete metric*† of the mesh. While such a discrete metric captures all the information of a (sampled) Riemannian metric, we inevitably lose fidelity when discretizing a more general (non-elliptic) norm in this way.

Violation of triangle inequality The computed intrinsic edge lengths might not fulfill the triangle inequality everywhere (strictly speaking, they do not form a discrete metric in this case). We found this to rather be the typical behavior than the exception, especially for high degrees of anisotropy, as also described by Kovacs et al. [KMZ11]. For some algorithms this can be unproblematic, for others (which need

† In literature not related to meshes the term *discrete metric* by contrast homonymously refers to a metric which is 0 or 1 everywhere.

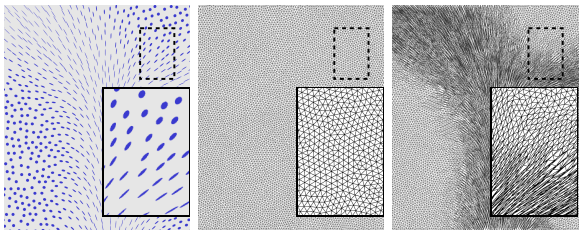


Figure 2: *Left: Anisotropic Riemannian metric, visualized through inverse tensor ellipses (intuitively showing the "speed profile" of the metric). Middle: Isotropic input mesh. Right: Corresponding intrinsic Delaunay retriangulation.*

to derive angles or areas from these lengths) this is catastrophic. Edge lengths $\ell'(e)$ fulfilling all triangle inequalities closest to the desired lengths could in such cases be found via a suitable inequality-constrained least-squares system:

$$\sum_e (\ell'(e)-\ell(e))^2 \to \min \quad \text{s.t.} \quad \ell'(e_i) \le \ell'(e_j)+\ell'(e_k)-\varepsilon$$

for all edge triples e_i, e_j, e_k incident to a common face, i.e. we have three inequality constraints per face.

Low triangulation quality The intrinsic roundness of the triangles is typically very bad, especially for higher degrees of anisotropy, i.e. there are angles close to 180° as well as angles close to 0°. For some algorithms this implies a low accuracy, for others a higher runtime. A remedy can be to retriangulate the mesh by means of an *intrinsic Delaunay triangulation* as detailed in the following.

4.2. Intrinsic Delaunay Triangulation

Using a Delaunay retriangulation algorithm which is based on an intrinsic discrete metric [FSSB07] we can adjust the connectivity of the mesh so as to obtain a mesh whose elements are nice with respect to this intrinsic metric (cf. Figure 2) – to the degree permitted by the given vertex set. A complete intrinsic remeshing, which not only adjusts the connectivity but also redistributes vertices, is a theoretical option that could lead to even better results but would be rather problematic depending on the application.

While this intrinsic Delaunay Triangulation (iDT) can significantly improve the distance computation results of several algorithms, there are some drawbacks, too:

- Numerical inaccuracies can hinder the iDT's correct execution and termination, thus demand epsilon tweaking,
- The application and underlying data structures must support non-regular meshes [FSSB07] (or the algorithm must be relaxed to avoid such configurations),
- The input edge lengths must fulfill the triangle inequality everywhere,
- The worst case runtime complexity is quadratic in the mesh size.

5. Individual Adaptation

We now outline the algorithm specific effects when using the generic adaptation possibilities presented in the previous section and show options for improved, individual adaptation where possible.

5.1. Dijkstra

Clearly, the simplest solution to compute approximations to distances with respect to non-standard metrics is to apply Dijkstra's algorithm, taking the intrinsic edge lengths of the discrete metric into account. Note that, due to the graph nature of the algorithm, fulfillment of the triangle inequality is not required. Unfortunately, the (already in the isotropic case relatively high) average approximation error increases with increasing anisotropy, quickly leading to unacceptably low accuracy. Using the iDT, however, accuracy can be brought closer to the level of the isotropic case. Figure 3 illustrates this using the setup from Figure 2 with anisotropy $\gamma = 20$.

The Dijkstra-based method of Lanthier [Lan99] which considers additional vertices and edges across faces can be adapted to the anisotropic case as follows: instead of deriving the intrinsic lengths of the additional edges from the discrete metric, we calculate the length of an edge across face f directly using $\|\cdot\|_{g_f}$. This allows for higher fidelity in the case of non-Riemannian metrics. Figure 4 illustrates the behavior for increasing numbers of added vertices and edges.

In the Dijkstra-based method of Campen et al. [CBK12] one can similarly compute intrinsic lengths for the additional edges directly based on the norms $\|\cdot\|_{g_e}$ instead of relying on the discrete metric approach, as described by the authors.

5.2. Fast Marching

The Fast Marching approach [KS98] can be applied in the case of a non-standard metric by relying on the corresponding discrete metric (which must fulfill the triangle inequality everywhere as we need to derive, e.g., angles from it). The general problem is that in this metric the number of obtuse triangles is often enormous, requiring the consideration of a vast number of virtual edges. An iDT can be used to avoid this – but has its own abovementioned shortcomings. Figure 5 illustrates the qualities of these options. Another option is to deal with the inaccuracies due to obtuse angles using recursive improvement techniques [KSC*07]; however, this means losing consistency and convergence properties.

Sethian and Vladimirsky [SV04] proposed a more general Ordered Upwind method (OUM) which is already specifically designed for non-standard metrics. It is able to consistently deal with arbitrary anisotropy, i.e. one, in theory, has the possibility to arbitrarily increase accuracy through mesh refinement. In comparison to the other discussed algorithms, the implementation is probably the most complex one and runtime and memory consumption is relatively

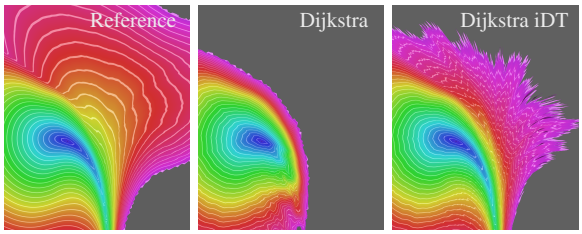


Figure 3: *Anisotropic distance field (up to a fixed maximum distance, so as to enhance clarity). Left: Highly accurate reference solution. Middle: Dijkstra result on an isotropic mesh. Right: Dijkstra result computed on the iDT.*

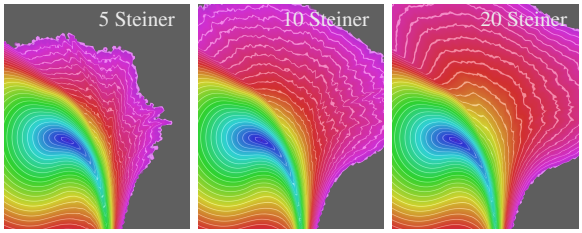
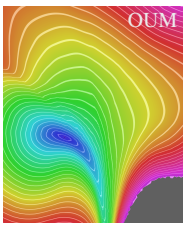


Figure 4: *Lanthier's method's distance field (on the isotropic mesh). Left: computed with 5 Steiner vertices per edge. Middle: 10 Steiner vertices. Right: 20 Steiner vertices.*

high (cf. Section 7). A main factor is that, in addition to the actual anisotropic distance propagation, as an ingredient isotropic distances from every single vertex of the mesh to all of its neighbors within a radius of $\|e_{\max}\|\gamma$ need to be computed. Here $e_{\max}$ is the longest edge of the mesh and γ the anisotropy, i.e. the radius can become quite large for high anisotropies. While simple extrinsic distances can be used for efficiency, higher accuracy on non-planar meshes is achieved by computing intrinsic isotropic distances.



It is worth noting that, while other methods typically overestimate distances, OUM can also heavily underestimate distances, as can be seen when comparing the inset figure to the reference solution. This is due to the fact that distances are propagated over virtual triangles that span a long distance across the mesh. While the norm potentially varies on the mesh under these virtual triangles, the propagation across them is computed atomically using only the norm at one end point, easily allowing for too long as well as too short results.

5.3. Heat Method

Just like the other algorithms discussed so far, the heat method [CWW13] can be adapted to non-standard metrics by formulating it in terms of the discrete metric, i.e. based on the intrinsic edge lengths. This amounts to calculating the cotangent weights, element areas, and divergence values involved in the Laplacian and the Poisson system accordingly.

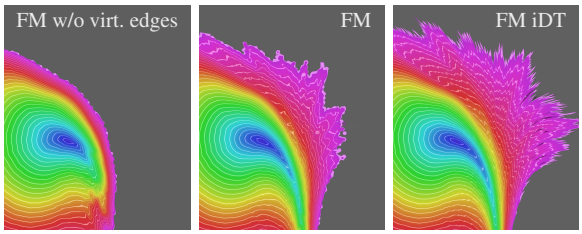


Figure 5: *Fast Marching distance field based on the discrete metric. Left: without virtual edges. Middle: with 13k virtual edges. Right: on the iDT, where only 6k virtual edges are necessary.*

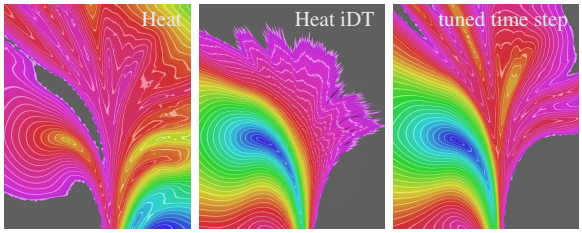


Figure 6: *Heat Method distance field based on the discrete metric. Left: computed on the isotropic mesh. Middle: computed on the iDT. Right: computed on the isotropic mesh but with individually tuned heat integration time step.*

Unless the anisotropy is very moderate, the low intrinsic element roundness, however, does negatively affect robustness. The resulting distance fields then often show distortions and degeneracies like local minima. We observed that improved results can be achieved by individually tuning the heat integration time step instead of using the standard $c = 5$ proposed in the original publication, but found no general rule for a good automatic choice. Again, an iDT can be an option to remedy these problems (cf. Figure 6).

6. Short-Term Vector Dijkstra

We now present a novel method for the approximate computation of distances with respect to arbitrary metrics. In order to develop an intuitive understanding of the underlying principle, let us consider Dijkstra's classical algorithm again. Due to its graph nature, the distance approximations resulting from this algorithm are the lengths of edge paths that meander over the surface (cf. Figure 7 left) – not the lengths of true geodesic paths. They are thus rather inaccurate and also very triangulation dependent.

Instead of first measuring lengths of edges and then (scalarly) summing these, an interesting alternative is to first (vectorially) sum edges and then measure the length of the sum. This *vector-valued Dijkstra* algorithm has been employed by Schmidt et al. [SGW06] to obtain geodesic distances for the purpose of local surface parameterization. Figure 7 illustrates the principle in the plane. In this planar case and with the standard metric the resulting distances are ac-

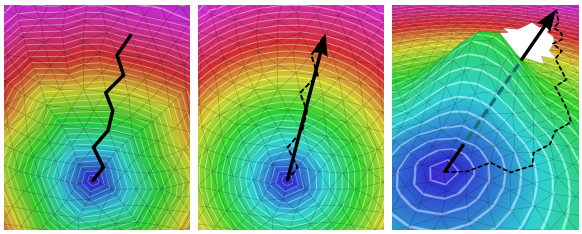


Figure 7: *Left: distance computation using Dijkstra's classical algorithm. Middle: computation using a vector-valued Dijkstra variant. Right: shortcomings of the vector-valued Dijkstra variant: it is oblivious to holes, obstacles, and geometric variations.*

tually exact – as, independent of the path, the result is the Euclidean length of the vector pointing from source to target point. The concept can be transferred to 2-manifold meshes by performing vector addition after transferring the vectors into a common 2D reference system, which can be done in different ways [SGW06, Sch13]. In any case, a major problem is that this method is basically oblivious to holes, obstacles, and, when applied in a non-planar setting, geometric variations in the surface. Figure 7 right demonstrates this issue. Hence, while being adequate and efficient for computations in a local neighborhood, it is unsuitable for global distance computations on manifolds.

The idea underlying our *Short-Term Vector Dijkstra* (STVD) is to form a hybrid out of the classical scalar-valued variant and the vector-valued variant so as to combine the respective advantages. Conceptually, we equip the scalar-valued variant with a vector-valued short-term memory. In this way the meanders of the edge paths through the triangulation can locally be smoothed out without globally disregarding the surface geometry. The following pseudo code clarifies the details.

Algorithm: Short-Term Vector Dijkstra (STVD)
Input: polygon mesh, metric g, a vertex designated *source*
Output: vertex based field of geodesic distances to *source*

---

source.dist $\leftarrow 0$ all other distance values initially ∞
Q.insert(*source*) priority queue Q ordered by distance
while not Q.empty
  $v \leftarrow Q$.extract_minimum() get min. dist. vertex out of Q
  v.final $\leftarrow$ true
  for all w adjacent to v **where not** w.final
    if update_dist(v,w) < w.dist
      w.dist $\leftarrow$ update_dist(v,w)
      w.pred $\leftarrow v$
      if not w in Q
        Q.insert(w)

This looks very much like the standard Dijkstra algorithm and indeed, if we use the following version of the update_dist function this is exactly what we get.

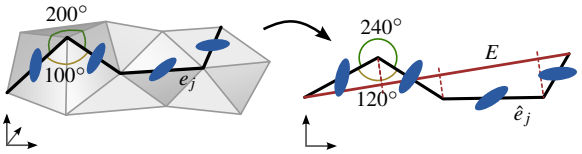


Figure 8: *Unfolding of edge chains to the plane. Edge lengths and relative 1-ring angles are preserved. The sum vector E (red) is then subdivided according to the ortho-projection of the individual edges. The resulting portions are measured by the respective norms $\|\cdot\|_{g_e}$ – here visualized as tensor ellipses (blue) – and their lengths summed to get ℓ_g.*

Function: update_dist(v,w) original Dijkstra version

---

return v.dist + $\ell_g(v,w)$ add length of edge (v,w) w.r.t. g

---

By instead exploiting a vector-valued short-term memory (a window of k preceding edge vectors), we obtain our STVD algorithm using the following variant of the distance update function:

Function: update_dist(v,w) our STVD version

---

tmp $\leftarrow w$.pred
w.pred $\leftarrow v$
dist $\leftarrow \min_{i=1}^{k}\ w.\text{pred}^i.\text{dist} + \ell_g\left(\sum_{j=1}^{i}(w.\text{pred}^{j-1}, w.\text{pred}^j)\right)$
w.pred $\leftarrow$ tmp
return dist

---

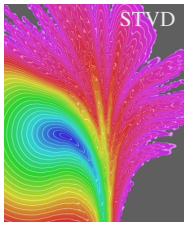
where the predecessor relation is recursively defined as $w.\text{pred}^{i+1} = w.\text{pred}^i.\text{pred}$, with $w.\text{pred}^0 = w$. It remains to be clarified how the edges $e_j = (w.\text{pred}^{j-1}, w.\text{pred}^j)$ are summed vectorially and how the length ℓ_g of the sum with respect to g is measured:

The norms $\|\cdot\|_{g_e}$ live in tangent planes, i.e. they allow to measure vector lengths in 2D. We thus unfold the chain of (directed) edges e_j into a common plane ($\mathbb{R}^2$) while preserving edge lengths and relative 1-ring angles. Figure 8 illustrates this. Note that no actual embedding of the mesh is required for this. The norms $\|\cdot\|_{g_e}$ (represented relative to their corresponding edges) now allow us to approximately measure the length of the sum $E = \sum_j \hat{e}_j$ of the unfolded edge vectors $\hat{e}_j$ in 2D. As these norms can differ, we decide how large portion of E is measured by which norm based on the edges' (signed) orthogonal projections onto E. Concretely, we use $\ell_g(\sum_j e_j) = \sum_j \hat{e}_j^\top \bar{E}\, \|\bar{E}\|_{g_{\hat{e}_j}}$ (clamped to $\mathbb{R}^{\geq 0}$), where $\bar{E} = E/\|E\|$, in the above distance update function.

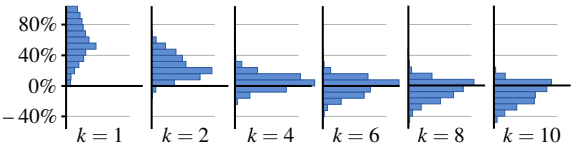
It is worth noting that we can choose between two alternatives regarding the representation of the conceptual memory. It can be represented either explicitly by storing vectors of the last $k-1$ edges $(v.\text{pred}^j, v.\text{pred}^{j+1})$ at each front vertex v (in its local 2D coordinate system), or implicitly by gathering these vectors using a short back-trace via the predecessor relation when we need them in the distance update function.

With an appropriate choice of the short-term memory's

depth k, the accuracy of the results is immensely improved. The inset illustrates this for $k = 10$ (detailed quantitative results are provided in Section 7). Despite the lack of smoothness in some regions, the result is closer to the reference than the alternatives from Section 5 (except Figure 4 middle, right).



Note that for $k = 1$ STVD is equivalent to Dijkstra's classical algorithm (generally overestimating distances), while for $k \to \infty$ it becomes an all vector-valued variant (which tends to underestimate unless the surface is developable and free of holes). This can also be observed in the following histograms which show the signed relative error distribution over all vertices of all examples from Section 7 (for $\gamma = 10$):



In a previous method [CBK12] also edge chains up to a length k were considered for distance computations. A fundamental difference is that the mesh graph is extended with additional edges, increasing their total number by about two orders of magnitude in practice. This has direct performance consequences as the runtime of Dijkstra's algorithm depends on the number of edges. By contrast, our method performs just one distance value update computation per original input edge, just like the standard Dijkstra algorithm. This difference is due to not constructing (in advance) *all* possible edge chains of length $\leq k$, but implicitly (on demand) only those which are actually proceeding in the direction of the propagating front. Further differences are that, due to the employed discrete exponential map, this previous method relies on an embedding of the surface, and the construction is specifically tailored to a direction field based metric.

6.1. Speed vs. Accuracy

Some of the discussed algorithms allow to trade speed for accuracy at the user's discretion. By using a higher number of Steiner vertices in the edge-subdivision approach [Lan99], or by refining the input triangle mesh using some steps of 1-to-4 splits prior to the application of the Ordered Upwind method [SV04] (or, in the case of a Riemannian metric, also the Fast Marching adaptation) higher accuracy can be achieved – at the cost of quickly increasing runtime. Note that such property is not to be taken for granted: for instance the accuracy of distance computations using Dijkstra's algorithm does not generally increase under mesh refinement.

We empirically observed that such property can also be established for our STVD. To this end we need to achieve two seemingly contradicting goals: k needs to be increased so as to increase the angular resolution of the distance propagation, while the lengths of the used vector sums need to be decreased so as to reduce the approximation errors of the unfolding-based measurement. We can achieve both by refining the mesh using 1-to-4 splits (reducing edge lengths by a factor of 2) while increasing k by a factor < 2. The following table shows the decreasing mean error for increasing levels of refinement on three exemplary models ($\gamma = 20$):

Level	k	ElkToy	Gargoyle	RockerArm
0	7	9.8%	25.7%	15.3%
1	10	5.1%	13.4%	11.6%
2	14	2.5%	8.2%	6.5%
3	20	1.5%	4.3%	3.9%

More interesting than this possibility, however, we deem that, even in the case of strong anisotropy, STVD achieves reasonable, relatively good results already on unrefined meshes of resolutions typically encountered in the Computer Graphics and Geometry Processing field (cf. Section 7).

6.2. Genericity

Polygonal Meshes It is worth noting that STVD does not rely on the property that M is a triangle mesh, i.e. it can also be applied to general polygonal meshes. Many other methods are designed specifically for triangle meshes and would need to be adapted – or the polygon mesh be triangulated.

3D Meshes While our focus here is on 2-manifold surface meshes, an interesting fact about STVD is that it can directly be applied also to volumetric meshes in $\mathbb{R}^3$ – whether they consist of tetrahedral, hexahedral, or general polyhedral cells. We simply skip the edge vector unfolding to the plane and sum the 3D edge vectors directly (cf. Figure 9). While also other methods can potentially be generalized to this setting, this is less trivial. Challenges lie in the correct determination of virtual simplices for the OUM or FMM, establishing an intrinsic Delaunay tetrahedralization or fixing violated tetrahedra inequalities, or in handling the large number of additional edges, which in the case of straight-

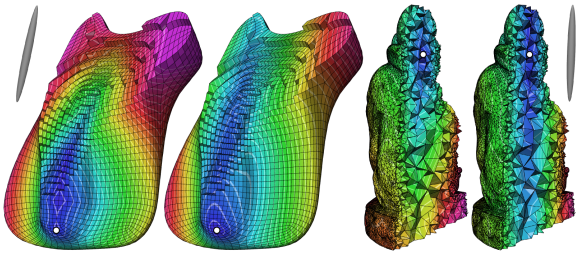


Figure 9: *Volumetric distance fields computed in a hexahedra and a tetrahedra mesh (sliced open to expose the interior). The constant Riemannian metrics used (depicted by ellipsoids next to the models) have an anisotropy of 12. Just like in the surface mesh case, Dijkstra's algorithm heavily overestimates distances, especially in vertical direction where true distances are shorter due to the anisotropy.*

forward generalization of Lanthier's concept to 3D grows quartically with the number of Steiner vertices per edge.

7. Results

In the previous sections we have discussed the qualitative differences of the approaches that are available for the computation of anisotropic distances. In order to get a quantitative understanding for the accuracy and runtime performance of all these options, we implemented all variants and performed extensive experiments using various kinds of input meshes (cf. Figure 10, 9K-183K triangles), metrics (vector field based as well as curvature tensor based), anisotropies (uniform as well as varying $\gamma(x)$), and algorithm parameters.

The reference solutions we compare against have been computed using the edge-subdivision method of Lanthier [Lan99] with 200 Steiner vertices per edge (and runtimes of up to several hours for a single distance field). In this setting for every single triangle there are more than 120,000 virtual edges crossing it, resulting in a highly accurate approximation of the true distances. We show the results using error-over-runtime plots in Figure 11 (and using error histograms in the supplemental material). Intuitively, the closer a method lies to the bottom left, the better – as this means that a low error (high accuracy) is achieved in a short time. These plots further allow to quickly see what other options apart from the "best one" are available, e.g. how much more accuracy can be achieved by spending how much more time.

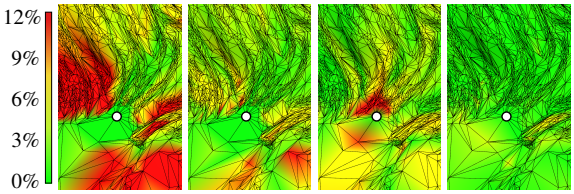
Regarding our STVD algorithm, we see that it lies in the bottom left region across the different levels of anisotropy, i.e. it achieves good results in short time when compared to the other options – especially for higher anisotropies. Good standard values for k can also be read from these plots: from around 5 for low anisotropy to around 10 for anisotropy 50. Note that such high anisotropy is not only of theoretical interest, e.g. anisotropies of 30+ have been used in [CBK12].

For lower anisotropies, the Fast Marching method applied to the intrinsic Delaunay triangulation of a subdivided version of the input mesh is very competitive. Note that this strategy requires several steps of preprocessing: 1) mesh subdivision, 2) discrete metric computation, 3) reestablishing triangle inequality fulfillment, and 4) iDT construction. This also implies the drawbacks discussed in Sections 4.1 and 4.2 and can take a considerable amount of (possibly amortizable) time. By contrast, STVD operates directly on the input mesh without the need for any preprocessing and without implying additional complexity.

Figure 10: *Models used for the evaluation, depicted with example fields. (From the AIM@SHAPE repository, the Image-based 3D Models Archive, Télécom Paris, or created using Cosmic Blobs® by Dassault Systèmes Solidworks Corp.)*

When a higher level of accuracy is required and more time is available, the edge subdivision method of Lanthier [Lan99] proves to consistently provide a good option.

In isotropic scenarios the advantage of STVD over, e.g., Fast Marching or the Heat Method diminishes. However, we observed that it is typically still very significant when dealing with meshes with badly shaped elements. This is illustrated here on an example mesh ($\gamma = 1$), where the error (w.r.t. exact geodesic distances [SSK*05]) is visualized:



8. Conclusion

We have explored and discussed numerous options for the computation of distance fields with respect to general anisotropic metrics, based on generic as well as specific adaptations of known algorithms. We compared all these in terms of their accuracy and runtime performance and enriched this zoo of methods with our Short-Term Vector Dijkstra. Despite its simplicity, this method proved to provide a very interesting novel option, allowing to quickly compute results with practical accuracy without the need for complex optimization or costly preprocessing.

In the future we would like to explore ways to enhance the smoothness of the results. Possibilities could be the use of averaging schemes during propagation [Sch13] or of a kind of "gradually fading memory" instead of a hard limit k. The use of a locally varying k (based on local feature size, anisotropy, mesh density) is another interesting direction.

Acknowledgements Funding for this work was provided by the DFG Cluster of Excellence *UMIC*, grant DFG EXC 89.

References

[BC11] BENMANSOUR F., COHEN L. D.: Tubular structure segmentation based on minimal path method and anisotropic enhancement. *Int. J. of Computer Vision 92*, 2 (2011), 192–210.

[BCLC09] BENMANSOUR F., COHEN L. D., LAW M. W. K., CHUNG A. C. S.: Tubular anisotropy for 2d vessel segmentation. In *CVPR* (2009), pp. 2286–2293.

[BPC08] BOUGLEUX S., PEYRÉ G., COHEN L. D.: Anisotropic Geodesics for Perceptual Grouping and Domain Meshing. In *ECCV* (2008), vol. 5303, pp. 129—-142.

[CBK12] CAMPEN M., BOMMES D., KOBBELT L.: Dual Loops Meshing: Quality Quad Layouts on Manifolds. *ACM Transactions on Graphics 31*, 4 (2012), 110:1—-110:11.

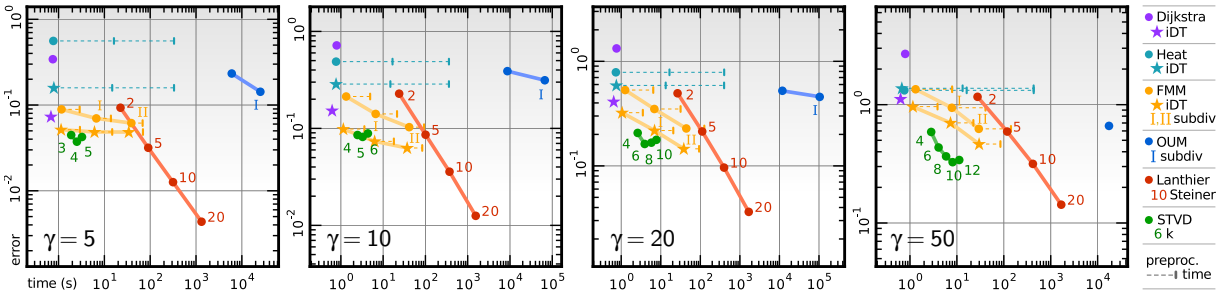


Figure 11: *Error-over-runtime plots of the results for various degrees of anisotropy (γ). Shown is the mean rel. abs. error (computed over all vertices) on the vertical axis, and the summed runtime (sec.) for the computation of a complete distance field per input test case (14 mesh/metric combinations). Mesh subdivision (1-to-4 split) levels are indicated by* I *and* II*; numbers indicate the STVD parameter k or the number of edge Steiner vertices, respectively. Preprocessing time (shown by dashed bars) includes triangle inequality fixing and (where applicable) subdivision and iDT. For the Heat method the first segment of the bar additionally indicates the time for setup and pre-factorization of the employed linear systems. Violated triangle inequalities have been fixed 1) for the heat method as described in Section 4.1 using* `Ipopt`*; 2) for the FMM using a simpler (not least-squares optimal) strategy of iteratively adjusting individual triangles, as on the subdivided meshes the solver takes unacceptably long.*

[CH90] CHEN J., HAN Y.: Shortest Paths on a Polyhedron. In *Proc. Symp. Comp. Geom.* (1990), pp. 360–369.

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in Heat. *ACM Transactions on Graphics* (2013).

[FSSB07] FISHER M., SPRINGBORN B., SCHRÖDER P., BOBENKO A. I.: An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. *Computing 81*, 2-3 (2007), 199–213.

[KMZ11] KOVACS D., MYLES A., ZORIN D.: Anisotropic quadrangulation. *Comp. Aided Geom. Design 28*, 8 (2011), 449–462.

[KS98] KIMMEL R., SETHIAN J. A.: Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. 95*, 15 (1998), 8431–8435.

[KS00] KANAI T., SUZUKI H.: Approximate Shortest Path on Polyhedral Surface Based on Selective Refinement of the Discrete Graph and its Applications. In *Geometric Modeling and Processing* (2000), IEEE Comput. Soc, pp. 241–250.

[KSC*07] KONUKOGLU E., SERMESANT M., CLATZ O., PEYRAT J.-M., DELINGETTE H., AYACHE N.: A recursive anisotropic fast marching approach to reaction diffusion equation: Application to tumor growth modeling. In *IPMI* (2007), pp. 687–699.

[Lan99] LANTHIER M.: *Shortest Path Problems on Polyhedral Surfaces*. PhD thesis, School of Computer Science, Carleton University, 1999.

[LMS97] LANTHIER M., MAHESHWARI A., SACK J.-R.: Approximating weighted shortest paths on polyhedral surfaces. *Proc. Symp. Comp. Geom. (SCG '97)* (1997), 274–283.

[LMS99] LANTHIER M., MAHESHWARI A., SACK J.-R.: Shortest Anisotropic Paths on Terrains. *ICAL 26* (1999), 523—-533.

[MMP87] MITCHELL J. S., MOUNT D. M., PAPADIMITRIOU C. H.: The Discrete Geodesic Problem. *SIAM Journal on Computing 16*, 4 (1987), 647–668.

[NK02] NOVOTNI M., KLEIN R.: Computing Geodesic Distances on Triangular Meshes. In *WSCG* (2002), Universität Bonn.

[PWKB02] PARKER G. J. M., WHEELER-KINGSHOTT C. A. M., BARKER G. J.: Estimating distributed anatomical brain connectivity using fast marching methods and diffusion tensor imaging. *IEEE Trans. Med. Imaging 21*, 5 (2002), 505–512.

[PWT05] PICHON E., WESTIN C.-F., TANNENBAUM A. R.: A Hamilton-Jacobi-Bellman approach to high angular resolution diffusion tractography. *Medical image computing and computer-assisted intervention 8*, Pt 1 (Jan. 2005), 180–7.

[RR90] ROWE N. C., ROSS R. S.: Optimal grid-free path planning across arbitrarily-contoured terrain with anisotropic friction and gravity effects. *IEEE Trans. Robot. Autom* (1990).

[Sch13] SCHMIDT R.: Stroke parameterization. *Comp. Graph. Forum 32*, 2 (2013).

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive Decal Compositing with Discrete Exponential Maps. *ACM Transactions on Graphics 25*, 3 (2006), 605—-613.

[SJC09] SEONG J.-K., JEONG W.-K., COHEN E.: Curvature-based anisotropic geodesic distance computation for parametric and implicit surfaces. *The Visual Computer 25*, 8 (Apr. 2009), 743–755.

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S. J., HOPPE H.: Fast exact and approximate geodesics on meshes. In *SIGGRAPH* (2005), vol. 24.

[SV00] SETHIAN J. A., VLADIMIRSKY A.: Fast methods for the Eikonal and related Hamilton- Jacobi equations on unstructured meshes. *Proc. Nat. Acad. Sci. 97*, 11 (2000), 5699–703.

[SV04] SETHIAN J. A., VLADIMIRSKY A.: Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms. *SIAM J. Num. Anal. 41*, 1 (2004), 325–363.

[Tsi95] TSITSIKLIS J. N.: Globally Optimal Trajectories. *IEEE Transactions on Automatic Control 40*, 9 (1995), 1528–1538.

[TWZZ07] TANG J., WU G.-S., ZHANG F.-Y., ZHANG M.-M.: Fast approximate geodesic paths on triangle mesh. *International Journal of Automation and Computing 4*, 1 (Jan. 2007), 8–13.

[WDB*08] WEBER O., DEVIR Y. S., BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics 27*, 4 (2008), 104:1—-104:16.

[XW09] XIN S.-Q., WANG G.-J.: Improving Chen and Han's algorithm on the discrete geodesic problem. *ACM Trans. Graph. 28*, 4 (2009), 104:1–104:8.

[YSS*12] YOO S. W., SEONG J.-K., SUNG M.-H., SHIN S. Y., COHEN E.: A triangulation-invariant method for anisotropic geodesic map computation on surface meshes. *IEEE Trans. Vis. Comput. Graph. 18*, 10 (2012), 1664–1677.