EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Stroke Parameterization

R. Schmidt

Autodesk Research
Toronto, Canada

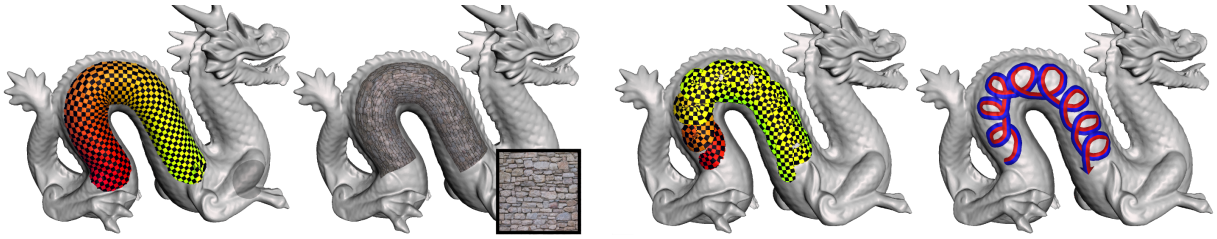

Figure 1: *Our technique parameterizes the geodesic neighbourhood around a curve embedded in a 3D surface, such that the parameterization is aligned with the stroke. This simplifies application of tiled and procedural stroke texture (left). We also handle self-intersecting strokes (right).*

Abstract

We present a novel algorithm for generating a planar parameterization of the region surrounding a curve embedded in a 3D surface, which we call a stroke parameterization. The technique, which extends the well-known Discrete Exponential Map [SGW06], uses the same basic geometric transformations and hence is both efficient and easy-to-implement. We also handle self-intersecting curves, for which a 1-1 map between the original surface and the plane is not possible. Stroke parameterizations provide an ideal coordinate space for solving a variety of computer graphics problems. We present applications including tiling texture and displacement along 3D brush strokes, procedural texturing along 3D paths, and user-guided crease extraction.

1. Introduction

Parameterization is a fundamental tool in computer graphics, and in the case of 3D surfaces, a wide range of parameterizations techniques have been developed to map 3D surfaces to useful spaces. Many works have considered the problem of embedding a disc-shaped region of a surface in the plane [FH05, SPR06], while more recently there has been interest in application-specific parameterizations such as the integral-coordinate embeddings used in quadrangulation [BZK09] and *decal* parameterizations used for local texture mapping [LHN05, SGW06, SGW09, CK11, MR12].

In this work we consider another type of local parameterization problem, namely how to parameterize the region around a curve embedded in a 3D surface. This problem arises in applying procedural textures along 3D brush strokes, and hence we refer to the solution as a *stroke parameterization*. As with decal parameterizations, stroke parameterizations are also useful in geometry processing contexts.

There are several desirable properties we would like a stroke parameterization to have (see Figure 2):

1. the parameterization should "follow the stroke", so that in parameter-space the stroke curve lies along the X axis
2. distortion should be minimized along the stroke curve
3. the mapping should be bounded, so that the distance to the X axis is less than the stroke width

Taken in combination, these properties (particularly Property 3) greatly simplify use of such stroke parameterizations in applications, as one can assume that the mapping lies within a predetermined bounding box. Existing parameterization algorithms are not designed to simultaneously control all these properties. In particular, a stroke parameterization *should* have areas of concentrated distortion in re-

gions where the stroke turns sharply. This runs counter to the premise of most existing techniques, which try to globally minimize distortion by spreading it evenly. In many application domains we also require fast computation on high-resolution meshes, making simple geometric algorithms desirable. Finally it can be the case that a curve embedded in a surface self-intersects. In these cases we must create a mapping that is not strictly 1-1.

In the following sections we will describe a geometric algorithm to compute stroke parameterizations. We first reformulate the Discrete Exponential Map [SGW06], adding a frame-propagation step that increases the accuracy of the exp/log map approximation. This new technique is then extended to propagate a parameterization from a curve, creating our stroke parameterization. We address self-intersecting stroke curves, and show how stroke parameterizations are useful in a variety of applications.

1.1. Related Work

Parameterizing the region around a 2D stroke for the purposes of applying procedural or stylized texture is a standard problem in NPR, and has been considered in many works [HL94]. Although explicit offset curves can be computed, more often one defines "rib vectors" at points along the stroke and creates a "thick polyline" [LM00]. This process can be applied in 3D, however for strokes on a surface the thick poly-path may intersect the surface.

Planar parameterization of arbitrary open surfaces has been studied extensively, and we refer the reader to recent surveys [FH05, SPR06] for a thorough treatment. Here we focus on parameterization techniques designed to flatten a local region in the neighbourhood of a point or curve.

Pedersen [Ped95] provides an early example, where the user specifies four geodesic boundary curves and the interior region is parameterized by cross-geodesics. The resulting "decals" were used to interactively texture surfaces. Lefebvre et al [LHN05] perform a similar task with local planar projections, while Schmidt [SGW06] created decal parameterizations using an approximation to the *exponential map* of differential geometry. This Discrete Exponential Map (DEM) has been applied in various domains such as procedural texturing [CG07, MCL*09], geometric modeling [TSS*], and geometry processing [WW11], and will form the basis for our approach.

The DEM does have limitations. For one, it can distort dramatically in high-curvature regions. Attempts have been made to address this via mass-spring relaxation [SGW09] and more robust geodesic computation [CK11]. The latter method also addresses the other major failing of the DEM, namely geodesic inaccuracy. Alternatives such as computing the gradient of the geodesic distance field [Bru08] or Discrete Geodesic Polar Coordinates [MR12] also produce much more accurate normal coordinates.

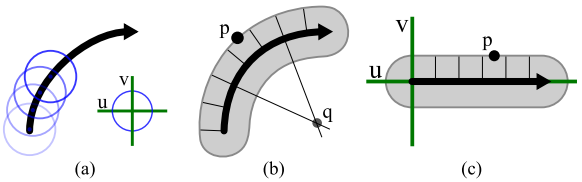

Figure 2: *Traditionally surface strokes have been textured by (a) combining separate* stamps *with individually-computed parameterizations. Our goal is to map (b) the region within some geodesic distance to the stroke to (c) a planar parameterization where the stroke lies along the unit axis. This mapping is not uniquely defined at points* **q** *which are equi-distant to multiple points on the stroke.*

None of these existing approaches address the problem of parameterizing the region surrounding a curve on a surface. Biermann et al's [BMBZ02] *spine parameterization* does map the region around a central curve to the plane, but this is accomplished using the Angle-Based Flattening conformal parameterization [SdS01], which does not actually take the curve into account.

2. Background

Our development of stroke parameterization is motivated by the problem of applying procedural maps (texture, displacement, etc) along a 2D path or *stroke* [HH90] lying on a 3D surface. For example, in 3D painting and sculpting tools the artist can select a *stamp* (a 2D image) and then apply that stamp along the stroke. For many stamps a convolution of the stamp with the stroke is the desired effect. In practice this is approximated by applying the stamp separately, in individually-computed 2D parameterizations, at regular increments along the stroke (Figure 2a).

Consider, however, a case where the stamp pattern could be *tiled*, as in Figure 1. Placing each stamp individually will not result in a correct tiling because the per-stamp local parameterizations not be continuous in where they overlap. To properly tile the stamp, we need a single consistent parameterization that follows the stroke (Figure 2b,c).

In the continuous domain a surface stroke is defined by (1) a curve $C(t)$ embedded in a 3D surface, parameterized by arc-length, and (2) a geodesic distance r. Points $\mathbf{x}$ such that geodesic distance $d_g(\mathbf{x}, C) < r$ can then be parameterized as

$$\mathcal{P}(\mathbf{x}) = (t_x \ , \ s(t_x) d_g(\mathbf{x}, C(t_x)) \ ) \qquad (1)$$

where $C(t_x)$ is the geodesically-closest point on C to $\mathbf{x}$ (Figure 3a), which we will denote C_x. We also require a sign $s(t_x)$ on the geodesic distance which indicates which "side" of C that $\mathbf{x}$ lies on, see below.

Note that for a given curved stroke, at some geodesic offset there will be points for which C_x cannot be uniquely de-

fined (Figure 2b). As a result, Equation 1 can only define a bijective mapping within some fixed geodesic offset from the stroke. This bijective region may not extend as far as our desired stroke width r, leaving our problem mathematically ill-posed (although our final algorithm will still do a reasonable job in such cases).

In practice, we are concerned with discretely-sampled domains where the surface is either a polygonal mesh or a point sampling with neighbourhood connectivity. In the case of a mesh, our parameterization algorithm operates only on the mesh vertices and mesh edge graph, although some of our applications involve a triangulation. The stroke is similarly a piecewise-linear curve defined by the points $\mathcal{S} = \{\mathbf{p}_i\}$ (Figure 3b). We assume that an orthonormal *Frame* $F_p = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{n}_p)$ is available at each point, where $\mathbf{n}_p$ (or $n(\mathbf{p})$) is the surface normal, defined by one of the many options available for discretized surfaces. For the points along the stroke, the frames are (approximate) Darboux frames, i.e. aligned such that $\mathbf{e}_1$ points along the tangent direction S' of S (estimated via finite differencing).

With these definitions we can define the sign computation mentioned above. Given the vector $\mathbf{v}$ in the tangent plane at S_x that points in the direction of the geodesic between S_x and $\mathbf{x}$, and tangent vector S'_x at S_x, $s(t_x) = \text{sign}(\,\mathbf{v} \cdot (S'_x \times n(S_x))\,)$.

Although various techniques are available to approximate the geodesic distance from S [CK11, CWW12], back-tracing through this distance field for each surface point $\mathbf{x}$ we wish to parameterize is prohibitively expensive [MR12]. Instead we propose a single-pass forward propagation to directly estimate the map $\mathcal{P}(\mathbf{x}) = (t_x, d_x)$ (to simplify exposition we will now use d_x as a shorthand for $s(t_x) d_g(\mathbf{x}, S(t_x))$).

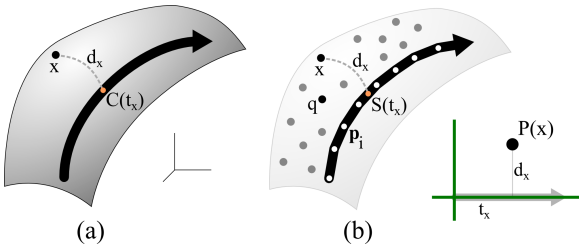

Figure 3: *The (a) continuous and (b) discretized versions of our stroke parameterization problem. See text for symbol definitions.*

2.1. Discrete Exponential Map

The *Discrete Exponential Map* (DEM) presented by Schmidt et al [SGW06] incrementally maps the region around a point $\mathbf{p}$ on a point-sampled surface to the *tangent space* T_p. Assuming some point $\mathbf{q}$ is already embedded in T_p, we can lift $\mathbf{q}$'s neighbour $\mathbf{x}$ into T_p by first estimating $T_q(\mathbf{x})$ via projection of $\mathbf{x}$ onto the tangent plane at $\mathbf{q}$ followed by re-scaling to preserve the distance $\|\mathbf{q} - \mathbf{p}\|$. We then transfer this vector to T_p via the 3D rotation aligning F_q to F_p, which reduces to a 2D rotation $R(\mathbf{q}, \mathbf{p})$ in the tangent space. Ultimately, then, $T_p(\mathbf{x}) = T_p(\mathbf{q}) + R(\mathbf{q}, \mathbf{p}) T_q(\mathbf{x})$.

The DEM utilizes a forward-propagation approach, where starting with $\mathbf{p}$, neighbours are incrementally lifted into T_p using the tangent-space vector summation described above. Applying Dijkstra's algorithm to the neighbour graph produces piecewise-linear shortest paths from $\mathbf{p}$ to its local neighbourhood, resulting in a suitable ordering of points.

Schmidt [Sch10] describes an improvement to the DEM called *upwind averaging*. In the description above, only one point $\mathbf{q}$ is used to determine $T_p(\mathbf{x})$, with $\mathbf{q}$ chosen based on the somewhat arbitrary ordering induced by the Dijkstra graph-distance. If $\mathbf{x}$ has another neighbour $\mathbf{r}$ for which $T_p(\mathbf{r})$ is already known, we can estimate $T_p(\mathbf{x})$ using both neighbours and average the result. A weighted averaging of predictions from all nearby known (or "upwind") neighbours significantly improves the robustness of the DEM.

Before we continue, a brief note about terminology. Although initially referred to as an "exponential" map [SGW06], the continuous process being approximated by the DEM is in fact the *log* map [Bru08], the inverse of the exponential map. However, the resulting pointwise correspondence between vertices and normal coordinates defines both *exp* and *log* at each input vertex.

3. Frame-Propagation DEM

In the DEM, each frame F_q is aligned with the fixed frame at F_p via a pair of rotations, to directly map vectors from T_q into T_p (Figure 4a). In this section we will reformulate the DEM to avoid this link between distant points, by propagating frames outward from $\mathbf{p}$ along with normal coordinates.

Assume we have a point $\mathbf{q}$ which is a neighbour of $\mathbf{p}$, and a third point $\mathbf{x}$ which is a neighbour of $\mathbf{q}$ but not $\mathbf{p}$. $T_p(\mathbf{q})$ is computed directly via projection, as is $T_q(\mathbf{x})$. In the DEM we would transform this vector to F_p via frame alignment, but another option would be to slide $T_q(\mathbf{x})$ along the (linear) curve between $\mathbf{q}$ and $\mathbf{p}$ via *parallel transport* (Figure 4b).

If we embed $T_q(\mathbf{x})$ in F_q, then parallel transport amounts to transforming F_q to align the normal n_q with n_p via the minimal rotation around $n_p \times n_q$. The inverse rotation will take F_p to F_q, and if we use this frame as the basis for T_q, we can compute $T_p(\mathbf{x})$ as the direct sum $T_p(\mathbf{q}) + T_q(\mathbf{x})$.

To generalize the above process, we can compute $T_p(\mathbf{x})$ at any point $\mathbf{x}$ with a known neighbour $\mathbf{q}$ by first shifting frame F_p to $\mathbf{q}$ via parallel transport, producing $F_{p \to q}$, and using this frame as the basis for the vector sum in T_q (Figure 4b). To transport an initial frame F_p along a path $\{\mathbf{q}_i\}$ through a point sampling, we need only compute the sequence of minimal rotations that aligns $n(\mathbf{q}_{i-1})$ with $n(\mathbf{q}_i)$ and apply

them to $\mathbf{e}_1(\mathbf{p})$ - the final $\mathbf{n}$ is the surface normal at the last point in the path, and $\mathbf{e}_2$ is defined by orthonormality.

On curved surface the effect of parallel transport on a vector depends on the type of *connection* in use. The approach we have described approximates the Levi-Civita connection, which accumulates an *angle defect* relative to the curvature along the transport path [CDS10]. So, the result of our frame propagation depends on the path taken from $\mathbf{p}$ to $\mathbf{x}$.

Since the path is somewhat arbitrary, we can transport F_p along multiple paths and average the result (Figure 4c). Note that as each resulting frame has the same $\mathbf{n}$, we need only average $\mathbf{e}_1$. Rather than explicitly compute multiple paths, we combine a propagation of frames with the propagation of normal coordinates. Just as $T_p(\mathbf{x})$ is determined for a point $\mathbf{x}$ by averaging estimates from multiple upwind neighbours $\mathbf{q}_i$, we also average the transported frames $F_{p\to q_i\to x}$.

The above process simply adds one more step to the DEM propagation. Note, however, that we have also discarded the global connection between $\mathbf{x}$ and $\mathbf{p}$ - the computation is now strictly local. As one might expect, the resulting normal coordinates are not identical to those of the original DEM. We find that they are in fact a more accurate approximation of the log/exp map. For example, the tangent-space vector sums used in the DEM assume a smooth connected neighbourhood, and significant deviations from analytic normal coordinates occur when this is not the case. Brun [Bru08] noted a serious problem with cones, which are developable except at the apex. Figure 5 demonstrates that our frame-propagation approach resolves this issue.

In Figure 6 we see the trade-off clearly - our approach "wraps around" the horn, resulting in a tear in the parameterization as the growing geodesic front self-intersects. In contrast, the DEM smoothly passes around the horn, but collapses much of the horn into itself in parameter space.

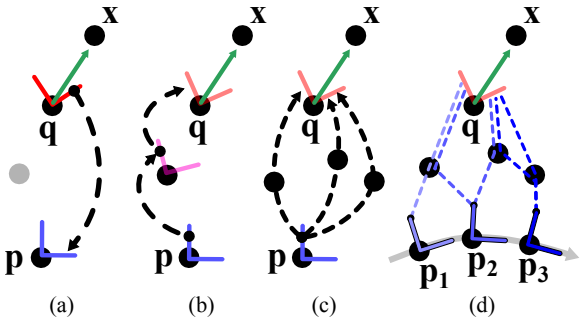

Figure 4: *In the DEM [SGW06], frames are (a) mapped from* $\mathbf{q}$ *directly to* $\mathbf{p}$*. We instead (b) parallel-transport the frame at* $\mathbf{p}$ *to* $\mathbf{q}$ *through intermediate points, and (c) average over multiple paths to get a smoother result. This path averaging extends (d) to multiple starting points, allowing our parameterization to grow outwards from a sampled curve.*

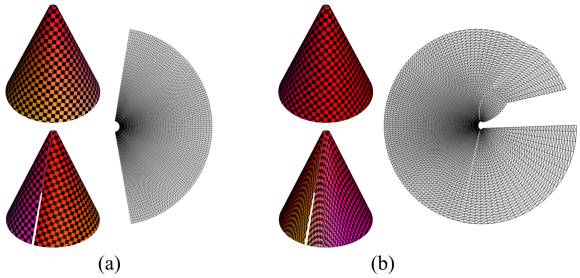

Figure 5: *Normal coordinates computed on a cone using (a) our frame propagation and (b) the original DEM.*

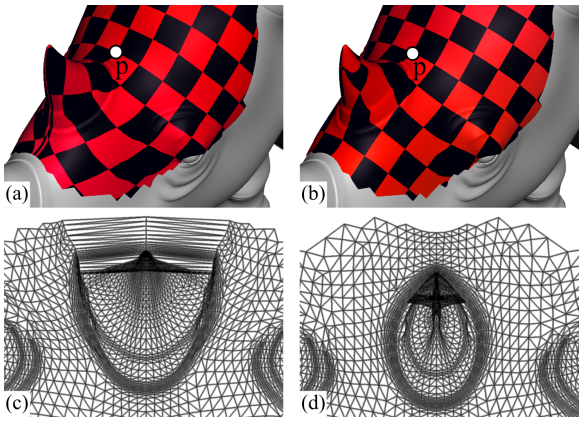

Figure 6: *Geodesic disc parameterized with normal coordinates estimated by (a,c) our frame propagation and (b,d) the original DEM.*

4. Stroke Parameterization

In the previous section we described an approach to computing normal coordinates around a point $\mathbf{p}$ which propagates frames in addition to planar coordinates. This process differs significantly from the original DEM because the computations are all strictly local - to compute a planar coordinate at $\mathbf{x}$ we need only the frames and planar coordinate values at its direct neighbours. There is no knowledge of the initial starting point $\mathbf{p}$.

This local property allows the frame propagation process to be adapted to multiple seed points. For our sampled stroke $\mathcal{S} = \{\mathbf{p}_i\}$ we have associated frames F_{p_i}. Clearly it should be the case that $\mathcal{P}(\mathbf{p}_i) = (\alpha(\mathbf{p}_i), 0)$, where $\alpha(\mathbf{p}_i)$ is the arc length along $\mathcal{S}$. We then insert each $\mathbf{p}_i$ into our point neighbourhood graph, then initialize Dijkstra's algorithm with the distance at each $\mathbf{p}_i$ set to 0. As the graph distance at a point $\mathbf{x}$ is fixed, we update its planar coordinate as

$$\mathcal{P}(\mathbf{x}) = \sum_{q_i \in N_u(x)} w(\mathbf{q}_i, \mathbf{x}) \left(\mathcal{P}(\mathbf{q}_i) + T_{q_i}(\mathbf{x})\right) \qquad (2)$$

where $N_u(\mathbf{x})$ is the set of neighbours of $\mathbf{x}$ that are already fixed by the propagation (ie "upwind" points). We set $w(\mathbf{q}_i, \mathbf{x})$ to be the inverse-distance weight $1/(\|\mathbf{q}_i - \mathbf{x}\| + \varepsilon)$,

where ε is a small multiple of machine precision. Next we compute the first tangent direction $\mathbf{e}_1$ at $\mathbf{x}$ as

$$\mathbf{e}_1(\mathbf{x}) = \sum_{q_i \in N_u(x)} w(\mathbf{q}_i, \mathbf{x}) R(\mathbf{q}_i, \mathbf{x}) \mathbf{e}_1(\mathbf{q}_i) \tag{3}$$

where $R(\mathbf{q}_i, \mathbf{x})$ is the minimal rotation that takes $\mathbf{n}_q$ to $\mathbf{n}_x$, i.e. the rotation around vector $\mathbf{n}_q \times \mathbf{n}_x$ by angle $\text{acos}(\mathbf{n}_q \cdot \mathbf{n}_x)$ As the normal $\mathbf{n}_x$ is fixed, we can find $\mathbf{e}_2(\mathbf{x})$ via orthonormality. The propagation is halted at $\mathbf{x}$ if $\|\mathcal{P}(\mathbf{x})\| > r$, the stroke width. We need not explicitly compute the sign of the geodesic relative to the stroke, as the local normal coordinates used in Equation 2 are signed.

Note that in Equation 3, each incoming $\mathbf{e}_1$ that we sum over is itself computed via a weighted average, and this continues back to the initial frames F_{p_i}. So, as in the single-point case, we are averaging across many paths to $\mathbf{x}$ (Figure 4d). In fact this averaging includes information from all stroke points within a sort of geodesic event horizon, so the process can be considered a sort of discretized integration.

Stroke parameterizations computed using our method are shown throughout the paper. In addition to the parameterization, our approach propagates an orthogonal vector field out from $\mathcal{S}$, which can be useful in various applications. See Figure 7 for an example.

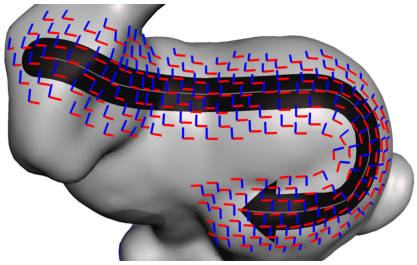

Figure 7: *An example of the vector field created by the propagation of frames out from the stroke curve.*

4.1. Handling Overlaps

So far we have assumed that each point $\mathbf{q}$ in the neighbourhood of $\mathcal{S}$ can be associated with a unique point on $\mathcal{S}$, and hence assigned a unique parameter value. This is not the case if the stroke self-intersects, as in Figure 8a, and Figure 9a shows the resulting failed parameterization. Conceptually, in this case we wish to assign multiple parameter values at some points. A straightforward way to do so is to create a separate copy of our surface data in the overlapping regions, as shown in Figure 8b.

Compared to 2D, determining where the geodesic neighbourhood around a stroke will overlap is complex. We describe a solution for mesh surfaces, which can be applied to point set surfaces with some additional complexity. Our approach is to slide a geodesic disc along $\mathcal{S}$, and accumulate a new mesh based on incremental appending of discs.

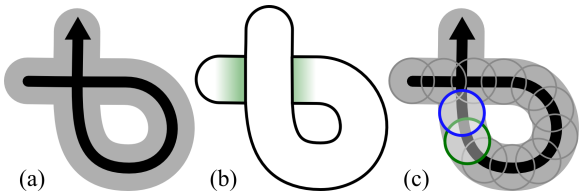

Figure 8: *We cannot assign a unique stroke parameter t to (a) points in the neighbourhood of a self-intersection. Instead we create a new surface which (b) overlaps at the self-intersection, by (c) appending sequential geodesic discs.*

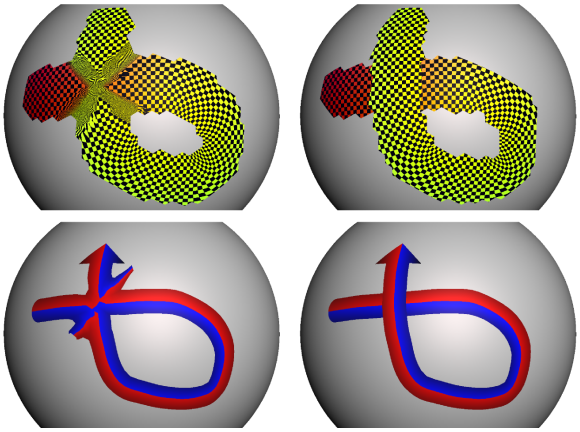

Figure 9: *Direct parameterization of (left) a self-intersecting stroke results in catastrophic failure. We (right) assign multiple parameter values to each vertex in the overlap region by generating copies with disconnected topology.*

The procedure is illustrated in Figure 8c. We denote by T_i the set of triangles contained within the geodesic disc centered at $\mathbf{p}_i$, and V_i the set of vertices contained in these triangles. We use approximate geodesic discs, found via Dijkstra's algorithm on the mesh edge graph. We initialize our new mesh with copies of V_0 and T_0. Then for disc $\mathbf{p}_i$, we append to the new mesh copies of the vertices vertices in V_i but not in V_{i-1}, ie the set $V_i \setminus V_{i-1}$. Then for each triangle in T_i we append a copy, linked to the appropriate new vertices, if it contains any vertex in $V_i \setminus V_{i-1}$.

If we apply this process in sequence of $\mathbf{p}_i$, and embed each $\mathbf{p}_i$ as we go along, the result is a new overlapping mesh that follows the stroke, with disconnected copies of the faces in the overlap region. The stroke parameterization is then computed on this new surface (Figure 9b).

Although we have parameterized a copy of the original mesh, each vertex in the new mesh maps corresponds to one of the original vertices, so we in effect have stored multiple parameter values for each vertex. How these multiple parameter values should be used is application-specific - for example, in texture painting the overlaps could be blended.

5. Evaluation

In Section 1 we stated that our goal was to compute a parameter value (t_x, d_x) for each point $\mathbf{x}$ in the neighbourhood of $\mathcal{S}$, but we have not *explicitly* computed either of these values. As we built upon an approach for computing normal coordinates around a point, there is some rationale for believing that we have in fact approximate geodesic distances. However, there is a simple test we can perform: given $\mathcal{P}(\mathbf{q}) = (u, v)$, we can measure the error $||v| - d_g(S(u), \mathbf{q})|$. Although this is only a *necessary* (i.e. not *sufficient*) condition for correctness, it is unlikely that our method could *consistently* produce an incorrect curve parameter and matching correct geodesic distance.

Figure 10 shows three test cases, where we have used the DEM to estimate d_g. The approximation is least accurate at sharp bends in $\mathcal{S}$, which is expected as in these regions the parameterization *should* be stretched or compressed. Note that where the checkerboard appears stretched, it is actually most compressed in the parameterization. Hence, the most visibly stretched regions have the lowest parameter-space sampling density, and we expect higher error there. As in the DEM, discontinuities in the geodesic distance field also result in non-smooth behavior.

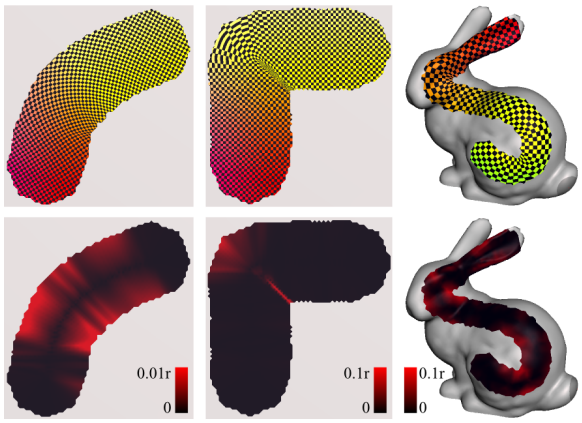

Figure 10: *Geodesic distance approximation error. In percentages of stroke width r, for (a) $\mu = 0.3\%$, $max = 1.6\%$, (b) $\mu = 0.4\%$, $max = 27.3\%$, (c) $\mu = 2.6\%$, $max = 17\%$.*

5.1. Comparison with Variational Parameterization

Most state-of-the-art planar parameterization techniques are variational, in that they attempt to find a low-distortion mapping via constrained energy minimization. We experimented with several such techniques, applied to the same geodesic neighbourhood as in our stroke parameterization. To make such techniques "follow" our input strokes, we added positional constraints.

Figure 11 shows a comparison between our method and Discrete Natural Conformal Parameterization (DNCP) [DMA02], which attempts to minimize angular distortion. Although the resulting parameterization does follow the stroke and is smoother than our geometric result, geodesic distances are not preserved. This is problematic because as shown in 11b the effective "width" of a mapped texture can vary unpredictably, while the result in 11d is unbounded in parameter space.

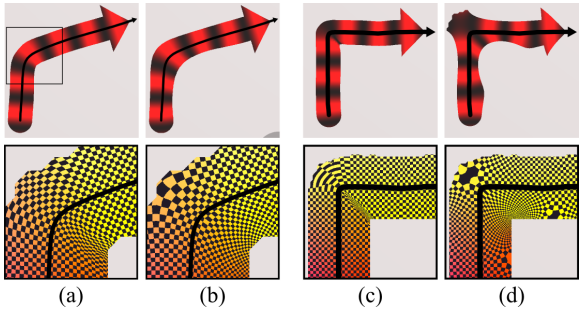

Figure 11: *Comparison of parameterizations computed using (a,c) our approach and (b,d) a conformal mapping.*

Figure 12 shows a comparison with As-Rigid-As-Possible mesh parameterization (ARAP) [LZX*08]. We tried two constraint techniques. In the first, existing vertices near the stroke are fixed to appropriate (t_x, d_x) values (this method was also used for DNCP). We also tried constraining barycentric points on the triangle interiors to the parameter-space stroke positions $(\alpha(\mathbf{p}_i), 0)$. We saw little difference between these two approaches.

It is clear that the ARAP optimization tries to pull the parameterization away from the constraints to reduce global distortion at the expense of local irregularities. Similarly nothing constrains ARAP or DNCP to maintain orthogonality to the input stroke, so in general they do not. These results are not unexpected - it is well-known that variational approaches suffer when point constraints attempt to introduce regions of high distortion. However, this is the very property which we expect a stroke parameterization to have.

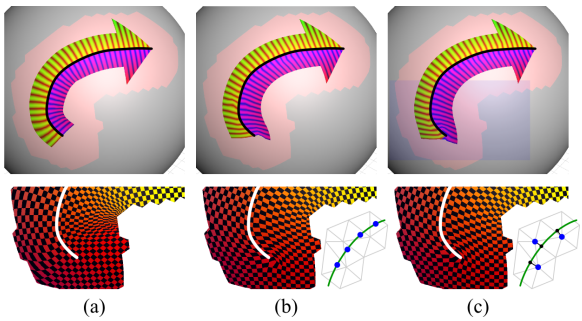

Figure 12: *Comparison of (a) our method with constrained ARAP, where in (b) vertices near the stroke are constrained, while in (c) barycentric points on face interiors are fixed.*

Finally, Figure 13 shows a comparison of ARAP and DNCP computation times versus our method (FP = Frame Propagation). We drew 30 example strokes on various meshes, and then sorted by total number of triangles in the parameterized region. It is clear from 13a that our method is generally an order-of-magnitude faster than ARAP. Note that for ARAP we only count the time to solve for a parameterization of the geodesic region, while the FP time includes the cost of finding this region. In 13b we break down the various steps of ARAP, which iteratively improves an initial parameterization. Following [LZX*08], we used DNCP, which itself is significantly more costly than the entire FP technique. Note also that the DNCP and ARAP solve steps employ highly-tuned state-of-the-art linear solvers, while our research code has seen only minimal optimization.

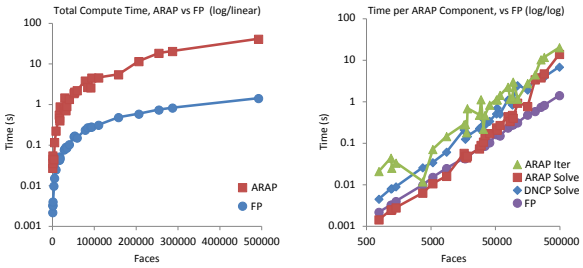

Figure 13: *Comparison (left) of computation times for our method (FP) and ARAP, and (right) breakdown of the various steps in ARAP computation. See text for details.*

5.2. Limitations

As with the DEM, there are limits to parameterization via discrete front propagation. Our approach is most effective when used locally, with relatively small stroke widths. If the desired stroke neighbourhood covers areas with large variations in surface curvature, the parameterization can degenerate. Similarly, very sharp bends relative to the stroke width lead to high distortion, see Figure 14 for a particularly difficult test case.

6. Applications

As described in Section 1, our initial motivation for stroke parameterization was to support tileable stamp textures along a user-drawn stroke. With the machinery we have developed in the previous sections this is completely straightforward, Figure 1 shows one such example.

Given its regular structure, our stroke parameter space is ideal for procedural texturing and stylization. For example, we can create *surface vector graphics* [SGW06] (Figure 15). An arrow along the stroke can be generated using a pixel shader which tests against the union of a circle, rectangle, and triangle laid out along the x axis. Similarly, long text strings can be added to the surface simply by rasterizing

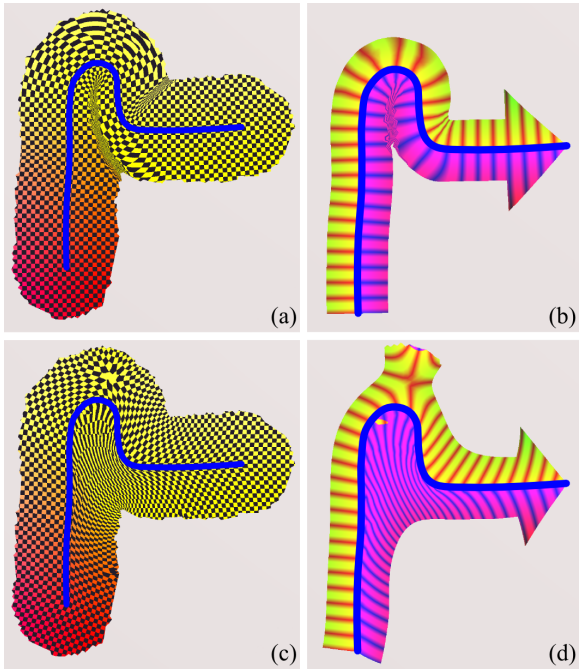

Figure 14: *A challenging test case, in which the stroke width is larger than the turning radius of the stroke. In such cases our method (a,b) compresses the parameterization at regions that correspond to geodesic discontinuities. These areas are quite noisy, as are areas of extreme stretching. In contrast, constrained ARAP (c,d) can smooth out the parameterization, but at the cost of undesirable changes in the shape of the mapped arrow.*

in 2D along the x axis. This simplifies the one-DEM-per-character approach taken in previous works [CG07].

Figure 16 shows a procedural displacement map generated along a stroke, as would be done in a 3D sculpting tool. Even on this high resolution (3m triangles) mesh, our stroke parameterization is fast enough to provide immediate feedback (see video). Stroke parameterization could also be used in systems like GeoBrush [TSS*], where the radial DEM parameterization is not suitable for elongated features. As in that system, geometry can be processed in parameter space, for example to refine the mesh so it can better approximate a procedural displacement.

6.1. Crease Extraction

Local parameterizations can be useful spaces for geometry processing, as shown in previous works [WW11,MR12]. We explore *snapping* a stroke to surface feature curve. This has been considered in many previous contexts [FKS*04], but existing methods tend to be specific to the type of feature being detected, and hardening the algorithms against issues like noise or weak features can be very complex. Using a stroke parameterization, we can sample an arbitrary function

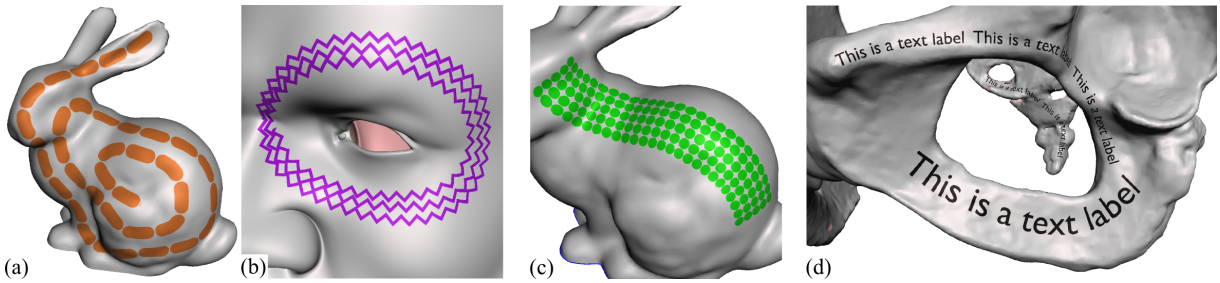

Figure 15: *Examples of vector-graphic style elements on surfaces - (a) dashed line, (b) patterned line, (c) patterned fill, and (d) text aligned with along a surface path.*

over the parameter space and create a 2D image, then apply standard feature detection methods from image processing, and map the results back to the 3D surface.

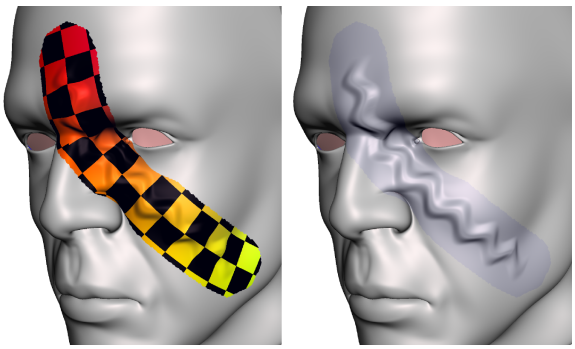

Figure 16: *Procedural displacement of a high-resolution (3m triangles) mesh along a stroke.*

Figure 17 shows several examples, where we sample mean surface curvature in the stroke parameterization. A simple snakes-style algorithm [STG98] is then used to fit a smoothed polyline to the ridge of locally-maximal values in the image. After mapping back to 3D the curves provide excellent approximations of the high-curvature features. As the initial parameterization is distorted, we can iterate this process to further refine the 3D curve, although we have yet to see significant improvement after the second iteration. This approach is effective even in the presence of very weak features, such as in Figure 17b.

Figure 18 shows another example where the function we sample is the dot product of the surface normal and the viewing vector, resulting in occluding contours from the current viewpoint. Note that the basic strategy we have implemented is sensitive to initialization, however far more robust snake techniques exist and are trivially applicable.

7. Discussion

We have presented a novel technique for parameterizing the geodesic neighbourhood around a curve on a surface, which we call a *stroke parameterization*. The frame-propagation that is the basis for our technique also improves accuracy in the well-known Discrete Exponential Map [SGW06]. We showed that our approach produces reasonable geodesic distance approximations, compares well to conformal parameterization under high distortion, and can be extended to handle self-intersecting stroke curves.

One interesting issue which we have yet to address is surface curves that form closed loops. In this case it would be useful to be able to compute a mapping onto a cylinder rather than a plane. This seems relatively tractable if we can replace the internal mathematics with operations that operate in a modulo space.

Another improvement would be better handling of sharp turns. Currently the resulting geodesic discontinuity creates a highly compressed region of parameter space. It may be better to create overlap regions here, however our technique is not yet able to do this, as the set of "live" triangles does not significantly differ as we go around the bend.

Finally, in some cases we may be willing to trade geodesic accuracy for smoother parameterization. Recent methods for efficiently computing smoothed geodesic distances [CWW12] could be applied here. Smoother parameterizations would be particularly useful for contour extraction. Another interesting possibility is to use our stroke parameterizations to track contours that move over time. The extended parameterization space could greatly simplify this sort of tracking, which may be particularly useful in imposing frame coherence on NPR renderings.

References

[BMBZ02] BIERMANN H., MARTIN I., BERNARDINI F., ZORIN D.: Cut-and-paste editing of multiresolution surfaces. *ACM Trans. Graph. 21*, 3 (2002), 312–321. 2

[Bru08] BRUN A.: *Manifolds in Image Science and Visualization*. PhD thesis, Linkoping University, 2008. 2, 3, 4

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009). 1

[CDS10] CRANE K., DESBRUN M., SCHRÖDER P.: Trivial connections on discrete surfaces. *Comp. Graph. Forum 29*, 5 (2010). 4

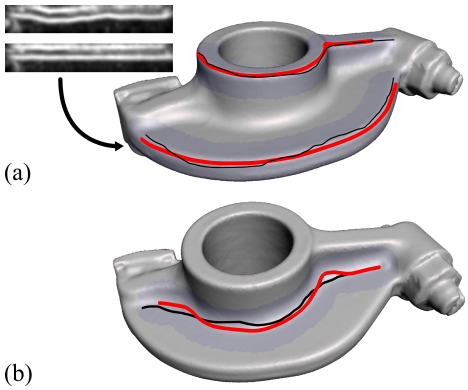

Figure 17: *Hand-drawn strokes (black) are aligned (red) to nearby curvature ridges. Parameter-space curvature image maps from first and second iteration of one curve in (a) are shown in top left. In (b), the stroke is aligned along a very faint curvature minima.*

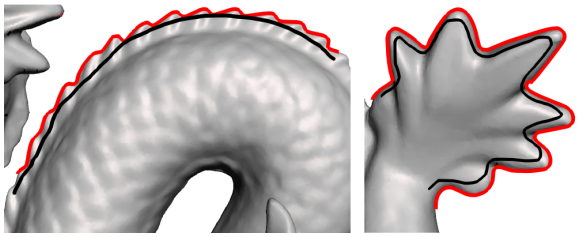

Figure 18: *Hand-drawn strokes (black) are snapped (red) to nearby silhouette contours.*

[CG07] CIPRIANO G., GLEICHER M.: Molecular surface abstraction. *IEEE Trans. Vis. and Comp. Graph. 13* (2007), 1608–1615. 2, 7

[CK11] CAMPEN M., KOBBELT L.: Walking on broken mesh: Defect-tolerant geodesic distances and parameterizations. *Comp. Graph. Forum 30*, 2 (2011), 623–632. 1, 2, 3

[CWW12] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat. *CoRR abs/1204.6216* (2012). 3, 8

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. In *Proc. Eurographics* (2002). 6

[FH05] FLOATER M., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*. Springer Verlag, 2005, pp. 157–186. 1, 2

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2004). 7

[HH90] HANRAHAN P., HAEBERLI P.: Direct wysiwyg painting and texturing on 3d shapes. *SIGGRAPH Comp. Graph. 24*, 4 (1990), 215–223. 2

[HL94] HSU S. C., LEE I. H. H.: Drawing and animation using skeletal strokes. In *Proc. SIGGRAPH '94* (1994). 2

[LHN05] LEFEBVRE S., HORNUS S., NEYRET F.: Texture sprites: Texture elements splatted on surfaces. In *Proc. I3D '05* (2005). 1, 2

[LM00] L. J. N., MARKOSIAN: Artistic silhouettes: a hybrid approach. In *Proc. NPAR '00* (2000), pp. 31–37. 2

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S.: A local/global approach to mesh parameterization. In *Proc. SGP '08* (2008), pp. 1495–1504. 6

[MCL*09] MITRA N., CHU H.-K., LEE T.-Y., WOLF L., YESHURUN H., COHEN-OR D.: Emerging images. *ACM Trans. Graph. 28*, 5 (2009). 2

[MR12] MELVÆR E. L., REIMERS M.: Geodesic polar coordinates on polygonal meshes. *Comp. Graph. Forum 31*, 8 (2012). 1, 2, 3, 7

[Ped95] PEDERSEN H. K.: Decorating implicit surfaces. In *Proceedings of SIGGRAPH 95* (1995), pp. 291–300. 2

[Sch10] SCHMIDT R.: *Part-Based Representation and Editing of 3D Surface Models*. PhD thesis, University of Toronto, 2010. 3

[SdS01] SHEFFER A., DE STURLER E.: Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers 17*, 3 (2001), 326–337. 2

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive decal compositing with discrete exponential maps. *ACM Transactions on Graphics 25*, 3 (2006), 605–613. 1, 2, 3, 4, 7, 8

[SGW09] SCHNEIDER J., GEORGII J., WESTERMANN R.: Interactive geometry decals. In *Proc. VMV 2009* (2009). 1, 2

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Found. Trends. Comp. Graph. Vis. 2*, 2 (2006), 105–171. 1, 2

[STG98] SINGH A., TERZOPOULOS D., GOLDGOF D.: *Deformable Models in Medical Image Analysis*. IEEE Computer Society Press, 1998. 8

[TSS*] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.:. *Comp. Graph. Forum*, 2. 2, 7

[WW11] WEI L.-Y., WANG R.: Differential domain analysis for non-uniform sampling. *ACM Trans. Graph. 30*, 4 (2011). 2, 7