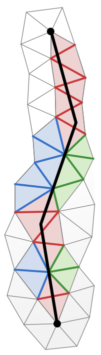
Quad Layout Embedding via Aligned Parameterization

Appendix

Appendix A: Layout Representation

The edge and face sequences corresponding to an arc are obtained as follows: for an oriented embedded arc a let $\gamma'(a)$ be the sequence of edges crossed (shown in red) when traveling along a from start to end. Every two subsequent edges in $\gamma'(a)$ share a common incident face except for when a runs through a vertex. In these cases we can additionally include the edges incident to the vertex from the left (blue) or the right (green) with respect to a. Let γ'_l and γ'_r denote the operators that return the two corresponding alternatives. We can use either one as discrete representation for the arc, but define both versions here as we need them in the next sections.



Where the distinction is unnecessary, we omit the subscript and refer to them as γ'. Note that $\gamma'_l(a) = -\gamma'_r(-a)$, where the minus-operator reverses the edge sequence and the arc orientation, respectively. We will also use the sequences of faces incident to the edges in $\gamma'(a)$, denoted by $\gamma(a)$. If $\gamma'(a)$ is empty, i.e. start and end node are incident to common faces, $\gamma(a)$ is defined to be the one of them which contains a.

The layout's rotation system (σ, θ) can be derived from the γ-paths as illustrated in Figure A.

Appendix B: Turning Numbers

In the discrete setting (on the mesh $\mathcal{M}$) one specifies the turning numbers for cycles of faces (or equivalently dual edges) of $\mathcal{M}$. Note that the signs of turning numbers depend on the cycle orientation.

For an irregular node h, the turning number t necessary for the face cycle clockwise around $\nu(h)$ (cf. Figure B left) is simply determined from the valence v of h as $t = -\frac{1}{4}v$. For nodes on boundary vertices no turning number needs to be prescribed.

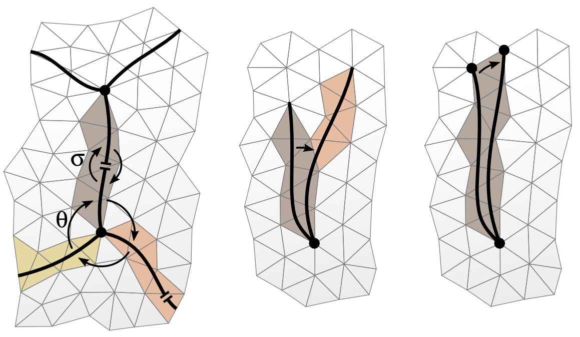


Figure A: *σ is trivially defined between the half-arcs of an arc. The definition of θ can be derived from the cyclic order of the γ-paths around a node. In the simplest case this can be decided directly at the node (left), if some paths coincide at first the order can be decided at the point where they diverge (center), or at the latest at the final vertices (right).*

A set of $2g$ homology generator cycles of $\mathcal{L}$ can easily be computed using the homotopy basis construction algorithm for combinatorial surfaces described in [EW05]: A spanning tree T of the layout graph G and a spanning tree T^* of the dual graph which does not cross T are computed. $2g$ edges of G will then not be contained in T nor crossed by edges of T^*. Connecting these $2g$ edges to the root vertex of T through T yields the $2g$ arc cycles. For a cycle of arcs $c = a_1 a_2 \cdots a_n$ (with consistent orientation of the a_i) we consider the corresponding face cycle $\gamma_l(c)$ – as the combinatorial surface $\mathcal{L}$ is homotopic to $\mathcal{S}$, these face cycles are homology generator cycles of $\mathcal{S}$. We count the number n_a of arcs (emanating from the n involved nodes) this cycle crosses. The turning number of the cycle then needs to be fixed to $t = \frac{1}{4}(n - n_a)$; in Figure B bottom left the depicted face cycle c crosses 2 arcs (dashed) which emanate from the 2 involved nodes, thus $t = 0$ in this case. The choice of γ_l (not γ_r) is to get sign compatibility with the above node turning number sign choice.

For each boundary loop d of $\mathcal{M}$ we find the cycle of arcs $c = a_1 a_2 \cdots a_n$ closest to d, i.e., such that no node lies between c and d, and choose its orientation such that d lies right of c when traveling along c. We then determine the turning number for $\gamma_l(c)$ as in the previous case. Figure B right illustrates this for the gray mesh boundary loop d: $t = \frac{1}{4}(4 - 4) = 0$ in this case.

Appendix C: Gradient Estimator

In order to compute the gradient descent vector $\tilde{\mathbf{d}}$ for a node currently embedded in vertex a, we first need to obtain a local 2D coordinate system of a's 1-ring. To this end we employ the commonly used geodesic polar map [WW94], effectively flattening the 1-ring to the plane while preserving

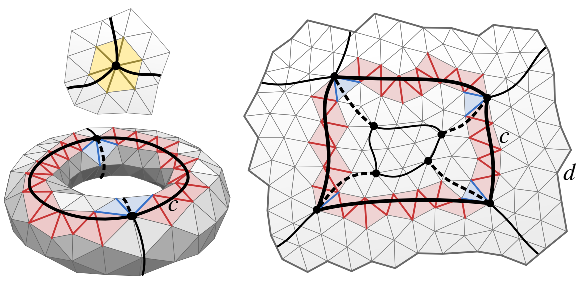
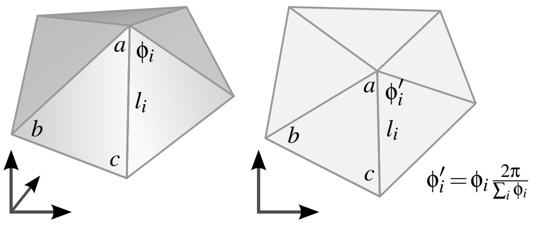


Figure B: *Visualization of the face cycles used for fixing the* turning numbers *of singularities (top left), homology generators (bottom left), and of boundaries of the mesh (right).*

radial lengths (l_i) and relative angles by uniformly scaling the inner angles ϕ_i incident to a such that they sum to 2π; origin and axes in the plane can be chosen arbitrarily.



Let (a_x, a_y), (b_x, b_y), and (c_x, c_y) denote the 2D coordinates of a triangle t's vertices in this system, and (a_u, a_v),(b_u, b_v),(c_u, c_v) their current (u, v) parameters. $\mathbf{u}_t$ and $\mathbf{v}_t$ are the first and second cross field vectors in t (expressed in the 2D system). The (per triangle) gradient $(\frac{\partial}{\partial a_x}\tilde{E}_t, \frac{\partial}{\partial a_y}\tilde{E}_t)$ at center vertex a in t can then be computed based on well-known expressions for triangle area A and triangle gradient ∇ as described in the following pseudo code, where we partly use (x, y) as a short-hand for (a_x, a_y).

$$A = \tfrac{1}{2}(a_x(b_y - c_y) + b_x(c_y - a_y) + c_x(a_y - b_y))$$
$$\tfrac{\partial}{\partial x}A = \tfrac{1}{2}(b_y - c_y)$$
$$\tfrac{\partial}{\partial y}A = \tfrac{1}{2}(c_x - b_x)$$
$$H_u = a_u\left(b_y - c_y, c_x - b_x\right) + b_u\left(c_y - a_y, a_x - c_x\right) + c_u\left(a_y - b_y, b_x - a_x\right)$$
$$H_v = a_v\left(b_y - c_y, c_x - b_x\right) + b_v\left(c_y - a_y, a_x - c_x\right) + c_v\left(a_y - b_y, b_x - a_x\right)$$
$$\nabla u = \tfrac{1}{2A}H_u$$
$$\nabla v = \tfrac{1}{2A}H_v$$
$$\tfrac{\partial}{\partial x}\nabla u = \tfrac{1}{2A}\left(0, b_u - c_u\right) - \tfrac{1}{2A^2}H_u\tfrac{\partial}{\partial x}A$$
$$\tfrac{\partial}{\partial y}\nabla u = \tfrac{1}{2A}\left(c_u - b_u, 0\right) - \tfrac{1}{2A^2}H_u\tfrac{\partial}{\partial y}A$$
$$\tfrac{\partial}{\partial x}\nabla v = \tfrac{1}{2A}\left(0, b_v - c_v\right) - \tfrac{1}{2A^2}H_v\tfrac{\partial}{\partial x}A$$
$$\tfrac{\partial}{\partial y}\nabla v = \tfrac{1}{2A}\left(c_v - b_v, 0\right) - \tfrac{1}{2A^2}H_v\tfrac{\partial}{\partial y}A$$
$$\begin{aligned}\tfrac{\partial}{\partial x}\tilde{E}_t &= 2\left(\tfrac{\partial}{\partial x}\nabla u\right)^{\mathsf{T}}\left(\nabla u - \mathbf{u}_t\right)A + \left(\nabla u - \mathbf{u}_t\right)^{\mathsf{T}}\left(\nabla u - \mathbf{u}_t\right)\tfrac{\partial}{\partial x}A\\ &\quad + 2\left(\tfrac{\partial}{\partial x}\nabla v\right)^{\mathsf{T}}\left(\nabla v - \mathbf{v}_t\right)A + \left(\nabla v - \mathbf{v}_t\right)^{\mathsf{T}}\left(\nabla v - \mathbf{v}_t\right)\tfrac{\partial}{\partial x}A\end{aligned}$$
$$\begin{aligned}\tfrac{\partial}{\partial y}\tilde{E}_t &= 2\left(\tfrac{\partial}{\partial y}\nabla u\right)^{\mathsf{T}}\left(\nabla u - \mathbf{u}_t\right)A + \left(\nabla u - \mathbf{u}_t\right)^{\mathsf{T}}\left(\nabla u - \mathbf{u}_t\right)\tfrac{\partial}{\partial y}A\\ &\quad + 2\left(\tfrac{\partial}{\partial y}\nabla v\right)^{\mathsf{T}}\left(\nabla v - \mathbf{v}_t\right)A + \left(\nabla v - \mathbf{v}_t\right)^{\mathsf{T}}\left(\nabla v - \mathbf{v}_t\right)\tfrac{\partial}{\partial y}A\end{aligned}$$

The final gradient descent vector $\tilde{\mathbf{d}}(a) = -(\frac{\partial}{\partial x}\tilde{E}, \frac{\partial}{\partial y}\tilde{E})$ is then computed by summation over the triangles $T(a)$ incident to a:

$$\tilde{\mathbf{d}}(a) = -(\tfrac{\partial}{\partial x}\tilde{E}, \tfrac{\partial}{\partial y}\tilde{E}) = -\textstyle\sum_{t\in T(a)}(\tfrac{\partial}{\partial x}\tilde{E}_t, \tfrac{\partial}{\partial y}\tilde{E}_t)$$

Appendix D: Per-Patch Parameterizations

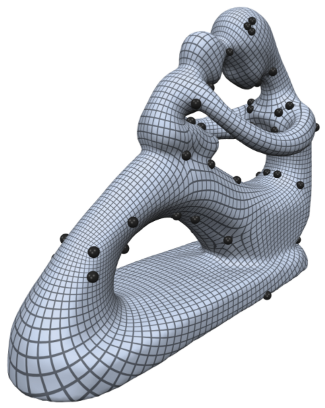
To extract a transition-free parameterization of an individual patch from the global parameterization $\mathcal{P}$, the parameterizations of the triangles T which are part of the patch region simply need to be expressed in a common chart. We start from an arbitrary triangle $seed \in T$ and from there conquer all others while transferring them to *seed*'s parameter system. Let f_{st} be the transition from face s to face t, Id the identity transformation, s.uv the set of (u, v) parameters of the corners of s, and Q a simple FIFO queue.

```
$Q$.push( [$seed$, Id] )
**while not** $Q$.empty
    $[t, f] \leftarrow Q$.pop()
    **for** $s \in T$ | adjacent to $t$ **and** not yet processed
        $g \leftarrow f \circ f_{st}$
        $s$.uv $\leftarrow g(s$.uv$)$
        $Q$.push( $[s, g]$ )
```

Note that if instead of individual patch parameterizations one *global* parameterization without transitions within the patches is desired (e.g. for visualization purposes), the transitions must necessarily be located at the patch borders. This is achieved by making the mesh conform with the layout by splitting the faces crossed during the arc tracing, effectively inserting edge strips that coincide with the arcs. Then performing the above procedure for one seed per patch directly results in all transitions getting shifted to the patch borders.

Appendix E: Comparison

In Section 9.1 zoom-ins on results of TPP are shown. Figure C shows the full models. In addition to the generally larger principal direction deviations of TPP demonstrated in Section 9.1, it can be summarized that nodes tend to get pulled away from extremal positions, e.g. at the feet, trunk, and ears of ELEPHANT, the wheels and nose of ELK, or the corners of BLOCK. Isolines of the parameterization tend to get straightened, leading to bad alignment at curved smooth features, e.g. at the cylindrical parts and holes of ROCKERARM and BLOCK. TPP uses the MIPS energy for parameterization. When using simpler barycentric mappings, with discrete harmonic coordinates as in [DBG*06] or with mean value coordinates as in [GVSS00, PSS01,KLS03], we observed these effects to be even stronger, as demonstrated here on the FERTILITY model.



Appendix F: Regular Node Optimization

Let us mention an option concerning the optimization of regular nodes in a layout. While irregular nodes (implying singularities) have an essential influence on the field

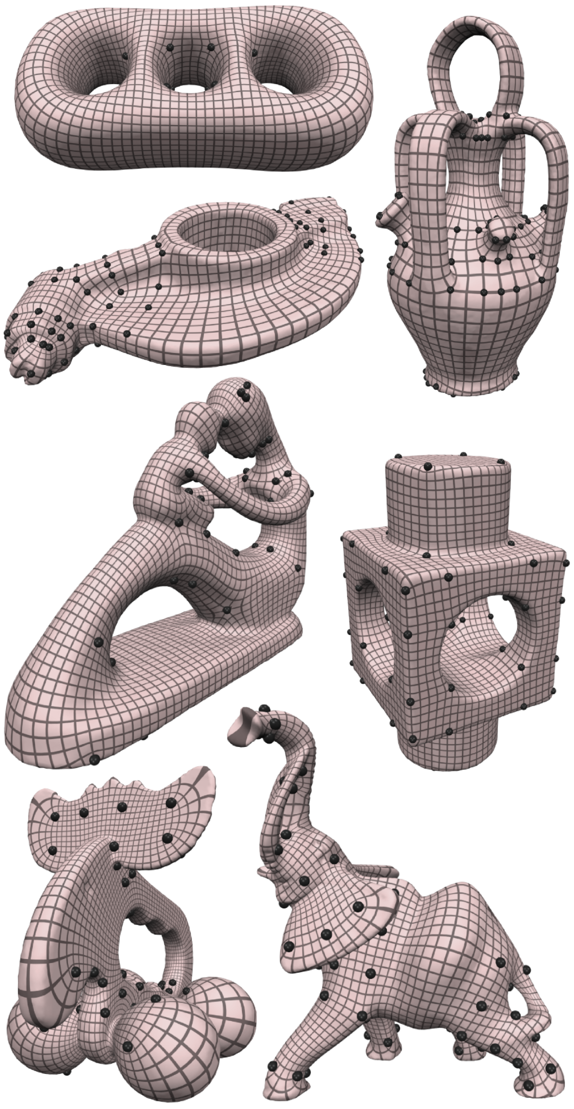
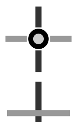


Figure C: *Results of TPP, shown in the same poses as the results of our method in Figure 6.*

construction and parameterization system structure, regular nodes do not. Using a simple modification, the optimization of the embedding of regular nodes can thus already be achieved during the parameterization step. We can simply remove a regular node's explicit occurrence in the node connection constraints by instead concatenating the four involved connection constraints in two pairs (unless this concatenates a constraint with itself), just as if the regular node was not present in the layout, but two arcs crossing in its place. The regular nodes' embedding is then optimized *implicitly* in the parameterization process: a node's new position can be found at the intersection of the two respective iso-parametric curves. This can be seen in the accompanying video: the regular node positions in the final layout lie at the crossings of arcs traced starting from the irregular nodes.



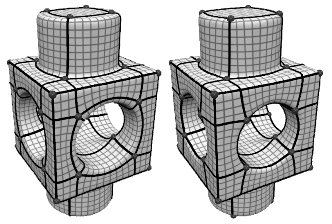
As the number of nodes to be moved in the gradient descent procedure does not significantly affect the total runtime, this option does not necessarily increase efficiency. It can, however, speed up convergence when the initial embedding of regular nodes is worse than that of irregular nodes – which can, for instance, be the case if it is a mere byproduct of the rough initial embedding of crossing arcs as in [CBK12].

It is important to notice that when this option is used the structure of the resulting layout is not guaranteed to be equivalent to the input. One can think of constellations where the arcs in the final parameterization could cross in another way than they do in the input (– except for layouts involving numerous dangling arcs, we have not come across such a situation). This can be seen as a form of implicit structure optimization, but can be undesirable if preservation of the structure is mandatory.

Appendix G: Directional vs. Positional Alignment

Our layout embedding optimization process generally aims for alignment of iso-parametric curves (hence also arcs and final patch parameterizations) to principal directions on the surface. Another form of alignment is that of *positional* alignment: one might want to fit an arc or a specific iso-curve of the parameterization to a specifically located curve on the surface. We described how to do this for the cases of fixing arcs and positionally aligning integer iso-curves to feature curves (Sections 6.2 and 8).

The optimization does, however, not explicitly aim at specifically positioning arcs onto non-sharp, smooth features or similar curvature extrema (unless fixed manually). While such alignment might be desirable from an aesthetic point of view, let us point out that it would often be suboptimal in terms of isometry and alignment. The inset demonstrates this on the BLOCK model which has numerous smooth feature curves:



On the left is the result of our optimization where some arcs align to smooth features, some do not – because this is the optimum in terms of isometry and principal direction alignment (as measured in a combined form by the parameterization energy functional). On the right we show a version where the nodes have manually been repositioned so as to achieve further arc-to-feature alignment, in particular around the holes. While this can be of interest for certain applications, in a general sense the left result is to be considered better: the parameterization residuum, reflecting isometric distortion and misalignment, is higher by a factor of 2.3 on the right. In particular, forcing an irregular node and two incident arcs to lie on a smooth curve implies patch corner angles of $\leq 60^\circ$ at valence 5 nodes and of 180° at valence 3 nodes – far from the general optimum of 90°.

Appendix H: Video

The accompanying video illustrates the embedding optimization process for several example layouts. For each example it shows:

- The initial layout embedding taken as input
- The parameterization with node connection constraints
- The gradient descent node relocation process
- The extraction of arcs
- The extraction of per-patch parameterizations (Section D)

For the gradient descent a step size factor of $\alpha = 0.1$ (instead of our default 0.75) has been used to slow down the process in order to achieve a fluent visualization – otherwise the nodes would jump to almost their final position in just a few steps. The video shows one step per frame.

During the process the global parameterization $\mathcal{P}$ (with visible transitions) is shown using an iso-line texture. The final continuous per-patch parameterizations after arc tracing are visualized using texture grids of size $m \times n$, where (m,n) is the rounded parametric extent of a patch.

The last example demonstrates how a willfully distorted layout still converges to the desired result, in spite of large dislocations and misoriented arcs (owing to the consistent labeling, cf. Section 5.1).

References

[EW05] ERICKSON J., WHITTLESEY K.: Greedy optimal homotopy and homology generators. In *Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms* (2005), pp. 1038–1046.

[WW94] WELCH W., WITKIN A. P.: Free-form shape design using triangulated surfaces. In *Proc. SIGGRAPH '94* (1994), pp. 247–256.