Abstract

Planar hexagonal-dominant (PHex) meshes are an important class of meshes with minimal vertex-degree. They are highly useful in the rationalization of freeform architectural surfaces, for construction with flat steel, glass, or wooden panels of equal thickness. A PHex mesh must contain both convex and concave faces of varying anisotropic shapes due to the planarity constraint. Therefore, while parameterization-based approaches have been successfully used for planar-quad meshing, applying such approaches to PHex meshing has not been attempted so far. In this paper, we show how to bridge this gap, effectively allowing us to leverage the flexibility of quad-remeshing methods (e.g. field alignment, singularities) to PHex remeshing. We have two main observations. First, the anisotropy can be handled by isotropically remeshing a modified geometry, which we denote by curvature shape, and then pulling back the result to the original surface. Second, various non-convex face shapes can be generated robustly by locally modifying the grid texture used for discretizing the parametrization. Together, these reductions yield a simple and effective method for PHex remeshing of triangular surfaces, which is additionally robust, and applicable to a variety of models. We compare our method to recent state-of-the-art methods for PHex meshing and demonstrate the advantages of our approach.

1 Introduction

Computational methods have become indispensable tools in architecture in recent years. By studying the properties of smooth and discrete surfaces, architectural geometry provides the mathematical machinery and the algorithms required for constructing structures with complex geometry. Polyhedral surfaces [Pottmann 2007], namely piecewise-linear surfaces with planar faces, form a subspace of discrete surfaces which are fundamental for architecture, due to their practical and aesthetic properties. In addition to face planarity, low vertex degree is a desirable property for architectural models, as these facilitate less complicated, thus cheaper, nodes connecting the faces [Pottmann et al. 2007a]. Hence, planar 3-web meshes, or meshes with degree 3 vertices and planar faces, are an important building block in the tool-set provided to architects.

From a practical standpoint, 3-webs have parallel face-offset meshes [Pottmann et al. 2007b], namely meshes with parallel edges which have constant face-to-face distance to the original surface. Such meshes are useful for the rationalization of freeform surfaces, as they can be constructed from parallel glass panels with supporting beams (Figure 1), or be covered with water-tight wooden panels of equal thickness (Figure 4).

In addition to their practical benefits, hexagonal meshes are sought after for their aesthetic properties and their prevalence in nature [Pearce 1990]. The most common hexagonal structure is a grid of convex regular hexagons (Figure 2(a)). However, it is well-established ([Pottmann 2007], Chapter 19) that it is not possible to approximate a surface which has regions of negative Gaussian curvature with hexagons that are all convex, planar, and which form a watertight surface. One possible solution is to remove the watertightness requirement (Figure 2(b)). However, such an approach might not be practical in many scenarios. It is therefore unavoidable to use non-convex faces to approximate a general surface with a planar hexagonal (PHex) mesh. Furthermore, planar faces must be compatible with the local curvature of the surface, and thus in most cases they are anisotropic (Figure 2 (c,d)).

In this work, we address the problem of approximating a given triangular mesh by a PHex mesh. Our method is based on reducing the problem to isotropic hexagonal remeshing of a modified geometry, allowing us to leverage the flexibility and robustness of existing remeshing machinery. To the best of our knowledge, our approach is the first that can handle a large variety of surfaces, allowing for some user control with a few intuitive parameters, and that can generate hexagonally-dominant planar meshes with singularities.

1.1 Related work

The problem of remeshing triangular meshes with other primitives, such as quads, hexagons, or arbitrarily shaped polygons, has received ample attention in recent years. In general, the problem we
edges with a conjugate curve network shown [Liu et al. 2011; Zadravec et al. 2010], that aligning the quad for the non-linear planarization step. More generally, it has been shown that a hex mesh is aligned with three, and it is not clear which direction of this approach is not trivial, as two main challenges arise. First, while a quad mesh is naturally aligned with two directions, a hex mesh is aligned with three, and it is not clear which directions should be conjugate in order for the resulting mesh to be close to planar. Second, assuming we found a conjugate curve network which provides a parameterization for the surface, it is not obvious how to discretize it. While there is a single tiling of the plane with quad faces and degree four vertices, there are various options for hexagonal tiles with degree three vertices (see Figure 3). We may be inclined to use regular hexagons, but as mentioned previously, it is not possible to tile a surface which has negative Gaussian curvature using only convex planar hexagons.

One approach for addressing these issues [Wang and Liu 2010], is to use conjugate directions to tile the surface with a quad mesh, and then extract from it a “brick” pattern, as in Fig. 3 (center). Post-processing this mesh for planarity tends to push the faces towards convex hexagons in elliptic regions, and concave hexagons in hyperbolic regions. It is not guaranteed however that the process would converge in complicated cases. Wang et al. [2008] suggest the construction of special triangulations that guarantee that the resulting dual hex mesh would have faces which do not self intersect, and that the faces converge in the limit to the Dupin indicatrix (see Section 2). Albeit theoretically sound, the creation of these triangular meshes requires an intricate algorithm, which is not robust where curvature directions are noisy or not well-defined (e.g. near umbilics and planar regions). Most importantly, none of these methods can generate PHex meshes with singularities, which are necessary for free-form modeling.

Non-planar hex meshes. The problem of constructing general, not necessarily planar, hexagonal meshes has been addressed in [Nieser et al. 2012], within a family of parameterization-based algorithms that also includes [Bommes et al. 2009]. Rotational-symmetry fields are constructed on triangular meshes and a parameterization, which is invariant to symmetric 6-fold rotations, is extracted. The mesh is then generated from the parameterization by tiling the parameter domain with regular convex hexagons. These methods can handle singularities and are robust. However, directly applying them to PHex meshing is challenging, as the resulting faces are not close enough to being planar and, therefore, do not provide a good starting point for the planarization process. This is also evident from the fact that the resulting faces are convex hexagons, whereas non-convex hexagons are necessary for watertight PHex meshing.

Our approach is based on reducing the problem to non-planar regular hex meshing on a modified geometry with a deformation of the generated hexagons, and thus uses such methods as building blocks.

2 Background

Before diving into the algorithmic details, we would like to provide some mathematical background and motivation for our approach. While parts of this material can be found in other places (e.g. [Strauik 1988]), we provide it here for completeness.

Given a smooth surface $M$ and a parameter $h$, we would like to construct a polygonal mesh $M_h = (V, E)$, such that its faces are planar, and the distance between $M$ and $M_h$ is on the order of $h$. In addition, we would like to have minimal degrees for the vertices of $M_h$, and thus require that the faces (except a few singular ones) are hexagons. Notice that it is not possible, except for genus 1 surfaces, to mesh a closed surface with pure hexagons such that all the vertex...
degrees are 3, and therefore non-hexagonal faces are required.

Consider the neighborhood of a point $p$ on $M$. The best planar approximation to this neighborhood is the tangent plane at $p$. However, if we pick a portion of the tangent plane as our face near $p$, then the vertices of the face would not lie on $M$. We can get a similar order of approximation, while keeping the vertices on the surface, by translating the tangent plane at $p$ in both the negative and the positive normal direction by $h/2$, and intersecting the result with the surface (see Fig. 5). The intersection between the shifted tangent plane and the surface is a set of planar curves. Our goal is to generate faces such that their vertices lie on (or close to) these intersection curves. This would yield faces which are (close to) planar, whose vertices lie on the surface. Therefore, our first goal is to understand the nature of these intersection curves.

### 2.1 The Dupin Indicatrix

A basic notion in continuous differential geometry, which is closely related to the intersection curves we are investigating, is that of the Dupin indicatrix. For every direction $V \in T_p M$ construct a geodesic from $p$ in the direction of $V$ whose length is $1/\sqrt{|k_p(V)|}$, where $k_p(V)$ is the normal curvature at $p$ in the direction $V$. The locus of the endpoints of these geodesics is a conic curve, called the Dupin Indicatrix ([Struik 1988]) (see Fig 5). If we express points in the tangent plane in the coordinate frame of the principal curvature directions $d_{\text{max}}, d_{\text{min}}$, then the Dupin indicatrix at $p = (x, y)$ is given by $D(p) = k_{\text{max}}x^2 + k_{\text{min}}y^2 = \pm 1$.

The normal curvatures are also an indication to the distance between the surface and the tangent plane at $p$. Intuitively, the bigger the normal curvature is at $p$ in the direction $V$, the faster the surface "pulls away" from the tangent plane in that direction, and the shorter the geodesic $1/\sqrt{|k_p(V)|}$ would be. This can be made precise by the following lemma, which describes the nature of the intersection curves we are investigating.

**Lemma 1.** ([Struik 1988], page 85): The intersection of the surface with a plane close to the tangent plane and parallel to it is (in a first approximation) related to the Dupin indicatrix by a similarity transformation.

This result has implications on the shapes of the faces we are generating. As we have seen previously, we would like to generate faces whose vertices lie on these intersection curves, as this would yield planar faces. Specifically, if we consider hexagonal faces, this Lemma implies that except at umbilic points, the hexagons cannot be both regular and planar. More generally, the hexagons would be convex in elliptic regions, rectangular in parabolic regions and concave in hyperbolic regions (see Fig. 6).

### 3 Algorithm outline

Lemma 1 provides us with a characterization of the faces we should generate if we wish them to be close to planar: a face near a point $p$ should approximate a curve which is related to the Dupin indicatrix at $p$ by similarity.

The next problem we need to address is how to sample the surface and connect edges, such that the induced hexagonal faces lie on such curves. For that, we take our inspiration from quad-remeshing algorithms. A standard way to generate new geometry is to construct a bijective map $\phi : M \rightarrow \Omega \subset \mathbb{R}^2$ from the surface to the plane, sample the plane with a grid, and then pullback the grid to the surface using the inverse map $\phi^{-1}$. This procedure generates both the vertices and the faces of the new quad mesh from the grid, as textured on the surface.

Therefore, our goal is to generate the map $\phi$ and sample the 2D plane with a hexagonal grid, such that the pulled-back hexagonal grid on the triangle faces would approximate the Dupin indicatrix of the surface at each triangle. For that, we need to conform to two properties of the indicatrix. First, the resulting hexagons need to be anisotropic, with the anisotropy induced by the Dupin indicatrix. Plainly put, the anisotropy of the map $\phi^{-1}(p)$ should be dictated by the normal curvatures at the point $p$. Second, as the map $\phi^{-1}$ is affine, the hexagonal grid in the parametrization domain must be modeled according to desired discrete face shape, and according to the proper curvature region: elliptic, parabolic, or hyperbolic. Additionally, since we are creating a watertight mesh, these different
discretizations should blend smoothly across curvature regions.

With these constraints in mind, our algorithm proceeds as follows (illustrated in Figure 7):

1. Analyze the curvature regions of the surface (a).
2. Construct an anisotropic Dupin-guided global parameterization $\phi$ (b-d).
3. Create a seamless convex hexagonal pattern on the surface by pulling back a regular hexagonal pattern on the plane through $\phi^{-1}$ (e).
4. Locally alter the hexagonal grid on the surface, such that faces approximate the shape of the local Dupin indicatrix (f).
5. Apply a standard post-processing planarization optimization to generate the final PHex mesh (g).

We start by providing guiding principles to our approach, and defer the implementation details to Section 6. We first consider the simplest case, a surface with positive Gaussian curvature everywhere.

### 4 Surfaces with positive Gaussian curvature

Let $\mathcal{M}$ be a surface with positive Gaussian curvature. In this case, as is evident in Figure 5, the Dupin indicatrix at each point is an ellipse, which is aligned with the principal curvature directions $d_{\text{min}}, d_{\text{max}}$, and whose radii are $1/\sqrt{k_{\text{min}}}, 1/\sqrt{k_{\text{max}}}$, corresponding to the respective principal direction. Locally, the map $\phi$ has to induce an anisotropy, such that a small circle in the parameter domain would be mapped to such an ellipse. Globally, the map $\phi$ has to fulfill some integer constraints, in order for a hex grid, pulled back from the parameter domain, to generate a seamless tiling.

While these constraints are seemingly difficult, we can take advantage of existing work for generating such anisotropic hexagonal tilings. First, note, that for a surface with positive Gaussian curvature, the second fundamental form $\mathbf{II}$ is positive definite, and therefore can be considered as a metric. Consider the shape operator $S_p$ in the tangent space $T_p$ of the point $p \in \mathcal{M}$, and its representation as a (positive definite) $2 \times 2$ matrix in local orthonormal coordinates. If we define a bijective map from $\mathcal{M}$ to an auxiliary surface $\hat{\mathcal{M}}$, such that the Jacobian of the map is $S^{1/2}$, then a small circle on $\mathcal{M}$ is mapped under the inverse map to an ellipse on $\hat{\mathcal{M}}$ whose axes are the eigenvectors of $S$, and whose radii are the eigenvalues of $S^{-1/2}$. Namely the axes are given by $d_{\text{min}}, d_{\text{max}}$, and the radii by $1/\sqrt{k_{\text{min}}}, 1/\sqrt{k_{\text{max}}}$. Therefore, a small circle on $\mathcal{M}$ is mapped to the Dupin indicatrix on $\mathcal{M}$. We denote $\mathcal{M}$ as the curvature shape of $\mathcal{M}$.

Using the curvature-shape construction, generating a PHex mesh that discretizes the Dupin indicatrix on a positively-curved surface is quite simple. We first construct an isotropic hexagonal mesh on the curvature shape, that discretizes local isotropic circles, using e.g. [Nieser et al. 2012], and then we pull back the locations of the vertices to the original surface $\mathcal{M}$. To construct the curvature shape we use the method proposed in [Panozzo et al. 2014] (where a similar approach was used for generating anisotropic quad meshes).

Figure 8 demonstrates this process for an ellipsoid shape. We first compute the curvature shape, then generate two different isotropic hexagonal parameterizations (a,b), which are both mapped to a Dupin remeshing of the original surface (c,d). In the convex case we enjoy a degree of freedom: we can discretize the circle on the curvature shape in any manner, and the resulting hexagon on the original shape would always discretize the Dupin indicatrix. If we choose a regular hexagon for this discretization (and a hexagonal grid for the global parameterization), we have a rotational degree of freedom inside the circle on the curvature shape.

### 5 General surfaces

We have shown that for convex surfaces we can reduce the problem of Dupin-guided parametrization to the isotropic parametrization of the curvature shape by a simple change of metric. In the general case, when the surface has regions with several types of Gaussian curvature, the simple approach described previously fails due to three reasons.

First, the shape operator is no longer guaranteed to be positive def-
The eigenvectors of \( \mathbf{\Pi} \) would be mapped to the Dupin indicatrix on \( \mathcal{M} \) discretization. Without loss of generality, we assume that conflict by design between the continuous Dupin indicatrix, and its parameterization step, we add an additional alignment on the curvature shape.

We therefore concede to try and define \( \mathbf{\Pi}_+ \) for that in the parameterization step, we add an additional alignment on the curvature shape.

We first solve the problem of defining a metric in order to create a curvature shape. That is, the curvature shape \( \mathcal{M} \) is such that the Jacobian of the map from the surface \( \mathcal{M} \) to \( \tilde{\mathcal{M}} \) is \( \mathbf{\Pi}_+^{1/2} \). As the elliptic case is clear, we discuss the issues stemming from the hyperbolic and parabolic cases.

**Hyperbolic Dupin:** the analogous condition in the hyperbolic case is to design \( \mathbf{\Pi}_+ \) such that a hyperbola on the curvature shape would be mapped to the Dupin indicatrix on \( \mathcal{M} \). Unfortunately, this is not possible to guarantee for any hyperbola, since, unlike a circle, a hyperbola is not symmetric under rotation. Therefore, uniform hyperbolae differing by rotation would be mapped under the same affine map to different hyperbolae on \( \mathcal{M} \).

We therefore concede to try and define \( \mathbf{\Pi}_+ \) such that a hyperbola on the curvature shape which is aligned with the eigenvectors of \( \mathbf{\Pi}_+ \) is mapped to the Dupin indicatrix. To account for that in the parameterization step, we add an additional alignment constraint, which we detail further on in this Section.

**Parabolic Dupin:** When considering parabolic shapes, there is a conflict by design between the continuous Dupin indicatrix, and its discretization. Without loss of generality, we assume that \( k_{\text{min}} = 0 \). Then, the Dupin indicatrix is infinite, whereas we discretize it with faces of finite proportions, mapped from the appropriate grid in the plane. Since a proper discretization of the Dupin indicatrix is defined up to similarity, any rectangular hex shape which is correctly aligned with the zero direction of \( \mathbf{\Pi} \) would serve. Therefore, we set the eigenvalue of \( \mathbf{\Pi}_+ \) in the minimal direction as 1 (isometric) instead of \( k_{\text{min}} = 0 \). However, in order to keep the resolution of the mesh uniform, parabolic regions with different nonzero curvature should have a proportionate discretization of the curved direction. Thus, to accommodate for the distance between the two parallel Dupin lines, the eigenvalue of \( \mathbf{\Pi}_+ \) in the curved direction should remain \( k_{\text{max}} \).

To summarize, our requirements from \( \mathbf{\Pi}_+ \) are as follows:

1. In elliptic regions, circles on the curvature shape should be mapped to the Dupin indicatrix, and \( \mathbf{\Pi}_+ = \mathbf{\Pi} \).
2. In hyperbolic regions, eigendirection-aligned hyperbolae on the curvature shape should be mapped to the (hyperbolic) Dupin indicatrices on the original shape.
3. In parabolic regions, the eigenvectors of \( \mathbf{\Pi}_+ \) should align with the Dupin indicatrix, and the eigenvalues should be as explained above.

Unfortunately, it is not possible to construct a continuous \( \mathbf{\Pi}_+ \) metric that conforms to these requirements exactly. In parabolic regions, the eigenvalue \( k_{\text{min}} = 0 \) is replaced by 1 without changing the approximation to the Dupin lines, but in near parabolic regions (hyperbolic or elliptic), the eigenvalue in the almost-zero direction remains almost zero, and we do not have continuity in the metric. We must therefore compromise the adherence of \( \mathbf{\Pi}_+ \) to the Dupin indicatrix in near parabolic regions to keep \( \mathbf{\Pi}_+ \) continuous.

We thus propose the following construction for \( \mathbf{\Pi}_+ \). The second fundamental form can be decomposed as \( \mathbf{\Pi} = USV^T \), where \( U \) contains the principal curvature directions \( d_{\text{min}}, d_{\text{max}} \), and \( S \) is a diagonal matrix that contains \( k_{\text{min}} \) and \( k_{\text{max}} \). We define the positive definite anisotropy metric \( \mathbf{\Pi}_+ = USV^T \), where \( S_+ \) is a diagonal matrix with entries \( f(k_{\text{min}}), f(k_{\text{max}}) \). Here, \( f \) is a smooth positive monotous function, such that \( f(0) = 1 \) and \( f(x) \approx x \) when \( x \gg 0 \). Thus, the third requirement holds by design, \( \mathbf{\Pi}_+ \) is continuous, and the anisotropies dictated by \( k_{\text{min}}, k_{\text{max}} \) to form the Dupin indicatrix are approximated reasonably away from parabolic regions.

When \( f(x) = x \), it is easy to see that the first requirement holds, since then \( \mathbf{\Pi}_+ = \mathbf{\Pi} \). To see that the second requirement holds, first note that \( \mathbf{\Pi} \) and \( \mathbf{\Pi}_+ \) share the same eigenvectors. Now, let \( d_{\text{min}}, d_{\text{max}} \) denote the pulled-back eigenvectors of \( \mathbf{\Pi}_+ \). Note that when a hyperbola on the curvature shape is aligned with \( d_{\text{min}}, d_{\text{max}} \), then this hyperbola is again aligned with \( d_{\text{min}}, d_{\text{max}} \) under the inverse mapping. Furthermore, if \( f(x) = x \), then the anisotropic scale the hyperbola would undergo is exactly \((1/\sqrt{k_{\text{min}}}, 1/\sqrt{k_{\text{max}}})\), mapping it to the Dupin indicatrix. Various choices for \( f \) would yield different curvature shapes, and would effect the way we switch between elliptic, parabolic and hyperbolic hexagons when \( f(x) \neq x \). We investigate the effect of \( f \) in the next Section.

Figure 9 shows a few examples of the resulting curvature shapes using this approach. While various constructions for \( \mathbf{\Pi}_+ \) are possible, we found this simple approach appropriate for our purposes.

### 5.2 Hex alignment

Having proposed a solution to the metric-embedding problem, we next face the problem of creating a proper isotropic parametrization on the curvature shape \( \mathcal{M} \). Recall that in the creation of \( \mathbf{\Pi}_+ \) we promise adherence to the Dupin indicatrix on \( \mathcal{M} \) in hyperbolic and parabolic regions only if the shapes on \( \mathcal{M} \) are aligned with...
to rotations by $\pi/3$ by a cross-field, no hexagonal pattern is symmetric with respect to rotations by $\pi/4$. Thus, $G_u$ and $G_v$ are distinct and non-interchangeable fields which are bound only by orthogonality. Conveniently, this is exactly the case with the principal curvature directions, which are similarly not interchangeable (as are their pull-back counterparts which we use on the curvature shape).

Non-interchangeable. Unlike quad meshing, which is governed by a cross-field, a hexagonal pattern is symmetric with respect to rotations by $\pi/4$. Thus, $G_u$ and $G_v$ are distinct and non-interchangeable fields which are bound only by orthogonality. Conveniently, this is exactly the case with the principal curvature directions, which are similarly not interchangeable (as are their pull-back counterparts which we use on the curvature shape).

Choosing the guiding direction. Our purpose is to match the pulled-back principal directions $d_u$, $d_v$ with $G_u$, $G_v$ in general. Since they are not interchangeable, one of the principal directions should be matched to one of the guiding fields throughout the mesh. Neither elliptic, nor hyperbolic regions have a specific choice, since they can both be discretized in several ways. However, in order to recreate the texture as in Figure 3(center), we need the $u$ direction to align with the curved direction of parabolic regions. This direction can be either the maximal curvature direction (if $d_{\text{max}} > 0$ and $d_{\text{min}} = 0$), or the minimal curvature direction (if $d_{\text{min}} < 0$ and $d_{\text{max}} = 0$). Making this choice is not always unique and coherent for all meshes, and we explain the nature of this conflict when we discuss limitations.

Alignment Weights. It is not equally important to align the parametrization gradients with the guiding fields $G_u$, $G_v$ everywhere. This requirement is especially important in parabolic regions, where misalignment with the Dupin lines would severely affect planarity. Figure 10 shows a cylindrical surface, and the hex remeshing result when the parameterization is aligned (left) and misaligned (right) with the alignment field. Note the bad planarity and face shapes results in the latter. We thus constrain the most curved portions of the mesh as hard constraints. For the Dupin-approximation quality explained before, it is also important to align to the principals in hyperbolic regions, and less so in elliptic regions.

Having obtained the guiding field $G_u$, $G_v$, and the alignment weights, we compute vector fields on the curvature shape $M$, and integrate them into a seamless isotropic parametrization of the curvature shape $M$.

5.3 Hex reshaping

Given a parametrization of the curvature shape, that we pull back to our original shape, we are still faced with the last problem: at every point, we aligned and stretched the parametrization such that when given the proper texture at that point (convex, concave or rectangular hexagons) we would get the proper Dupin discretization. However, it is not clear yet how to continuously vary between all three in a simple manner, and thus complete our pipeline.

We therefore devise a simple way to solve this problem: we first texture and generate the entire mesh with convex hexagons from a regular hexagonal grid. Next, we locally deform the vertices of the regular hexagons into the other desired shapes in the appropriate regions, such that they sample the Dupin indicatrix correctly.

In the practical sense, for each regular hexagon mapped unto a point the surface, we also mark the positions each textured vertex would take, had we used a different grid hex. In case we see fit to use either grid according to the local curvature of the point, we simply deform these vertices into the proper position. Thus, we create a proper blend of generated textures with ease.

To conclude, if the parameterization is aligned with the curvature directions, then simply locally switching between the textures shown in Figure 3 and using the anisotropy metric discussed previously, would yield hexagonal faces which lie on the Dupin indicatrix, and are therefore close to planar. Figure 11 shows the smooth blend between hexagon shapes we achieve using this approach.

6 Implementation Details

In this section we discuss the algorithmic details required for reproducing our results.
Anisotropic metric. The anisotropic scale induced by the metric \( \mathbf{H}_a \) is given by \( 1/\sqrt{f(k_{\text{min}})} \) and \( 1/\sqrt{f(k_{\text{max}})} \), where we require a function \( f \) which can handle the degeneration of \( \mathbf{H} \) in parabolic regions, where either \( k_{\text{min}} \) or \( k_{\text{max}} \) is 0.

Note that in parabolic regions the scale of the hexagon can be arbitrary. However, a scale too coarse in the curved direction would generate a bad approximation of the original surface. Furthermore, as mentioned in the previous Section, the chosen scale in the parabolic regions should be smoothly interpolated to the neighboring hyperbolic, yet nearly parabolic, regions, where we need to trade-off exact Dupin anisotropy for smoothness. Therefore, we take \( f(x) = 1 + ax/\kappa \), where \( a \) is a global constant which controls this trade-off, and \( \kappa \) is the maximal absolute curvature of the mesh. This approach effectively shifts and scales the values of \( x \) from \([0, \kappa]\) to \([1, 1 + a]\). The larger \( a \) is, the faster the anisotropy becomes closer to its correct Dupin based values when moving away from parabolic regions to hyperbolic and elliptic regions. The smaller \( a \) is, the more isotropic the shape of the hexagon is. Note, that to avoid dependence on the global scale of the mesh, we have that for \( x \gg 0 \), \( f(x) \approx ax/\kappa \).

Figure 12 shows the effect of the parameter \( a \) and the resulting scaling function \( f \) on the results. In the top row, \( a = 0.1 \), and therefore the metric is almost isotropic across the mesh. Note that in this case the curvature shape (shown in the left column) is almost identical to the source shape. Accordingly, the hexagons are isotropic, with almost square hexagons in parabolic regions. While this is favorable from an aesthetic standpoint, it leads to a large distance between the hexagonal mesh and the original mesh after planarization (note the flat region on the upper part of the torus). In the middle row, \( a = 5 \), and therefore the anisotropy far from parabolic regions is close to its correct value. Note, that the curvature shape scales in one direction to accommodate for this anisotropy. The resulting hexagons are therefore more stretched, yet the resulting mesh is a better approximation of the source shape. When we take \( a = 50 \) (bottom row), the anisotropy is closest to its correct value. However, the hexagons in parabolic regions are very stretched, and therefore less aesthetic. The average Hausdorff distance, given as percentage of the bounding box diagonal, is: 0.000481, 0.000337 and 0.000265 for \( a = 0.1, 5 \) and 50, respectively. Unless otherwise stated, we used \( a = 5 \) in all our examples.

Choosing the guiding direction. As explained in the previous Section, we wish to choose \( G_w \) to align globally with the single principal direction which is the curved direction in the parabolic regions of the mesh. However, this choice is not necessarily unique for general meshes; in some meshes \( d_{\text{max}} \) is the curved direction for some parabolic regions, and \( d_{\text{min}} \) is the curved direction in other parabolic regions. This is not a problem encountered in quad remeshing, where \( \pi/4 \) singularities are allowed. In practice, we wish to correctly align the most dominant regions to minimize this problem. We give a weight for each parabolic triangle that is equal to its absolute mean curvature. The group of parabolic triangles that belong to one class (e.g., \( d_{\text{max}} \) is their curved direction), and has the greatest sum of absolute mean curvature among them, chooses \( G_w \) for the mesh globally, as their curved direction. The “losing” triangles would necessarily be misaligned. In the absence of parabolic regions there is no preference for \( d_{\text{min}} \) or \( d_{\text{max}} \).

Alignment. We seek to find a smooth 6-RoSy field, representing the symmetries of a convex hexagon, on the curvature shape, such that one of the directions in this 6-RoSy field aligns with the guiding field \( G_w \). This is done as in [Knöppel et al. 2013], by minimizing the energy \( E = E_{\text{smoothness}} + \lambda E_{\text{alignment}} \). The alignment factor \( \lambda \) is given per triangle according to the curvature classification of the triangle: in parabolic regions we use \( 4\lambda \), in hyperbolic regions \( 2\lambda \), and in elliptic regions \( \lambda \), where \( \lambda \) is a global parameter provided by the user (we used \( \lambda = 0.1 \)). Furthermore, in the most highly curved regions of the surface, we treat the alignment constraints as hard constraints by specifying very high weights.

The parametrization then follows as in [Nieser 2012]: The mesh is cut, and a single field is chosen from the 6-Rosy to be the candidate gradient of the parametrization function \( u \), the orthogonal to this field is the candidate gradient to \( v \). Then, the parametrization is computed by solving a mixed-integer Poisson system to retrieve the scalar values for every vertex.

Hex-Mesh Generation. Generating a hexagonal mesh from a parametrization is a more difficult task than the quad mesh generation, as there are no simple integer lines to trace throughout
the mesh. Instead, we create the regular hexagonal grid as an arrangement on the plane, and overlay each triangle with its associated parameterization 2D coordinates upon it, to impress the triangle with the proper texture. This overlay is done using the arrangement package implemented in [cga]. Next, we lift the texture into 3D by barycentric interpolation in each triangle to receive a triangle mesh refined with hexagons within each face. Finally, we remove the triangle vertices and edges and obtain a hexagonally-dominant mesh.

**Reshaping.** Our hex reshaping is based on smoothly alternating between multiple hexagonal grids (as in Figure 11(a)). Implementation-wise, however, making sure that different hex textures are seamless is not trivial. Therefore, we first use a regular convex texture, which additionally includes hints on the potential alternative locations of each vertex. Then, we generate a hex mesh with convex faces, including these additional geometry lines from every vertex to its alternative locations on the surface. In the actual reshaping stage, we choose how and whether to move each vertex to its alternative locations according to the following algorithm.

1. Each vertex has three possible lines of motion. Pick the direction which is best aligned with the guiding field $G_u$.
2. In that direction, we have three possible target vertex positions, for a convex, concave and rectangular hex.
3. To decide which vertex to pick, we consult the curvature classifications of the original mesh faces, and vote between the three. If they are all different, we pick parabolic.

Figure 13 demonstrates this process. We first generate the texture hints in three directions for each vertex (center), where elliptic, parabolic and hyperbolic grid points correspond to red, green and blue points, respectively. Then, we pick one of the three directions according to the alignment field on the original triangulation, and then pick the final target location of a vertex depending on the curvature regions (left). The final result is shown on the right.

**Planarization post processing.** The generated faces lie close to the Dupin indicatrix, and therefore, they are approximately planar. However, they still benefit from planarity optimization of the following type: for each consecutive four vertices in the hex, we use quad planarization, as explained in [Liu et al. 2006]. The planarity of the hexagons in all our examples are measured by the average planarity of all quads within, where the planarity of any quad is measured as the ratio of distance between the diagonals to average diagonal length in percentage. The accepted upper limit is typically 1%.

**Time complexity.** Our method is quite efficient, and the parameterization and hex-generation step, including hex reshaping, took under three minutes for all the examples shown, in which the hex generation step is the heaviest, owing to the need to use exact rational numbers in CGAL arrangements. The post-processing planarization step is the most time consuming (on the order of 5-7 minutes). However, our planarization is using a non-optimized MATLAB implementation which is not part of our core algorithm. Furthermore, this is a standard post-processing step which is required for most planar-remeshing method.

**Limitations.** Our method has a few limitations. First, it is mostly applicable to architectural meshes, where the curvature behavior is mostly smooth, and there are large homogeneous curvature regions. For other types of meshes, the curvature might be too noisy, and our approach would not generate aesthetic structures. Note, however, that as long as the curvature regions are mostly smooth, our method still provides close to regular PHex meshes.

Second, as explained previously, we need to choose as an alignment direction either the $d_{min}$ or the $d_{max}$ directions, and we pick according to the parabolic regions which are most curved. However, there might be meshes where such a choice is not easy, and we have to align some of the parabolic regions in the wrong direction. Avoiding such cases is non-trivial without allowing for $\pi/4$ singularities in the alignment field, which are again not compatible with a hexagonal mesh. Figure 14 shows such an example. Note the large planarization errors for the interior cylinder, which is due to the misalignment of the tiles to the curvature direction.

![Figure 14: In some cases it is not possible to choose an alignment filed such that all the cylinders are properly aligned. In this case, the misaligned cylinders generate a large planarization error.](image)

7 Experimental Results

**Comparison to [Zimmer et al. 2013]** We compare our approach to [Zimmer et al. 2013], which finds proper tangent planes for a given triangulation with a variational approach. Figure 15 demonstrates the result. Note, that while our result (left) exhibits mostly symmetric and regular faces, the approach by [2013] (right) generates meshes which are less structured. In addition, our mesh is a better approximation of the source mesh, with average Hausdorff distance of 0.001900, whereas the other approach achieves 0.002414 (as percentage of the bounding box diagonal).

![Figure 15: Comparison of our method to [Zimmer et al. 2013]. Our method yields a more regular structure with symmetric hexagons.](image)

**Comparison to [Wang et al. 2008]** We have additionally compared our approach to [Wang et al. 2008], using a planar hex mesh
provided by the authors. The results are shown in Figure 16. While both results are mostly regular and planar, our method is more versatile, as it can generate hexagonal meshes of varying resolution, with varying alignment and with singularities for a wide range of inputs.

**Figure 16:** Comparisons with our method (left) with [Wang et al. 2008] (right). Notice that as a parametrization method, we can adhere to the original boundary of the mesh.

Figure 17 demonstrates the planarization results with (top) and without (bottom) face reshaping. Note that when no face reshaping is done, the shapes of the hexagons are less regular, and the resulting surface is farther from the source mesh.

**Figure 17:** Comparing our final result (top) to planarization without face reshaping (bottom). Note, that the latter is less smooth and less regular.

Figures 18 and 19 demonstrate the versatility of our approach. The first demonstrates applicability to typical meshes used in Geometry Processing, and the second shows that we can easily remesh with varying resolutions, by modifying the grid size. The former has not been shown by any previous method.

Figure 20 shows the hexagonal remeshing results for additional models, and Table 1 provides the statistics.

### 8 Discussion

We have presented a simple yet effective and versatile approach for generating planar hex-dominant meshes from triangular meshes. Our method is based on leveraging well-established remeshing tools by reducing the difficult anisotropic non-homogeneous remeshing problem to simple isotropic remeshing on an alternative geometry, the curvature shape, which encapsulates all the required information about the second fundamental form of the original surface. We demonstrated the applicability of our method to various architectural models, and discussed the advantages compared with state-of-the-art.

We expect that using a modified metric would also help in other aspects of architectural geometry, e.g. in mesh deformation and form-finding. In the future we plan to investigate further aspects of this approach. For example, it is possible that it is not necessary to actually construct the surface with the modified metric, but it suffices to work with an abstract manifold whose metric is defined explicitly and not provided by a Euclidean embedding. For architectural purposes, we would like to extend out insights about planarity approximation for the design of more general surface planar tilings from the wallpaper group. Finally, we wish to explore the possibility to weigh and balance the relaxation of the 3-web requirement, or the watertight requirement, in order to control and achieve more regular face shapes automatically.

**Table 1:** Statistics. From left to right: model, number of faces in original model, number of faces in remeshed model, number of non-hexagonal faces, time in seconds: curvature shape creation, parameterization, mesh generation.

<table>
<thead>
<tr>
<th>Model (Fig #)</th>
<th># Tri</th>
<th># Hex</th>
<th># Sing</th>
<th>$t_{c,s}$</th>
<th>$t_p$</th>
<th>$t_{m,g}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roof (1)</td>
<td>4288</td>
<td>1825</td>
<td>4</td>
<td>6.27</td>
<td>4.16</td>
<td>10.34</td>
</tr>
<tr>
<td>Soumaya (2)</td>
<td>17718</td>
<td>3518</td>
<td>18</td>
<td>33.27</td>
<td>25.62</td>
<td>58.89</td>
</tr>
<tr>
<td>Metro (4)</td>
<td>3559</td>
<td>2455</td>
<td>6</td>
<td>5.89</td>
<td>3.26</td>
<td>9.15</td>
</tr>
<tr>
<td>Six (7)</td>
<td>2744</td>
<td>1357</td>
<td>9</td>
<td>4.83</td>
<td>3.52</td>
<td>8.35</td>
</tr>
<tr>
<td>T-Bar (10)</td>
<td>9728</td>
<td>2089</td>
<td>7</td>
<td>16.33</td>
<td>20.94</td>
<td>37.27</td>
</tr>
<tr>
<td>Vase (11)</td>
<td>2404</td>
<td>659</td>
<td>0</td>
<td>4.82</td>
<td>1.26</td>
<td>6.08</td>
</tr>
<tr>
<td>Interior (15)</td>
<td>3136</td>
<td>2286</td>
<td>7</td>
<td>5.38</td>
<td>2.39</td>
<td>7.77</td>
</tr>
<tr>
<td>Shoe (16)</td>
<td>9123</td>
<td>1462</td>
<td>2</td>
<td>16.43</td>
<td>6.13</td>
<td>22.56</td>
</tr>
<tr>
<td>Gitter (17)</td>
<td>6610</td>
<td>2292</td>
<td>2</td>
<td>11.07</td>
<td>3.80</td>
<td>14.87</td>
</tr>
<tr>
<td>Moomoos (18)</td>
<td>7776</td>
<td>4441</td>
<td>57</td>
<td>13.79</td>
<td>16.23</td>
<td>30.02</td>
</tr>
<tr>
<td>Stadium (19)</td>
<td>7892</td>
<td>1713</td>
<td>4</td>
<td>14.1</td>
<td>6.06</td>
<td>20.17</td>
</tr>
<tr>
<td>Schwarz (20)</td>
<td>6144</td>
<td>3821</td>
<td>24</td>
<td>10.64</td>
<td>7.21</td>
<td>17.85</td>
</tr>
<tr>
<td>Lilium (20)</td>
<td>6690</td>
<td>1201</td>
<td>28</td>
<td>12.02</td>
<td>26.69</td>
<td>38.71</td>
</tr>
<tr>
<td>Half tunnel (20)</td>
<td>5592</td>
<td>1333</td>
<td>13</td>
<td>10.07</td>
<td>3.61</td>
<td>13.70</td>
</tr>
<tr>
<td>Double Wiggly (20)</td>
<td>22464</td>
<td>3634</td>
<td>16</td>
<td>41.05</td>
<td>20.62</td>
<td>61.67</td>
</tr>
</tbody>
</table>

**References**

Figure 19: Different resolutions by modifying the grid size. Hausdorff distances from source are: 0.002927, 0.002953 and 0.003000.

Figure 20: Additional hexagonal meshing results.


