Field-Aligned and Lattice-Guided Tetrahedral Meshing

Saifeng Ni\textsuperscript{1}, Zichun Zhong\textsuperscript{2}, Jin Huang\textsuperscript{3}, Wenping Wang\textsuperscript{4}, and Xiaohu Guo\textsuperscript{1}\textsuperscript{†}
\textsuperscript{1} The University of Texas at Dallas
\textsuperscript{2} Wayne State University
\textsuperscript{3} State Key Lab of CAD&CG, Zhejiang University
\textsuperscript{4} The University of Hong Kong

Abstract
We present a particle-based approach to generate field-aligned tetrahedral meshes, guided by cubic lattices, including BCC and FCC lattices. Given a volumetric domain with an input frame field and a user-specified edge length for the cubic lattice, we optimize a set of particles to form the desired lattice pattern. A Gaussian Hole Kernel associated with each particle is constructed. Minimizing the sum of kernels of all particles encourages the particles to form a desired layout, e.g., field-aligned BCC and FCC. The resulting set of particles can be connected to yield a high quality field-aligned tetrahedral mesh. As demonstrated by experiments and comparisons, the field-aligned and lattice-guided approach can produce higher quality isotropic and anisotropic tetrahedral meshes than state-of-the-art meshing methods.

1. Introduction
Field-aligned quadrilateral and hexahedral meshing are active research topics in recent years [PPTSH14, SRUL16, GTJP17]. For quadrilateral and hexahedral meshes, field alignment is very natural because the edges of those meshes are expected to agree with the vectors defining the underlying frame fields. Field-alignment includes the alignments of both Riemannian distances and directions. In triangular meshing, there has been researches focusing on Riemannian metric (i.e. distance between nodes) control [ZGW\textsuperscript{*13}, FLSG14, NPPZ12], direction control [JTPSH15, DLY\textsuperscript{*18}] or even both [HZP\textsuperscript{*11}]. However, in tetrahedral meshing, only the Riemannian metric has been considered [LS03, DW05a, FLSG14, BSTY15]. So far we have not found any tetrahedral meshing work that takes control of both Riemannian metric and direction into consideration.

The triangle and the tetrahedron are the simplest elements in 2D and 3D, respectively. The dihedral angle of a regular tetrahedron is 70.53\textdegree. Unlike tiling regular triangles for 2D Euclidean space, it is impossible to tile regular tetrahedra for 3D Euclidean space. For most of the existing variational tetrahedral meshing algorithms, e.g., either Centroidal Voronoi Tessellation (CVT) based [DW03, ACSVD05, LWL\textsuperscript{*09}], or Optimal Delaunay Triangulation (ODT) based methods [CH11, CWL\textsuperscript{*14}], the majority of their outputs are close to regular tetrahedra, accompanied by some badly shaped tetrahedra. This is one of the reasons that slivers are notoriously hard to remove in tetrahedral meshing [KS07, TSA09], and also one of the reasons that direction-aligned tetrahedral meshing has not been discussed.

Our motivation is to generate tetrahedral meshes with high quality elements, instead of regular tetrahedra, which can pack the 3D Euclidean space. Body-Centered-Cubic (BCC) and Face-Centered-Cubic (FCC) lattices are two close packing scheme of spheres in 3D. The corresponding tetrahedra formed by BCC and FCC lattices have high quality [DW05b], which has been confirmed and used in mesh generation [LS07] and applications [ATW13]. Besides that, the symmetric cubic structures of BCC and FCC also allow us to build field-aligned anisotropic tetrahedral meshes.

In this paper, we propose a particle-based variational method to generate field-aligned cubic lattice, which leads to anisotropic tetrahedral meshes. We design a Gaussian Hole Kernel to construct a potential energy of the particle system, and optimize the energy to effectively and efficiently distribute the particles for the desired lattice patterns. To the best of our knowledge, our particle-based cubic lattice (BCC and FCC) optimization method is the first approach that can generate field-aligned isotropic and anisotropic tetrahedral meshes. As illustrated by our experiments, the field-aligned and lattice-guided tetrahedral meshing provides two benefits: (1) for isotropic tetrahedral meshing, having a direction field to guide the mesh could potentially improve mesh quality, especially for models with rotational features; (2) for anisotropic tetrahedral meshing, having BCC/FCC to guide the mesh can generate higher quality meshes as compared to state-of-the-art methods.
A Body-Centered-Cubic (BCC) lattice is formed by vertices of cubic cells along with cell centers as shown in Fig. 1a. Their Voronoi cells are truncated octahedra, and each dual Delaunay tetrahedral element has dihedral angles $[60°(4), 90°(2)]$, called bic cells along with cell centers as shown in Fig. 1a. Their Voronoi A Body-Centered-Cubic (BCC) lattice is formed by vertices of cubic cells. Every point in the BCC lattice has one-ring neighbor structures, consisting of 8 nearest neighbors and 6 second-nearest neighbors. For a regular cubic lattice, FCC lattice has identical one-ring neighbor structures, denoted as: $\{\pm 1, 0, 0\}, \{0, \pm 1, 0\}, \{0, 0, \pm 1\}$. Every point in the BCC lattice has the same one-ring neighbor structures, denoted as: $\{\pm 0.5, \pm 0.5, \pm 0.5\} \cup \{0, \pm 1, 0\} \cup \{0, 0, \pm 1\}$. (1)

A Face-Centered-Cubic (FCC) lattice consists of vertices of cubic cells and their face centers as shown in Fig. 1b. Their Voronoi cells are rhombic dodecahedra, and the dual Delaunay tetrahedral elements include two kinds of tetrahedra with dihedral angles $[54.735°(4), 90°, 109.47°]$ and $[70.528°(6)]$. Comparing to BCC, FCC is preferred as a finite-element mesh generation in terms of the mean squared error [BS83]. Every point in the BCC lattice is associated with a stretched/sheared cube with edges $t_1, t_2, t_3$. The matrix $B_i$ locally maps a stretched/sheared cube with edges $t_1, t_2, t_3$ to a regular cube aligned with the three axes of the 3D Cartesian coordinate system. $B_i$ also defines a local mapping of any vector from the anisotropic space into isotropic space. The details are discussed in Sec. 3.1.

### 2.3. Related Works on Tetrahedral Meshing

Tetrahedral meshing has been studied for several decades in both engineering and computer science fields. The algorithms can be categorized into four types: advancing front methods, octree and lattice methods, Delaunay-based methods, and particle-based methods.

Advancing Front Methods [MH95] start from the domain boundary and gradually add vertices and tetrahedra until the domain is completely meshed. They preserve the domain boundary explicitly. However, the difficulty of this type of method is to resolve the intersected tetrahedra inside the domain.

Octree and Lattice Methods: Quadtree/Octree is a Cartesian grid structure in 2D/3D. Quadtree encoding an input curve in 2D was first introduced by Yerry and Shephard [YS83], then generalized to 3D [YS84] and refined by Shephard and George [SG91]. These methods start from encoding the surface as an adaptive grid structure, i.e., an octree, which converts the surface to a volumetric representation. The tetrahedral meshes are then constructed by local tetrahedralization of each octree cell with special treatment for the cells intersecting the input surface [SG91, MV92].

Similar to octree methods, lattice methods utilize a space-filling tetrahedral lattice instead of Cartesian grid as volume representation, which omits the local tetrahedralization for the interior cells. The boundary tetrahedra are deformed to preserve smooth boundary. For a better quality, finer cells are generated by Fredenthal subdivision of a grid [VdMG97] or adaptive FCC lattice [MBF03] along the boundary surface. The isosurface stuffing method [LS07] proposed an option to cut the boundary tetrahedra to resolve the input surface. The cutting rules yield theoretical bounds for the smallest/largest dihedral angles. Doran et al. [DCB13] extended the method with A15 lattice. However, these stuffing-based methods cannot generate either field-aligned isotropic tetrahedral meshes, or anisotropic tetrahedral meshes.

Delaunay-Based Methods [CDS12] can be further categorized into two groups: (1) Delaunay refinement-based methods [Che97, JAYB15, Si15] improve the mesh quality by inserting new vertices until certain user-specified conditions are met, e.g., the minimal dihedral angle. (2) Variational methods iteratively minimize an energy, e.g., CVT [DW03, ACSYD05, LWL*09], ODT [CH11, CWL*14], by optimizing positions of vertices and their connectivities.

Particle-Based Methods use repulsive particles to resample surfaces or volumes. It was first introduced by Turk [Tur92], and later extended by Witkin and Heckbert [WH05] for implicit surface meshing. They introduced Gaussian kernel to model the interaction between particles which sample an implicit surface. Researchers have tried different choices of kernels, such as a modified cotangent function with finite support [MGW05], or a bounded...

![Figure 1: BCC and FCC lattice](image-url)
cubic function [YS00], and packing ellipsoidal bubbles instead of spherical bubbles to get anisotropic tetrahedral meshes. Zhong et al. [ZGW*13] used the Gaussian kernel to model the inter-particle energy in an embedding space to solve anisotropic surface meshing. Note that the traditional Gaussian kernel is radially-symmetric. Even though it can be distorted to elliptically-symmetric under Riemannian metric, the interaction between particles still resembles packing of circles/spheres isotropically, or ellipses/ellipsoids anisotropically. This makes it impossible to explicitly control field-valuation of particles. In this paper, we propose Gaussian Hole Kernel as potential energy between particles to guide their distribution into either BCC or FCC pattern, which will be introduced in the next section.

3. Particle-Based Lattice Optimization

In the particle-based framework, each vertex in the resulting mesh is modeled as a particle with certain inter-particle potential energy, the derivative of which determines the inter-particle forces. The position of particles are optimized according to the forces from their neighbors until they reach the equilibrium. In the following subsection, we introduce how to design the potential energy which can guide particles to form the field-aligned lattice pattern, either BCC or FCC, when they reach the equilibrium.

3.1. Gaussian Hole Kernel

Gaussian kernel is radially-symmetric, thus defining the inter-particle energy using it resembles packing of circles/spheres in 2D/3D, as demonstrated for anisotropic triangular meshing of surfaces [ZGW*13]. Suppose two neighboring particles $i$ and $j$ are located at $p_i$ and $p_j$, respectively, their radially-symmetric energy can be defined as:

$$E_{ij} = e^{-\frac{|v_{ij}|^2}{2\sigma^2}},$$

where $v_{ij} = p_i - p_j$, and $\sigma$ is the standard deviation of the Gaussian kernel. However, such radial-symmetry means that this potential energy does not have directional alignment property. In other words, given two different cross fields (of rotation only), their particle optimization results will be the same.

We need to construct a more specific potential energy to get the desired lattice structure. Once the frame field specifying the target edge direction and length of the lattice cube are given, the one-ring neighbors of a particle are fixed accordingly. Radial-symmetry is not enough to form the particular one-ring structures locally. Our goal is to force neighbor particles to fall into each others’ desired one-ring neighbor positions exactly by minimizing the potential energy. To achieve such property, we place negative Gaussian kernels right at the desired one-ring neighbor positions, which is like digging a hole at those positions in the energy field:

$$E_{ij} = e^{-\frac{|v_{ij} - \text{Onering}(k)|^2}{2\sigma^2}},$$

where $\text{Onering}(k)$ is the desired offset from $p_j$ to its k-th one-ring neighbor $p_i$ for either BCC in Eq. (1) or FCC in Eq. (2), $k = 1, \ldots, N_{\text{or}}$. When we minimize such a potential energy, the neighboring particles will be pushed exactly to those holes. Besides, that we also include a positive Gaussian kernel at the position of the particle itself which will push its neighbors away to avoid particles being optimized to the same positions. We call this potential energy as Gaussian Hole Kernel (GHK):

$$E_{ij} = e^{-\frac{|v_{ij}|^2}{2\sigma^2}} - \frac{1}{N_{\text{or}}} \sum_{k=1}^{N_{\text{or}}} e^{-\frac{|v_{ij} - \text{Onering}(k)|^2}{2\sigma^2}}. \tag{3}$$

To generate anisotropic field-aligned lattice pattern, we will transform the anisotropic lattice alignment problem to an isotropic one locally based on the given frame field. When particles form a regular cubic lattice aligned with the axes of Cartesian coordinate system, particles in the anisotropic space will exhibit a BCC/FCC pattern aligned with the desired frame field. Each particle $i$ is associated with a matrix $T_i$ expanded by three vectors $\{t_1, t_2, t_3\}$. Those three vectors define the local alignment of cubic lattice. Suppose there is no degenerate case, i.e., $|T_i| \neq 0$, then the corresponding matrix $B_i = T_i^{-1}$ transforms the anisotropic space to an isotropic one locally: $B_i T_i B_i^T \{t_1, t_2, t_3\} = I$. In other words, $B_i$ transforms an anisotropic lattice to an isotropic one locally. If we take $\{t_1, t_2, t_3\}$ as basis of the anisotropic space, then any vector $v = k_1 t_1 + k_2 t_2 + k_3 t_3$ in the anisotropic space has a corresponding vector $v' = B_i v = \text{diag}(k_1, k_2, k_3) v$ in the mapped isotropic space, where the one-ring neighbors of each vertex in BCC and FCC are well defined as in Eq. (1) and Eq. (2).

Suppose there are $N$ particles $V = \{p_i | i = 1 \ldots N\}$. For two neighboring particles $i$ and $j$, we use $T_{ij} = T_i^T \frac{T_j}{2}$ as the frame evaluated ($T'$ at the field sample closest to $E_{ij}$) at the middle of two particles, and the correspondingly matrix $B_{ij} = T_j^{-1}$ transforms $v_{ij}$ from its anisotropic space to the isotropic one. The energy of Eq. (3) between two neighboring particles $i$ and $j$ can be modified as:

$$E_{ij}' = e^{-\frac{|v_{ij}|^2}{2\sigma^2}} - \frac{1}{N_{\text{or}}} \sum_{k=1}^{N_{\text{or}}} e^{-\frac{|v_{ij} - \text{Onering}(k)|^2}{2\sigma^2}}. \tag{4}$$

Here $\sigma$ should be proportional to the expected edge length $\ell^*$ of cubic lattice. We discuss the choice of a proper value for $\sigma$ in Sec. 4.

Note that we denote all the symbols in the isotropic space with a prime symbol ('). The energy $E_{ij}'$ is defined in the isotropic space. The negative of first-order derivative of $E_{ij}'$ with respect to $p_i'$ is the force defined in the isotropic space: $F_{ij}' = -\frac{\partial E_{ij}'}{\partial p_i'}$. Since the particle positions $p_i'$ are optimized in the isotropic space, we transform the force back to the anisotropic space: $F_{ij} = T_j T_i^{-1} F_{ij}'$, which is:

$$F_{ij} = \frac{v_{ij}}{\sigma^2} e^{-\frac{|v_{ij}|^2}{2\sigma^2}} - \frac{T_j T_i}{N_{\text{or}}} \sum_{k=1}^{N_{\text{or}}} B_{ij} v_{ij} - \text{Onering}(k) \frac{e^{-\frac{|v_{ij} - \text{Onering}(k)|^2}{2\sigma^2}}}{\sigma^2}.$$ \tag{5}

Our energy definition in Eq. (4) satisfies $E_{ij}' = E_{ij}'$, and the force defined in Eq. (5) satisfies $F_{ij} = -F_{ij}'$.

3.2. Lattice Optimization

Once the inter-particle energy is defined, the particle optimization problem is modeled as an energy minimization problem. The variables are the particle positions $V = \{p_i | i = 1 \ldots N\}$, which are constrained in domain $\Omega$. The problem is formulated as follows:

$$\min_{\Omega, V} E(V) = \sum_{i,j} E_{ij} \approx \sum_{i} \sum_{j \in \mathcal{N}(i)} E_{ij} \tag{6}$$

s.t. $p_i \in \Omega$, $\forall i = 1 \ldots N$.
where \( \mathbb{N}(i) \) is the set of neighbors of particle \( i \) within distance \( R \). Instead of considering the inter-particle energy between every pair of particles, we only consider the energy of two particles within distance \( R \). We call \( R \) the neighbor radius. Gaussian energy is close to 0 when \( R \gg 5\sigma \). This approximation affect very little to the total energy while significantly reducing the number of items in the energy summation from \( O(N^2) \) to \( O(N) \). We use k-d tree to query the neighbors for each particle. When the frame field has large stretching ratio, it is also necessary to adjust the query radius accordingly since k-d tree is built based on Euclidean distance. The energy and force related to particle \( i \) is \( E_i \) and \( f_i \), which is the sum of inter-particle energies and forces from its neighbors \( \mathbb{N}(i) \).

We use L-BFGS [LN89] to minimize the energy by loosening the constraint. The particles are labeled as four types: fixed particles, sharp edge particles, boundary particles, and free particles. Fixed particles are corner points of the domain boundary. In our implementation, we simply calculate the dihedral angles between neighboring triangles to detect all the sharp edges in the input surface mesh. A corner is identified if it is shared by more than two sharp edges. During the optimization, the gradient of the sharp edge particle will be projected onto the direction of its underlying sharp edge, and the gradient of boundary particles will be projected onto the tangent plane of its boundary surface. After each round of L-BFGS optimization, we will project particles to the domain boundary if it is either outside the domain or inside but close to the boundary. If a boundary particle is close to a sharp edge, then it is projected to the sharp edge and labeled as a sharp edge particle. This is used to maintain the constraint in Eq.(6).

The details of our L-BFGS particle optimization algorithm are illustrated in Alg. 1.

### 3.2.1. Particle Insertion and Deletion

Minimizing the GHK energy encourages each particle to fall into a nearby hole. If there is no initial particles near a hole, then that hole will be left empty. If more than one particles are close to a hole, then those particles will compete for that hole. So the random initialization of particles results in some regions missing particles and some regions packing with extra particles. Hence we need a particle insertion and deletion algorithm to overcome this problem and obtain the desired BCC and FCC lattice patterns.

The existing mesh refinement schemes are designed based on the mesh structure, e.g., inserting a vertex at the center of an edge or the centroid of a face. In the particle optimization stage, we do not build the mesh, which provides efficiency especially for anisotropic cases. Inspired by the existing mesh refinement scheme, we design the following “mesh-free” insertion and deletion schemes.

**Particle Deletion Scheme:** Without connectivity, each particle does not have a well-defined one-ring neighbor. But we can query the neighbors \( \mathbb{N}(i) \) using k-d tree for any particle \( i \). After that we calculate the anisotropic distance to its neighbors and sort the distance in ascending order. Suppose we store the sorted distance in array \( D_i \), the particle \( i \) is deleted if any of the following condition holds:

- \( D_i[0] < 0.5 * l_{\text{closest}} \);
- \( \frac{1}{3} \sum_{k=1}^{3} D_i[k] < 0.75 l_{\text{closest}} \);

**Algorithm 1: L-BFGS Particle Optimization Algorithm**

- **Input:** \( \mathbb{F}, \Omega, \mathbf{T}, \mathbb{V} \)
- **Output:** Optimized \( \mathbb{V} \)

```
while stopping criteria not satisfied do
    Build k-d tree for \( \mathbb{V} \);
    \( E \leftarrow 0 \);
    foreach \( p_i \in \{ \mathbb{V} - \text{fixed particles} \} \) do
        Query the neighbors \( \mathbb{N}(i) \) from k-d tree;
        Calculate \( E_i \) and \( f_i \);
        if \( p_i \) is a sharp edge particle or a boundary particle then
            Update \( f_i \);
        end
        \( E \leftarrow E + E_i \);
    end
    Run L-BFGS with \( E \) and \{\( f_i \mid i = 1...N \}\) to update \( \mathbb{V} \);
    foreach \( p_i \in \mathbb{V} \) do
        if \( p_i \) is outside of domain or its distance to boundary \( \leq 0.3 \) then
            Project and mark it as a boundary particle;
        end
        if \( p_i \) is a boundary particle and its distance to sharp edge is \( \leq 0.3 \) then
            Project and mark it as a sharp edge particle;
        end
    end
end
```

- \( \frac{1}{3} \sum_{k=1}^{3} D_i[k] < 0.85 l_{\text{closest}} \);
- \( \frac{1}{3} \sum_{k=1}^{3} D_i[k] < 0.9 l_{\text{closest}} \);
- \( \frac{1}{3} \sum_{k=1}^{3} D_i[k] < 0.95 l_{\text{closest}} \);

where \( l_{\text{closest}} \) is the closest one-ring neighbor distance in the lattice, e.g., \( l_{\text{closest}} = \sqrt{3}/2l^* \) for BCC, and \( l_{\text{closest}} = \sqrt{2}/2l^* \) for FCC. We denote the set of particles to be deleted as \( S_D \). The general rule of setting the coefficients is stricter constrain for the average distance of more neighbors. The coefficients given above are experimentally chosen based on our observation in the experiments, and used in all results in the paper.

**Particle Insertion Scheme:** Unlike the particle deletion scheme, the first step of particle insertion is to get the insertion candidates. Inspired by one-ring structure of the desired lattice, we collect the candidate set by going through each particle and add all expected positions of its one-ring neighbors to the candidate set \( S_I = \bigcup_{i=1}^{N} \bigcup_{j=1}^{N} \{ p_i + \mathbf{T}_i \times \text{Onering}(k) \} \), where \( \mathbf{T}_i \) is the frame field at particle \( i \). If a candidate is outside the domain, we will project it to the domain boundary. The coinciding duplicates will be removed from the set \( S_I \) and also candidates coinciding with any particle will also be removed. We define two positions as coincidence if their distance is less than 0.1l*. After filtering the candidate set based on coincidence, we will calculate GHK energy for each remaining candidate and sort them in ascending order. Then we examine candidates one by one and pick a candidate if the nearest particle, including previously picked candidates, is at least 0.75l_{closest} away. Too small insertion threshold will cause unnecessary vertices being
inserted and may slow the convergence. Too large insertion threshold will insert less vertices than required. The coefficient 0.75 is set according to our observation in the experiments.

The particle deletion and insertion schemes are performed after each round of L-BFGS optimization. In an L-BFGS optimization round, the particle number is fixed. After particle deletion and insertion, another round of L-BFGS optimization is performed. As the optimization processed, less particles are deleted and inserted. The overall lattice optimization process is given in Alg. 2.

The complexity of our particle optimization algorithm is related to the number of particles. A small target edge length indicates large number of particles, which is time-consuming. To speed up the optimization, we can start with the particle optimization (Alg. 2) by setting the target edge length as 2 l∗. After the optimization is completed, we get a particle set \( V \) with edge length 2 l∗. Using a similar strategy as the particle insertion scheme, we collect the candidates to refine the particle set. For each particle, the point which is \( l^* \) distance away along the frame field vectors is added to the refined candidates \( V_r \). After removing the coincidence candidates in \( V_r \), we take the particles in \( V \) and \( V_r \) as initial particles, and start another round of optimization by setting the target edge length as \( l^* \). With a better initialisation, the optimization converges much faster. If \( l^* \) is too small, we can start with \( 2 l^* \) and do the above trick for \( k \) iterations. Such refinement strategy not only accelerates the optimization, but also helps converge to a better result. Both gradient norm \( EpsG \) and max iteration number \( MaxIts \) are set as the stopping criteria in Alg. 1. \( EpsG \) is set to 1e−2 for each round, except the last round, where \( EpsG = 1e−4 \). If \( k = 0 \), \( MaxRoundNum \) is set to 16. If \( k = 1 \), \( MaxRoundNum = 8 \). If round number is less than 5, \( MaxIts \) is set to 13, otherwise it is set to 8. The last round is without the \( MaxIts \) as stopping criteria. We use the kitten Model as an example to show the energy decreasing with respect to computation time.

### 3.3. Tetrahedral Mesh Generation

After particle optimization, we will connect the particles to build a tetrahedral mesh. Restricted Voronoi Diagram (RVD) [YLL09] is used to build the surface boundary using the boundary particles. We use the RVD class provided by GEOGRAM [Lév15]. Once we get the boundary triangle mesh, we perform the restricted Delaunay tetrahedralization by TetGen [Si15], which does not consider the anisotropic frame field. To get a tetrahedral mesh with respect to the frame field, we perform a set of topological operations [She02b], by using the Gradient-Based Shape Matching Energy [NZL17] as guidance to flip the tetrahedral mesh.

### 4. Experiments

We compare our methods with the state-of-the-art methods [JAYB15, FLSG14, ZGW13]. The implementation of our algorithms are based in C++. The experiments are conducted on a workstation with Intel(R) Xeon E5645 2.40GHz CPU and 32GB DDR3 RAM. The calculation of energy and force for each particle is independent, so we speed up the program with OpenMP. The input of our program includes a volume domain, its associated frame field, and also the target edge length \( l^* \) of the cubic lattice defined in the isotropic space.

**Frame Field**: Frame fields are given as an input. Several existing state-of-the-art algorithms can be used to generate a high-quality cross field for any arbitrary volumetric domain [HTWB11, RSL16, GJTP17, SVB17]. For the convenience, we denote the discrete cross field as \( \mathbf{D} \). We also test our methods with some user-designed frame fields, e.g., rotation along y-axis on torus and highly anisotropic frame fields on cubes.

**Quality Metrics**: To calculate the quality of anisotropic tetrahedral meshes, we first transform the elements \( \tau \) from anisotropic space to isotropic space \( \mathbf{\tau} \). Many anisotropic mesh quality metrics are discussed in [She02a]. We measure the quality by dihedral angles \( \theta \), edge-ratio \( \rho \) and condition \( \kappa = 3\sqrt{\min(\theta)} / (2l_{\mathrm{rms}} A_{\mathrm{rms}}) \), where \( r_{\mathrm{circ}} \) is the circumradius, \( \min(\theta) \) is the shortest edge length, \( l_{\mathrm{rms}} \) is the volume, \( l_{\mathrm{rms}} \) and \( A_{\mathrm{rms}} \) are the root mean square of edge lengths and face areas of a tetrahedron. When \( \mathbf{\tau} \) is a regular tetrahedron, \( \rho_{\mathbf{\tau}} = 1, \kappa_{\mathbf{\tau}} = 1 \). We report histograms and minimum, average and standard deviation for \( \theta_{\min}, \rho \) and \( \kappa \), denoted as \( \theta_{\min}, \rho_{\min}, \kappa_{\min} \), \( \theta_{\max}, \rho_{\max}, \kappa_{\max} \), and \( \sigma(\theta), \sigma(\rho), \sigma(\kappa) \). \( \theta_{\min} \) and \( \theta_{\max} \) are the smallest and the largest dihedral angles of a tetrahedron, respectively.

**Alignment Error**: The alignment quality \( \varepsilon \) is evaluated on the resulting meshes. For each edge \( v_{ij} \) of the resulting tetrahedral mesh, we first transform it to the isotropic space \( v'_{ij} = B_{ij} v_{ij} \), then the smallest angle between \( v_{ij} \) and vectors in ONERING is used to measure the alignment error of edge \( v'_{ij} \), i.e., \( \varepsilon = \min_{k=1}^{N_{\text{closest}}} \arccos \left( \frac{v'_{ij} \cdot \text{ONERING}(k)}{\|v'_{ij}\| \|\text{ONERING}(k)\|} \right) \). Histograms, mean \( \bar{\varepsilon} \) and \( \sigma(\varepsilon) \) are reported for the result meshes.

**Experiment Parameters**: Our experiments show that setting \( \sigma \) in the range \([0.25, 0.35]\) has the similar performance, so we use \( 0.30 \) for all the experiments. The neighbor radius \( R \) is set as \( 1.3 l^* \), which includes all the one-ring neighbors. The

---

**Algorithm 2**: Particle-Based Lattice Optimization Algorithm

<table>
<thead>
<tr>
<th>Input: ( I^*, \Omega, T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: particle set ( V )</td>
</tr>
<tr>
<td>1 Estimate vertex number ( N );</td>
</tr>
<tr>
<td>2 Randomly initialize ( V ); Optimize ( V ) by Alg. 1;</td>
</tr>
<tr>
<td>for ( i \leftarrow 0 ) to MaxRoundNum do</td>
</tr>
<tr>
<td>4 Apply particle deletion scheme, ( S_D \leftarrow ) deleted particles;</td>
</tr>
<tr>
<td>5 ( V \leftarrow V \setminus S_D );</td>
</tr>
<tr>
<td>6 Apply particle insertion scheme, ( S_I \leftarrow ) inserted particles;</td>
</tr>
<tr>
<td>7 ( V \leftarrow V \cup S_I );</td>
</tr>
<tr>
<td>8 if (</td>
</tr>
<tr>
<td>9 ( \text{break}; )</td>
</tr>
<tr>
<td>10 end</td>
</tr>
<tr>
<td>11 Optimize ( V ) by Alg. 1;</td>
</tr>
<tr>
<td>12 end</td>
</tr>
<tr>
<td>13 Optimize ( V ) by Alg. 1;</td>
</tr>
</tbody>
</table>
Gaussian Hole Kernel definition in Eq. (4) takes the inverse of $N_{or}$ as the weight for the negative Gaussian kernels. This is to balance the force used to push particles away and the forces to drag particles to holes. Larger weight will result in more coinciding particles, while with smaller weight, particles are more evenly pushed away but may be more off the desired one-ring structure. However, after a few rounds of optimization with deletion and insertion scheme, the performance of different weights are similar. The other parameters for deletion, insertion and projection of particles are given in the Sec. 3.2. All the experiments are conducted with the same parameters.

**Comparison with Particle2013 [ZGW+13]:** Zhong et al.’s method [ZGW+13] used the traditional Gaussian kernel for generating anisotropic triangular mesh, and can be trivially extended for tetrahedral mesh generation. We compare our Gaussian hole kernel methods, named $BCC$ and $FCC$, with their method, named Particle2013, on three models as shown in Fig. 3 and Table 1. Particle2013 cannot achieve field-aligned meshing results for discrete cross fields. The experiments of our method use both $\mathbf{T} = \mathbf{I}$ and $\mathbf{T} = \mathbf{D}$ (discrete cross fields). Under either rotation field, our method achieves higher quality, e.g., about 3° to 5° growth on $\theta_{\min}$, 0.05 to 0.11 gain on $\theta_{40}$ and thousands less of tetrahedra in $\#T_{\leq 40}$. Those gains are coming from the lattice-guided alignment, producing high quality BCC and FCC tetrahedra.

BCC alignment provides higher $\theta_{\min}$ but also higher $\theta_{\max}$ than FCC alignment. FCC alignment has smaller $\#V$ and $\#T$, on three models as shown in Fig. 3 and Table. 1. Comparisons with CVT and ODT: We also compare with the
Rotation Field Alignment for Improving Mesh Quality: After showing our better mesh quality results as compared to Particle2013, CVT, and ODT methods, we would like to show that for some models with rotational features, the alignment with its rotation field can produce meshes with better quality. We use two models for such illustration: Torus (Fig. 6) and Fancyring (Fig. 7). The frame field we tried on Torus is the rotation along y-axis \( R_y \), and the frame field for Fancyring is a discrete cross field generated by [HTWB11], denoted as \( D \). We compare them with the results generated by identity field \( I \). It can be seen from these two experiments: if the rotation field aligns very well with the shape or the features of the geometry, we can get better shape approximation as well as higher tetrahedral qualities as shown in Table 3.

Field Alignment for Anisotropic Tetrahedral Meshing:
To further explore the performance of our field-aligned and lattice-guided methods, we conduct experiments using the highly anisotropic fields on Cube, and compare with LCT method [FLSG14] as shown in Fig. 8 and Fig. 9. Compared with LCT, our BCC and FCC results show higher quality due to the strong directional control and the advantage of lattice-alignment. The detailed quality statistics are given in Table 4.

**Robustness:** We demonstrate the robustness of our method by experiments on Teddy with different numbers of vertices on the same discrete frame field. The minimal dihedral angle $\theta_{\text{min}}$, edge radius ratio $r$, condition $\kappa$, alignment error $\varepsilon$ of the resulting meshes, and their optimization time along with different vertex numbers are shown in Fig. 10. The results of BCC and FCC are shown in red and blue curves, respectively.

5. Discussion and Future Work

It should be noted that our method can be easily extended to solve field-aligned anisotropic triangular meshing for surfaces. This can be achieved by defining six holes in the GHK of Eq. (3) on the tangent plane of surface. As shown in Fig. 11, our method can obtain better mesh quality of anisotropic triangular meshes compared to Particle2013 [ZGW13] and LCT [FLSG14]. Here $r_0$ is the ratio.
Table 3: The quality statistics of rotation alignment experiments on Torus and Fancyring. \#V and \#T are the numbers of vertices and tetrahedra of the result meshes. \#T_{\theta<20\degree} and \#T_{\theta<40\degree} are the numbers of tetrahedra with \(\theta_{\min} < 20\degree\) and \(\theta_{\min} < 40\degree\), respectively. The minimum, mean, and standard deviation of smallest dihedral angle \(\theta_{\min}\), edge radius ratio \(\rho\), condition \(\kappa\), and computation time are provided. Note that the best values are highlighted in bold for each group.

| Model  | T | Alg. | \#V | \#T | \(\theta_{\min}/\theta_{\max}/\theta_{\kappa}\) | \(\rho_{\min}/\rho/\rho_{\max}\) | \(\kappa_{\max}/\kappa/\kappa_{\min}\) | \(E/I/G(\epsilon)\) | \#T_{\theta<20\degree} | \#T_{\theta<40\degree} | \#T | Dist | Time(s) |
|--------|---|------|-----|-----|---------------------------------|---------------------------------|---------------------------------|-----------------|-----------------|-----------------|-----|-------|
| Torus  | I | BCC | 2,274 | 18,70/54,06/77.85 | 0.37/0.85/0.11 | 0.34/0.90/0.09 | 6.02/7.65 | 5 | 798 | 10,605 | 0.500 | 6.98 |
| Torus  | I | FCC | 2,299 | 15.62/55.37/70.70 | 0.34/0.85/0.09 | 0.30/0.89/0.08 | 4.98/6.37 | 1 | 541 | 10,486 | 0.341 | 6.66 |
| Torus  | R | BCC | 2,440 | 28.10/55.05/66.54 | 0.43/0.86/0.09 | 0.55/0.91/0.07 | 5.69/6.97 | 0 | 520 | 11,389 | 0.377 | 7.91 |
| Torus  | R | FCC | 2,347 | 20.87/54.38/38.44 | 0.35/0.85/0.10 | 0.42/0.89/0.09 | 5.60/6.19 | 0 | 677 | 11,106 | 0.318 | 11.76 |
| Fancyring | I | BCC | 4,775 | 17.22/51.76/68.65 | 0.23/0.81/0.13 | 0.37/0.88/0.10 | 9.66/9.97 | 14 | 2,587 | 21,420 | 0.542 | 8.42 |
| Fancyring | I | FCC | 4,810 | 13.79/52.05/9.56 | 0.19/0.80/0.12 | 0.29/0.87/0.10 | 8.15/7.70 | 15 | 2,493 | 21,094 | 0.480 | 14.44 |
| Fancyring | D | BCC | 4,187 | 22.93/53.98/64.04 | 0.29/0.85/0.08 | 0.37/0.90/0.07 | 8.55/8.84 | 0 | 417 | 18,992 | 0.195 | 21.56 |
| Fancyring | D | FCC | 3,979 | 22.14/53.57/6.96 | 0.33/0.84/0.08 | 0.48/0.90/0.07 | 6.76/5.76 | 0 | 527 | 17,256 | 0.203 | 14.93 |

(a) BCC \((T = I)\) (b) BCC \((T = R)\) (c) FCC \((T = I)\) (d) FCC \((T = R)\)

Figure 6: BCC and FCC experiments on Torus with \(T\) and \(R\) rotation fields. The yellow ones are the clipping views.

Table 4: Statistics of mesh quality and time consumption compared with LCT [FLSG14]. \#V and \#T are the numbers of vertices and tetrahedra of the result meshes. \#T_{\theta<20\degree} and \#T_{\theta<40\degree} are the numbers of tetrahedra with \(\theta_{\min} < 20\degree\) and \(\theta_{\min} < 40\degree\), respectively. The mean of smallest dihedral angle \(\theta_{\min}\), edge radius ratio \(\rho\), condition \(\kappa\), and computation time are provided. Note that the best values are highlighted in bold for each group.

| Model  | T | Alg. | \#V | \#T | \(\theta_{\min}/\theta_{\max}/\theta_{\kappa}\) | \(\rho_{\min}/\rho/\rho_{\max}\) | \(\kappa_{\max}/\kappa/\kappa_{\min}\) | \(E/I/G(\epsilon)\) | \#T_{\theta<20\degree} | \#T_{\theta<40\degree} | \#T | Dist | Time(s) |
|--------|---|------|-----|-----|---------------------------------|---------------------------------|---------------------------------|-----------------|-----------------|-----------------|-----|-------|
| LCT    |   | BCC  | 1,369 | 24.84/53.71/8.63 | 0.89/3.19 | 0 | 503 | 8,117 | 4.3 |
| Fig. 8 |   | BCC  | 2,476 | 36.38/59.95/8.92 | 0.91/3.19 | 0 | 0 | 11,988 | 16.28 |
| FCC    |   | BCC  | 2,471 | 52.85/58.75/9.09 | 0.91/3.19 | 0 | 0 | 11,520 | 11.13 |
| LCT    |   | FCC  | 6,338 | 15.51/49.87/8.78 | 0.89/20.94 | 88 | 4,220 | 31,840 | 72.6 |
| Fig. 9 |   | FCC  | 6,410 | 14.26/53.73/8.85 | 0.89/5.37 | 8 | 2,297 | 33,089 | 56.88 |
| FCC    |   | FCC  | 6,498 | 23.44/55.27/8.55 | 0.91/3.94 | 0 | 632 | 32,993 | 57.59 |

Figure 7: BCC and FCC experiments on Fancyring with different rotation fields. The red ones are the tetrahedra with \(\theta_{\min} < 40\degree\).

of vertices with degree-6. The quality of a triangle is measured by \(\xi = 4\sqrt{3}a\rho/h\), where \(a\) is its area, \(p\) is its perimeter and \(h\) is its longest edge length in its mapped isotropic space.

Fig. 6 and Fig. 7 show two examples that our rotation-field-aligned BCC and FCC methods might improve mesh quality for models having rotational shapes and features. In the future we would like to investigate in depth the relationship between the generation of frame-fields and the quality of field-aligned BCC and FCC meshes, in order to come up with some better field genera-

© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.
Figure 8: Anisotropy variation along a single direction on Cube $[0.1, 1.1]^3$. The inverse of frame field is defined as $B = \text{diag} \left( 1.0125 - e^{-|x-0.6|} \right)^{-1}, 1, 1 \right)$. The second row shows the clipping views of the result tetrahedral meshes. The last four rows show the histograms of dihedral angle, edge radius ratio, condition, and alignment error.

6. Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the paper. Saifeng Ni and Xiaohu Guo are partially supported by National Science Foundation (NSF) under Grant No. IIS-1149737. Zichun Zhong is partially supported by NSF ACI-1657364 and the Wayne State University Startup Grant. Jin Huang is partially supported by NSFC (No. 61522209, 61210000).

References


Figure 10: The first two rows show BCC and FCC results in different vertex numbers. The next two rows show the quality statistics and time consumption with different numbers of vertices on Teddy.

Figure 11: Field-aligned anisotropic triangular meshing on Cylindar, compared with Particle2013 [ZGW∗13] and LCT [FLSG14]. The three columns are the results of Particle2013 method, LCT method, and our method, respectively. The first two rows show the resulting surface meshes and the zoom-in views of the narrow part. $r_2$ shown in the first row is the ratio of vertices with degree 6 in the result mesh. The last two rows are the histograms of $\theta_{min}$ and $\xi$ of.


fast computation.
2
ACM Transactions on Graphics 28, 4 (2009), 101. 1, 2


