Computers & Graphics 53 (2015) 28–36

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

40 years of Computer Graphics in Darmstadt

Beam meshes

Ronald Richter, Marc Alexa *

Marchstr. 23, 10587 Berlin, Germany

ARTICLE INFO

Article history:
Received 18 August 2015
Accepted 20 August 2015
Available online 9 October 2015

Keywords:
Geometry processing
Centroidal Voronoi diagram
Digital manufacturing

ABSTRACT

We present an approach for representing free-form geometry with a set of beams with rectangular cross-section. This requires the edges of the mesh to be free of torsion. We generate such meshes in a two step procedure: first we generate a coarse, low valence mesh approximation using a new variant of anisotropic centroidal Voronoi tessellation. Then we modify the mesh and create beams by incorporating constraints using iterative optimization. For fabrication we provide solutions for designing the joints, generating a cutting place for CNC machines, and suggesting a building sequence. The approach is demonstrated at several virtual and real results.

1. Introduction

The recent trend in what is commonly referred to as 3d printing has also reignited the interest in other forms of automatic or semi-automatic manufacturing of arbitrary free-form shapes. A particularly interesting approach is based on cutting out planar pieces from a sheet of material and assembling the pieces to represent the volume or boundary of the shape [1–5].

In this work, we aim at physically realizing an arbitrary polygon mesh in timber, plastic, or steel by one planar beam for each edge in the mesh (see Fig. 1). Compared to an arbitrary polygon mesh, the main geometric limitation for this to be possible is that each edge needs to be torsion-free: edges lie in planes that intersect in a common line in the vertices (Fig. 1, bottom). Moreover, we consider additional constraints that make manufacturing simpler, for example extruding the beam from a constant cross section, i. e. not only its thickness is constant but also its height. We call meshes with torsion-free edges *beam meshes*. The goal of this work is to approximate a given 3d geometry with a beam mesh and to manufacture it.

Requiring beams to be straight and have constant height means that the polygon mesh has an offset mesh with parallel edges. Meshes with *planar faces* that have parallel offsets have been analyzed in detail in the seminal work of Pottmann et al. [6]. However, such piecewise planar meshes, for which a parallel offset mesh exists, form a restricted (linear) space and it is notoriously difficult to approximate arbitrary free-form shapes with planar meshes.

In contrast to the work by Pottmann et al. [6], in our approach we consider polygon meshes with arbitrary, not necessarily planar faces. We observe that generically *dual triangle meshes* are torsion free (Section 2). Consequently, we first approximate the given input geometry with a coarse dual triangle mesh (Section 4). Then we optimize the edges of this mesh to satisfy several constraints derived from practical considerations (Section 5). This two-step approach creates the desired beam mesh.

We also explain how to physically realize the beam mesh (Section 6). The non-zero thickness of the beams requires consideration of the geometry at the joints. The beams are then cut out of planar sheets of material using CNC machining. We show how to lay out the beams on a planar surface and how to mark the beams such that the subsequent construction is easy.

We provide several examples of physically realizable beam meshes, some of which we have actually constructed out of wood.

2. Preliminaries

We represent the coarse polygon mesh, which will be the basis of the beam mesh, by its $\mathfrak{v}$ vertices

$$V = (\mathbf{v}_0, \mathbf{v}_1, \ldots), \quad \{\mathbf{v}_i \in \mathbb{R}^3\} \tag{1}$$

and denote by Δ the matrix that generates $\mathfrak{e}$ directed edge vectors

$$\mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i \tag{2}$$

from the vertices. Faces are shortest cycles of edges. We assume that the faces form a closed manifold. Under this assumption Δ represents the combinatorics of the mesh. For convenience we denote by $\mathcal{L}_i$ the cyclically ordered set of vertices connected to vertex i by an edge. Vertices have an associated (non-vanishing) normal vector $\mathbf{n}_i \in \mathbb{R}^3$, $\|\mathbf{n}_i\| > 0$.

The object of interest in this work is a *beam quad*, describing a planar beam as an offset to the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ of an edge $\mathbf{e}_{ij}$

* Corresponding author. Tel.: +49 30 314 73100; fax: +49 30 314 23596
E-mail address: marc.alexa@tu-berlin.de (M. Alexa).

http://dx.doi.org/10.1016/j.cag.2015.08.007
0097-8493/

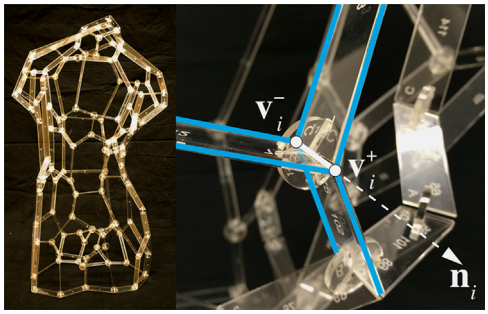

Fig. 1. A geometric shape represented by a mesh of beams, i.e. edges with a rectangular cross section. Beam quads are planar and intersect in one line segment $(\mathbf{v}_i^+, \mathbf{v}_i^-)$ point in the direction of the surface normal $\mathbf{n}_i$ in each vertex.

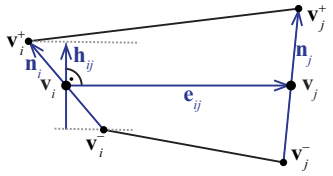

Fig. 2. Notation for a beam in the beam mesh representation.

along the positive and negative normal directions:

$$\begin{aligned}\mathbf{v}_i^- &= \mathbf{v}_i - \mathbf{n}_i\\ \mathbf{v}_i^+ &= \mathbf{v}_i + \mathbf{n}_i\\ \mathbf{v}_j^+ &= \mathbf{v}_j + \mathbf{n}_j\\ \mathbf{v}_j^- &= \mathbf{v}_j - \mathbf{n}_j\end{aligned} \tag{3}$$

For reference, we note that a vertex $\mathbf{v}_i$ is located in the centroid of the offset vertices, i. e.

$$\mathbf{v}_i = \frac{\mathbf{v}_i^- + \mathbf{v}_i^+}{2}. \tag{4}$$

We call the edge

$$\mathbf{v}_i^+ - \mathbf{v}_i^- = 2\mathbf{n}_i \tag{5}$$

the *normal edge* of a beam quad incident at vertex i; and we call the edges

$$\mathbf{v}_j^\pm - \mathbf{v}_i^\pm = \mathbf{e}_{ij} \pm (\mathbf{n}_j - \mathbf{n}_i) \tag{6}$$

offset edges of the beam quad for edge $\mathbf{e}_{ij}$.

We measure *height* of the beam quad orthogonal to the mid-edge $\mathbf{e}_{ij}$. The height vector in vertex i relative to the beam quad ij is

$$\mathbf{h}_{ij} = 2\left(\mathbf{n}_i - \frac{\mathbf{n}_i^\top \mathbf{e}_{ij}}{\mathbf{e}_{ij}^\top \mathbf{e}_{ij}}\mathbf{e}_{ij}\right) \tag{7}$$

and yields the height of the beam ij in vertex i as $\|\mathbf{h}_{ij}\|$. The height of the quad is not necessarily constant and for the height vector in vertex j of the same quad we use the notation $\mathbf{h}_{ji}$. The notation is summarized in Fig. 2.

3. Overview

Using beam quads, the requirement that an edge is torsion-free is equivalent to the associated beam quad being planar. This is the case if the two normals $\mathbf{n}_i, \mathbf{n}_j$ and the edge vector $\mathbf{e}_{ij}$ are contained in the same plane, that is when

$$\det(\mathbf{n}_i, \mathbf{n}_j, \mathbf{e}_{ij}) = 0, \tag{8}$$

which includes the case in which the normals $\mathbf{n}_i, \mathbf{n}_j$ are parallel and the edge direction is arbitrary. However, we do ask that each of the normals $\mathbf{n}_i, \mathbf{n}_j$ and the edge vector $\mathbf{e}_{ij}$ are linearly independent, which means the height vectors $\mathbf{h}_{ij}, \mathbf{h}_{ji}$, and thus the heights, are non-zero. Under this assumption, Eq. (8) implies that all beam quads incident in a vertex $\mathbf{v}_i$ meet in the common line defined by the normal $\mathbf{n}_i$.

Given the vertex geometry of a general polygon mesh, the beam quads will in general not be planar if the normal vectors for the vertices are computed in the usual manner by summing up the cross products of adjacent incident edge vectors:

$$\mathbf{n}_i^* = \sum_{j \in \mathcal{L}_i} \mathbf{e}_{ij} \times \mathbf{e}_{i(j+1)}, \tag{9}$$

then the beam quads will in general not be planar.

Hence, we need to optimize the normals $\{\mathbf{n}_i\}$ to satisfy the torsion free constraint. This gives us $2\mathfrak{v}$ degrees of freedom (because the length of the normals is irrelevant for the torsion), while the planarity induces $\mathfrak{e}$ constraints. Consequently, it is important to have a large number of vertices relative to the number of edges. This just means the vertex degree, i. e. the number of edges incident on a vertex, should be as small as possible. The smallest useful vertex degree is 3, so we aim at using a polygon mesh with constant vertex degree of 3 or, in other words, a dual triangle mesh. Note that, indeed, the ratio of triangle and edges in a triangle mesh is 2/3, so that this is also the ratio of vertices and edges in a dual triangle mesh. As we have 2 degrees of freedom per constraint, we conclude that dual triangle meshes always have normals that are torsion free.

Based on this observation, our strategy is as follows:

1. Generate a coarse dual triangle mesh that represents the input geometry. We do this by first tessellating the input geometry into featureless faces and then extracting a polygon mesh with straight edges. This step generates the combinatorial structure Δ and a first approximation of the geometry $\{\mathbf{v}_i\}$.
2. Compute beam quads or, equivalently, directions $\{\mathbf{n}_i\}$ that satisfy the constraints. Apart from planarity of the quads we also consider constraints on the height of the elements or the smallest edge lengths. To make these constraints feasible we also allow the initial vertex positions to be updated.

Based on the resulting beam mesh, we compute the details to actually manufacture the shape from a given material with non-zero thickness. This requires additional geometric considerations, in particular for adding joints at the vertex positions. These steps are illustrated in Fig. 3.

4. Base mesh generation

For a given boundary representation of a three-dimensional shape $\mathcal{S}$, for example a triangulated surface, we search for a coarse dual triangle mesh. The goal is to balance the desire for a small number of beams with the required preservation of the prominent features of the shape.

We employ Voronoi diagrams for generating a dual triangle mesh [7,8]. Given locations $\mathbf{x}_i \in \mathbb{R}^3$, the Voronoi diagram is defined by cells Ω_i formed by all points in space that are closest to $\mathbf{x}_i$:

$$\Omega_i = \{\mathbf{p} \in \mathbb{R}^3 : d(\mathbf{x}_i, \mathbf{p}) < d(\mathbf{x}_j, \mathbf{p})\}. \tag{10}$$

Intersecting the Voronoi cells Ω_i against the shape $\mathcal{S}$ results in a tessellation of the surface into faces $\{\Omega_i \cap \mathcal{S}\}$. Generically, three faces meet in one point on the surface, i. e. vertices have degree three, as desired.

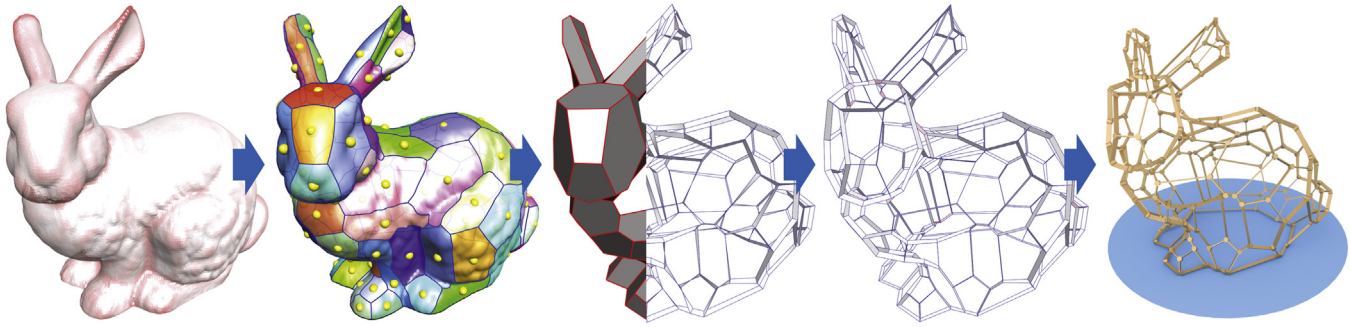

Fig. 3. Overview of the pipeline: the input mesh (left) is tessellated using anisotropic CVT (middle left). Connecting the Voronoi vertices with straight edges yields initial beams, which are not torsion free (middle). The beams quads are then optimized to planar and satisfy other manufacturing constraints (middle right), leading to the constructible beam mesh representation (right).

In order to generate a Voronoi diagram with k cells approximating the shape, we minimize the following tessellation energy:

$$E(\mathbf{x}_0, \mathbf{x}_1, \ldots) = \sum_{i=0}^{k-1} \int_{\Omega_i} (\mathbf{p}-\mathbf{x}_i)^\top \mathbf{M}_i (\mathbf{p}-\mathbf{x}_i)\, \mathrm{d}\mathbf{p} \tag{11}$$

The minimizer of this functional is well behaved and commonly called the *centroidal Voronoi diagram* (CVT), as the centers $\mathbf{x}_i$ end up in the centroid of the cells [7]. The symmetric positive semi-definite matrix $\mathbf{M}_i \in \mathbb{R}^{3\times 3}$ defines the metric of each cell and leads to anisotropic CVTs [8]. It has recently been shown that a good choice for the metric is the inverse of the covariance matrix $\mathbf{C}_i$ of the surface geometry in the cell [9], given by

$$\mathbf{C}_i = \int_{\Omega_i \cap \mathcal{S}} (\mathbf{p}-\mathbf{x}_i)(\mathbf{p}-\mathbf{x}_i)^\top \, \mathrm{d}\mathbf{p}. \tag{12}$$

This choice is also motivated by techniques from machine learning, namely expectation maximization for Gaussian mixture models [10, Chapter 9].

As the input geometry is represented by the beams only, it is desirable that the faces contain no features. This motivates setting the metric proportional to the inverse covariance, i. e. $\mathbf{M}_i \propto \mathbf{C}_i^{-1}$, so that distances are measured according to the anisotropy of the cell; concretely, in flat cells, distances in tangent direction are smaller compared to cells with a non-trivial height. This leads to larger cells in flat areas and hence the desired result.

Our main idea is to homogenize this variation in normal direction by scaling the metric according to the variation in normal direction. For this, consider the eigen-decomposition of the covariance matrix

$$\mathbf{C}_i = \mathbf{Q}_i \,\mathrm{diag}(\lambda_{i_2}, \lambda_{i_1}, \lambda_{i_0}) \mathbf{Q}_i^\top. \tag{13}$$

Because $\mathbf{C}_i$ is positive semi-definite by construction, the eigenvectors and eigenvalues are real and, moreover,

$$0 \le \lambda_{i_0} \le \lambda_{i_1} \le \lambda_{i_2}. \tag{14}$$

We consider the eigenvector corresponding to the smallest eigenvalue λ_{i_0} to be the normal direction. Now to homogenize the variation in normal direction means to measure distances in this direction identically in all cells. This can be achieved by normalizing the metric with respect to the eigenvalue associated with the normal direction, i. e. we use

$$\mathbf{M}_i = \lambda_{i_0} \mathbf{C}^{-1} = \mathbf{Q}_i \,\mathrm{diag}\left(\frac{\lambda_{i_0}}{\lambda_{i_2}}, \frac{\lambda_{i_0}}{\lambda_{i_1}}, 1\right) \mathbf{Q}_i^\top. \tag{15}$$

The last expression shows how this approach also overcomes a limitation of our earlier work [9]: for the desirable planar regions, the covariance matrix is singular, and the metric would be undefined. The solution in (15) is well defined for planar surfaces – it would cause numerical difficulties only when the surface in a cell degenerates to a line or a point, i. e. if $\lambda_{i_1} \approx 0$. This, however, is unlikely in practice, and we never encountered the problem in our numerical experiments.

Minimization: We now tackle the minimization problem

$$\arg\min_{\{\mathbf{x}_i, \Omega_i\}} E(\mathbf{x}_0, \mathbf{x}_1, \ldots) \tag{16}$$

for the tessellation energy defined in (11). We obtain a minimum by gradient descent, resulting in Lloyd's algorithm [9]. In the first half-step, the cells Ω_i are fixed so that the energy can be considered per cell. It is well known that the Euclidean geometric mean minimizes the functional in (11), i. e. it is optimal to update

$$\mathbf{x}_i \leftarrow |\,\Omega_i \cap \mathcal{S}\,|^{-1} \int_{\Omega_i \cap \mathcal{S}} \mathbf{p}\, \mathrm{d}\mathbf{p}. \tag{17}$$

In the second half-step, we fix the $\mathbf{x}_i$ and then compute optimal Ω_i. Assuming that the surface $\mathcal{S}$ is smooth, $\mathbf{M}_i$ varies only slowly with Ω_i and we can fix it as well. Then the optimal choice for Ω_i is simply the Voronoi cell. The two steps comprise Lloyd's algorithm, with the only difference to the original algorithm being the distance metric $\mathbf{M}_i$ defined per cell.

Initialization: The optimization algorithm finds a local minimum of the objective function, but it is unlikely that this is also the global minimum. This means a proper initialization of the sites is essential. In general, the initial solution should be as close as possible to the optimal solution. In our setting the density of sites directly relates to the mean curvature of the surface. Based on this observation we first estimate the mean curvature of the surface (per face) and use it as a density function for randomly sampling initial sites on the surface. The metric tensor of each site is initialized with the identity matrix.

Number of sites: In the current approach we empirically estimate the number of sites, and hence also the number of polygons in the coarse mesh. Although this works sufficiently well in practice, determining k automatically is preferable. We experimented with farthest point insertion [11] and heuristically adding new sites by splitting the region with the largest variance in normal direction. We stop adding new sites when the variance is below a fixed threshold. However, finding the threshold which produces the optimal number of sites is again a non-trivial estimation problem.

Results: In Fig. 4 we show several results for different numbers of sites and in Fig. 5 we compare to our earlier approach, which we already showed in [9] to be more robust than VSA [11].

5. Beam optimization

We now describe how to generate the beam mesh from the coarse mesh obtained as described in the last section. As variables for this process we use the vertices of the beam quads, i. e. the two vertices $(\mathbf{v}_i^-, \mathbf{v}_i^+)$ for vertex i defined as the positive and negative

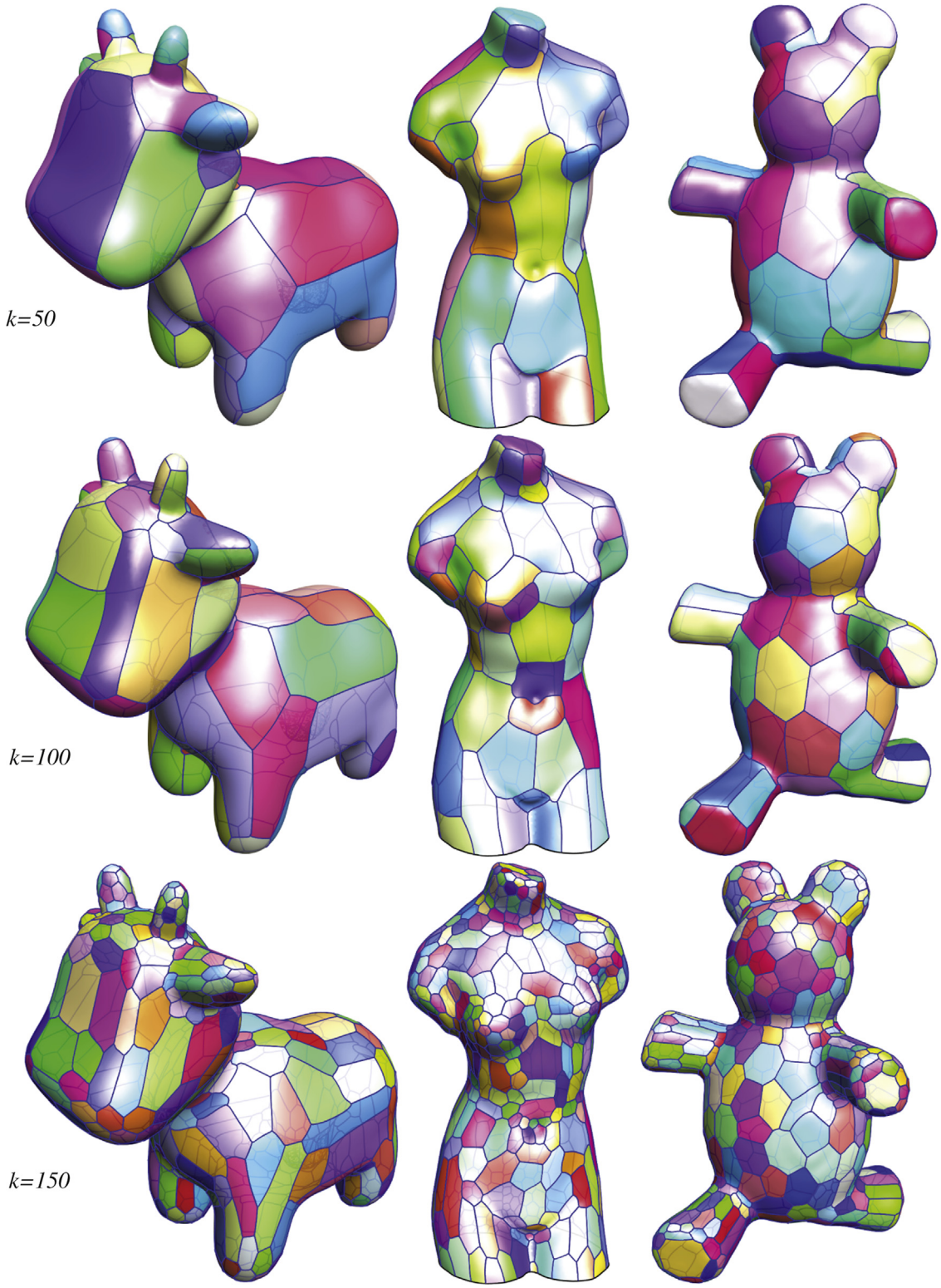

Fig. 4. Tessellation results for different numbers of sites k.

normal offset, cf. Eq. (3). Each beam quad is optimized to conform to several desirable properties (which we detail later).

Our optimization procedure follows the ideas of ShapeUp [12]. We describe each desirable property as a *projection*. Then the optimization alternates between projecting the *edges* of each beam quad into the different constraint spaces and globally reconciling the contradicting projections into new beam quad vertex positions $\{(\mathbf{v}_i^-, \mathbf{v}_i^+)\}$ using least squares. This iterative procedure minimizes an energy that measures (weighted) squared distance from the constraint spaces [12]. The global step requires solving only a (sparse) linear system, while the local projections may be non-linear. For details about the resulting global linear system we refer the reader to the original publication [12].

We start the iterations with the vertices $\{\mathbf{v}_i\}$ obtained as described in Section 4. Based on the normals $\mathbf{n}_i^*$, either computed from the polygon mesh (see Section 2) or from the underlying

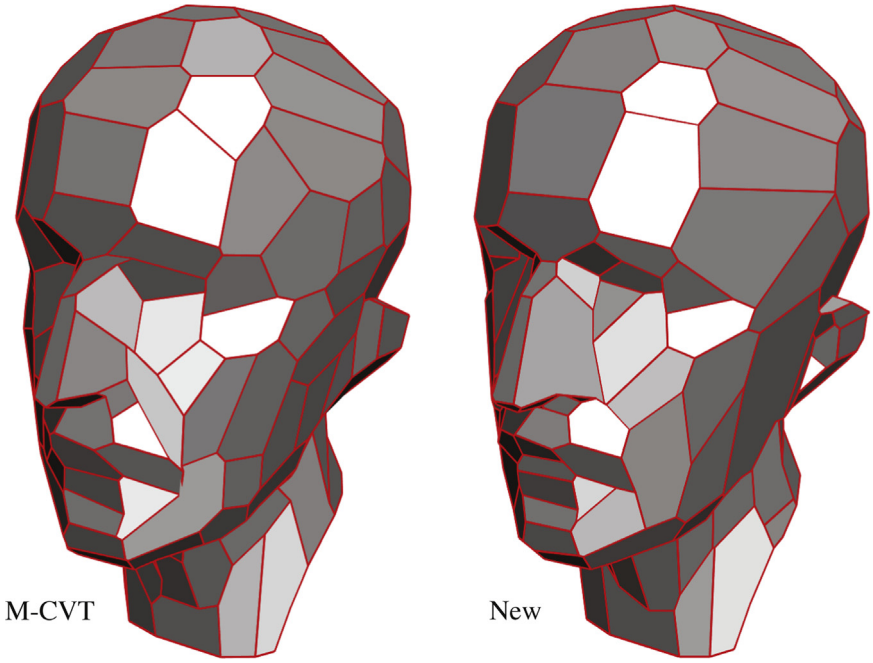

Fig. 5. Comparison of tessellation results between M-CVT [9] (left) and our new variant (right). Both results were generated without isotropic regularization (see Section 4 for more details). Note how M-CVT tends to generate faces of rather equal size, while our new method better adapts to featureless areas with larger faces.

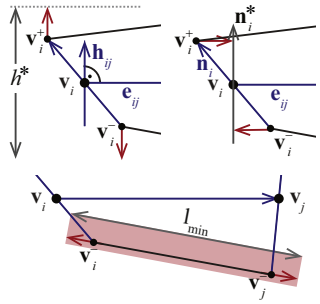

Fig. 6. Beam quads satisfy several constraint. Shown here are the projections for constraints on height, normal direction, and minimal length.

shape $\mathcal{S}$, we set

$$\mathbf{v}_i^{\pm} = \mathbf{v}_i \pm \frac{\mathbf{n}_i^*}{\|\mathbf{n}_i^*\|}. \tag{18}$$

Then we consider the following constraint projections:

Planarity: We compute the area vector of the beam quad, which is, coincidentally, the normal of a best fitting plane. Elementary computations based on the notation defined in Section 2 yield

$$\mathbf{n}_{ij} = (\mathbf{n}_i + \mathbf{n}_j) \times \mathbf{e}_{ij}. \tag{19}$$

Then we project all edge vectors of the quad onto the best fitting plane. So for any edge $\mathbf{q}$ of the beam quad, its projected version is

$$\mathbf{q} - \frac{\mathbf{q}^\top \mathbf{n}_{ij}}{\mathbf{n}_{ij}^\top \mathbf{n}_{ij}} \mathbf{n}_{ij}. \tag{20}$$

Height: We consider several options to constrain the heights $\|h_{ij}\|$, $\|\mathbf{h}_{ji}\|$. We focus on the height in vertex i and, therefore, on how to project $(\mathbf{v}_i^+ - \mathbf{v}_i^-)$. Vertex j and its normal edge are treated analogously:

- Using a single target height h^* and a tolerance ϵ_h: If $\|h_{ij}\|$ is outside $h^* \pm \epsilon_h$ the edge $(\mathbf{v}_i^+ - \mathbf{v}_i^-)$ is scaled by $(h^* \pm \epsilon_h)/\|h_{ij}\|$ to fit the desired tolerance interval.
- Using several target heights $\{h_l^*\}$: The edge $(\mathbf{v}_i^+ - \mathbf{v}_i^-)$ is then scaled to fit the target height h_l^* that minimizes the absolute difference $|\,\|h_{ij}\| - h_l^*\,|$.

Parallelity: If we allow a variety of different heights based on existing beam cross-sections, it might be good to constrain the offset edges to be parallel. Parallel edges can be enforced by projecting both offset edges onto the mid-edge, i. e.

$$\frac{\mathbf{e}_{ij}^\top (\mathbf{v}_j^{\pm} - \mathbf{v}_i^{\pm})}{\mathbf{e}_{ij}^\top \mathbf{e}_{ij}} \mathbf{e}_{ij}. \tag{21}$$

Offset direction: Recall that we may compute a normal direction $\mathbf{n}_i^*$ for vertex i based on the original geometry $\mathcal{S}$ or from the vertex geometry of the polygon mesh (see Section 2). We may ask that the edge $(\mathbf{v}_i^+ - \mathbf{v}_i^-)$ is parallel to this direction. This constraint results in a projection of the edge onto the normal, i. e. the projection is simply the product with the normalized normal direction $\mathbf{n}_i^*/\|\mathbf{n}_{ij}\|$.

Length: We ask that the offset edges are not shorter than a minimum length $l_{min} \geq 0$. Note that this constraint is necessary to avoid that beam quads are not simply (do not self-intersect). If any of the edges $(\mathbf{v}_j^{\pm} - \mathbf{v}_i^{\pm})$ is shorter than l_{min}, we scale it by $l_{min}/\|\mathbf{v}_j^{\pm} - \mathbf{v}_i^{\pm}\|$.

6. Fabrication

Having discussed how to create a beam mesh with edges of non-zero height, we now consider the practical manufacturing process. For this it is necessary that the thickness τ of the beam edges is not only positive but constant. The actual thickness has implications on the dimensions of beam quad and, thus, for the manufacturing process.

For generating the physical realization of the beams we focus on cutting them using computer numerical controlled (CNC) machines from *sheets* of material. As material we consider wood, plastic, and steel; however, any material that can be cut and joined is possible. The main goal is to reduce the amount of unused material, which in many cases is wasted (wood for example), or at least produces extra cost when re-used (metals).

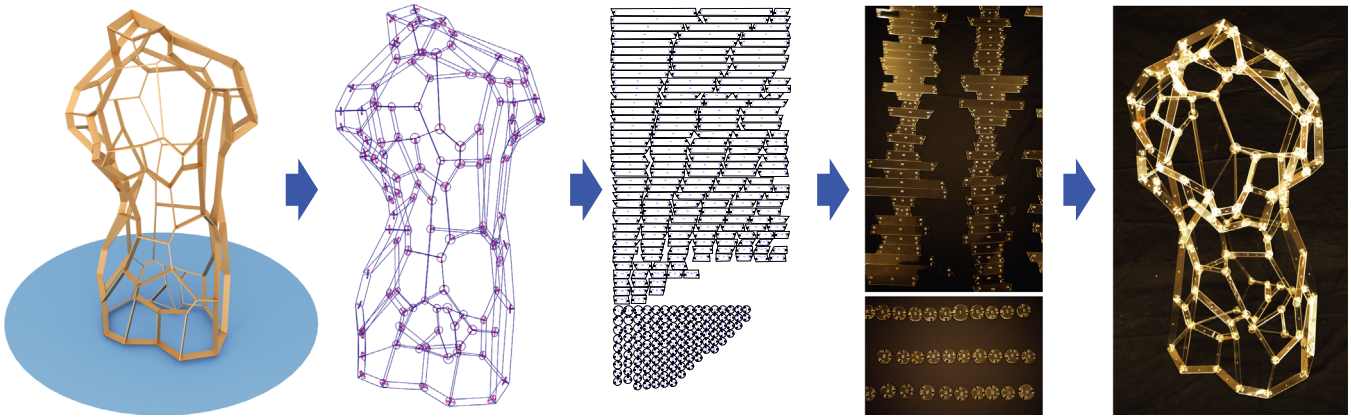

Fig. 7. Starting from the beam mesh, the beam geometry is computed, taking into account the non-zero thickness of the material. The cut plan includes disks that can support joints for assembly. The photograph shows the constructed beam mesh including the support disks.

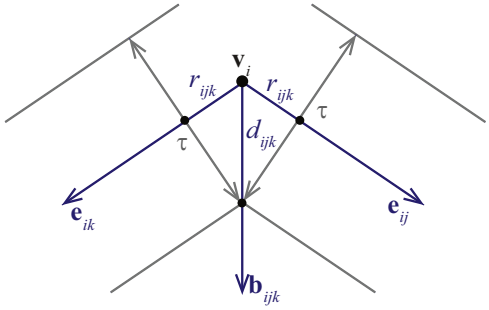

Fig. 8. Two beams with finite thickness τ meeting in vertex $\mathbf{v}_i$. The surfaces of the beams intersect along the bisector $\mathbf{b}_{ijk}$ at a distance d_{ijk} from the vertex.

Lastly, we suggest markers for the beams, potentially helpful supports for the assembly, and a reasonable order for joining the pieces. The necessary steps together with a result made out of acrylic are illustrated in Fig. 7.

6.1. Joints

Assume that the beams have a rectangular cross section. The direction of constant thickness for beam ij is orthogonal to the beam quad, so parallel to $\mathbf{n}_i \times \mathbf{e}_{ij}$. Since this direction is orthogonal to $\mathbf{n}_i$ for all beams incident on i, the geometry of the situation can be analyzed by projecting the beams along $\mathbf{n}_i$. The unit projection of the edges onto the plane orthogonal to $\mathbf{n}_i$ is simply

$$\mathbf{e}'_{ij} = \mathbf{n}_i - \mathbf{h}_{ij}. \tag{22}$$

Two adjacent edges $\mathbf{e}'_{ij}$ and $\mathbf{e}'_{ik}$ extruded orthogonally intersect on the bisector $\mathbf{b}_{ijk} = \mathbf{e}'_{ij} + \mathbf{e}'_{ik}$ between the edges. Let $\hat{\mathbf{e}}_{ij}$ and $\hat{\mathbf{b}}_{ijk}$ be the unit vectors corresponding to the projected edges and the bisector. Then the linear system

$$r_{ijk}\hat{\mathbf{e}}_{ij} + \tau\hat{\mathbf{e}}_{ij}^{\perp} = d_{ijk}\hat{\mathbf{b}}_{ijk} \tag{23}$$

defines the intersection of the beams ij and ik with thickness τ to be at a distance d_{ijk} from $\mathbf{v}_i$ or, in other words, the line of intersection between the beams goes through $\mathbf{v}_i + d_{ijk}\hat{\mathbf{b}}_{ijk}$. Moreover, the distance of the beams ij and ik from the center is r_{ijk}. This setup is illustrated in Fig. 8.

Now consider the three beams ij, ik, il meeting in vertex i (Fig. 9, left). If the beams could be cut at two angles, one could simply use the cut geometry between any two edges to define the cuts, i. e. edge ik would end in the geometry given by $d_{ijk}\hat{\mathbf{b}}_{ijk}, \mathbf{0}, d_{ikl}\hat{\mathbf{b}}_{ikl}$. If cuts were planar but at an angle, the cut could be defined from the points $d_{ijk}\hat{\mathbf{b}}_{ijk}, d_{ikl}\hat{\mathbf{b}}_{ikl}$ (Fig. 9, middle left). However cutting sheets of materials at an angle is difficult and we assume cuts are orthogonal to the beam quads.

This leaves us with the difficult situation to connect three line segments of length τ, yet not necessarily at equal angles. If one of the angles between the projected edges is less than π it is possible to make sure that the three beams mutually touch each other (not shown). However, if all three angles are larger than π, this is generically impossible (Fig. 9, middle right).

Therefore, we suggest to use elements with a cylindrical profile. The radius of the cylinder should be taken as the maximum of the three distances:

$$r_i = \max\{r_{ijk}, r_{ikl}, r_{ilj}\}. \tag{24}$$

This makes sure the two beams enclosing the smallest projected angle touch, while the others are not intersecting (Fig. 9, right).

It may be impractical to generate joint cylinders with different radii. Thus two simple options are to take the largest radius $\max_i r_i$ for all joints, or to consider available cylinder radii of the given material and round each r_i up to the next available radius.

The next step is to adjust the beams. We consider beam ij and adjust it for vertex i. This means adjusting the beam quad coordinates $\mathbf{v}_i^-, \mathbf{v}_i^+$. Both vertices have to be moved in direction $\hat{\mathbf{e}}_{ij}$ by r_i and we define the geometry of the normal edge of beam ij in vertex i as

$$\mathbf{v}_{ij}^{\pm} = \mathbf{v}_i^{\pm} + r_i\hat{\mathbf{e}}_{ij} \tag{25}$$

Based on this updated geometry, it is now straightforward to define the height of the cylinder in vertex i based on the averaged normal edges of the beam quads incident on i:

$$h_i = \tfrac{1}{3}\,\|\,\mathbf{v}_{ij}^{+} + \mathbf{v}_{ik}^{+} + \mathbf{v}_{il}^{+} - \mathbf{v}_{ij}^{-} - \mathbf{v}_{ik}^{-} - \mathbf{v}_{il}^{-}\,\| \tag{26}$$

Thus, the final beam shape consists of (modified) beam quads $\{(\mathbf{v}_{ij}^{\pm}, \mathbf{v}_{ji}^{\pm})\}$, having non-zero thickness τ, and cylinders with radii $\{r_i\}$ and heights $\{h_i\}$ that are attached to the vertices.

Lastly, we wish to mention that it is not necessary that the joints are rigid – it generically suffices if the beams are rigid. The geometry of the beam mesh is defined by the shape and connectivity of the beam, i. e. the mesh is rigid. However, if the pieces are rigidly joined using, e. g., glueing, soldering, or welding, it may be useful to generate temporary supports for keeping the beams at the correct angles. We suggest to cut out disks for this purpose (see Figs. 9, right). The disks could also be used to keep the structure together, as in our example made of acrylic (Fig. 7).

6.2. Layout

In the following we lay out each beam on a rectangular sheet. For this we compute bounding boxes for each beam shape. The bounding for beam ij is aligned with the edge $\mathbf{e}_{ij}$ and the height $\mathbf{h}_{ij}$.

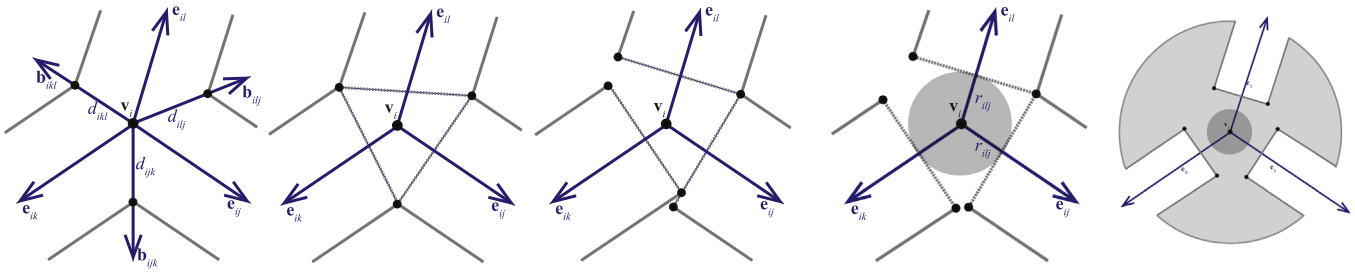

Fig. 9. Three beams meeting in a joint viewed along the direction of the normal. Left: the basic configuration. Middle left: joining the beams based on angled cuts. Middle: joining edges with orthogonal cuts potentially leaves a gap. Middle right: additional cylinder with radius chosen so that no cylinders intersect. Right: disks can be cut to support the three beams for assembly.

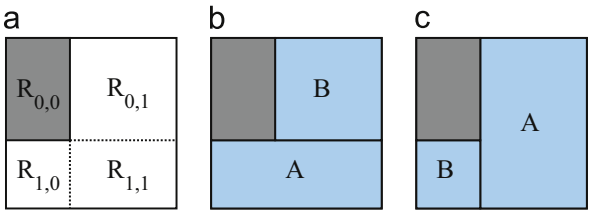

Fig. 10. Binpacking subdivision: The upper left part $\mathbf{r}^{0,0}$ of a free rectangular area $\mathbf{r}$ is allocated (a). The remaining area is subdivided into two new rectangles A, B. If $\|\mathbf{r}^{1,0}\| > \|\mathbf{r}^{0,1}\|$ then configuration (b) is used, else configuration (c).

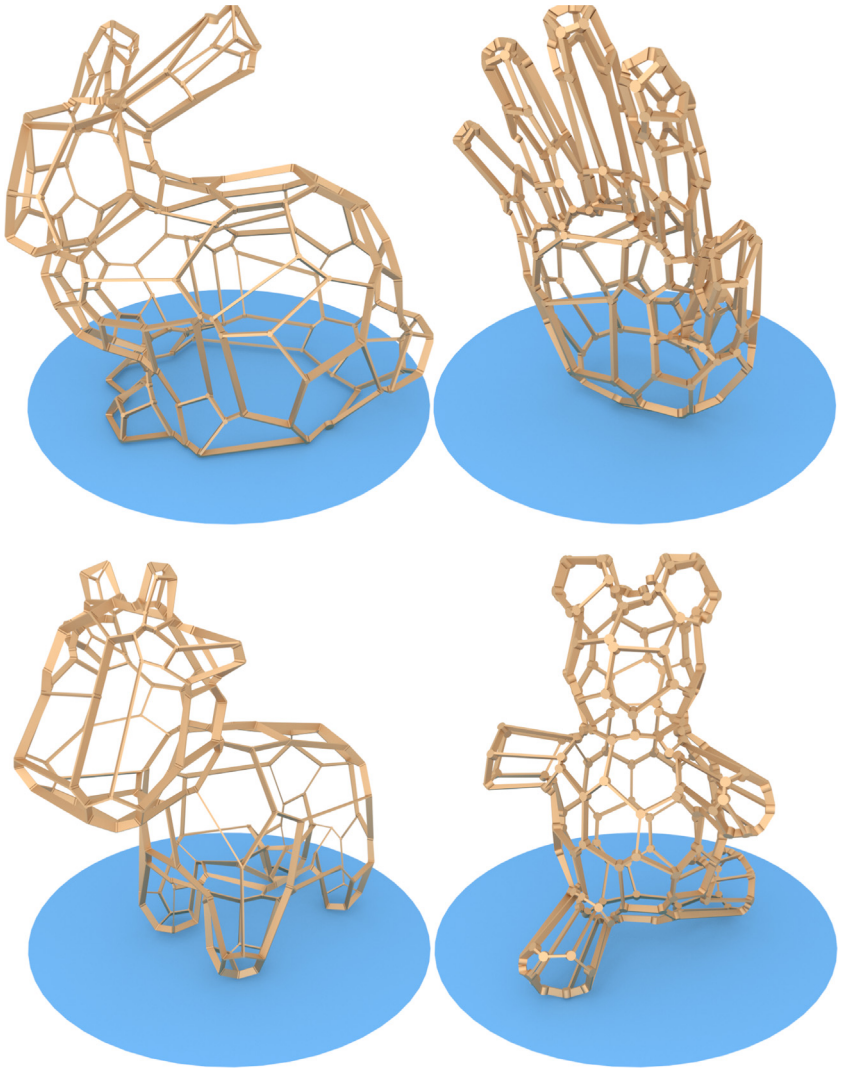

Fig. 11. Rendered beam models with 100 sites each.

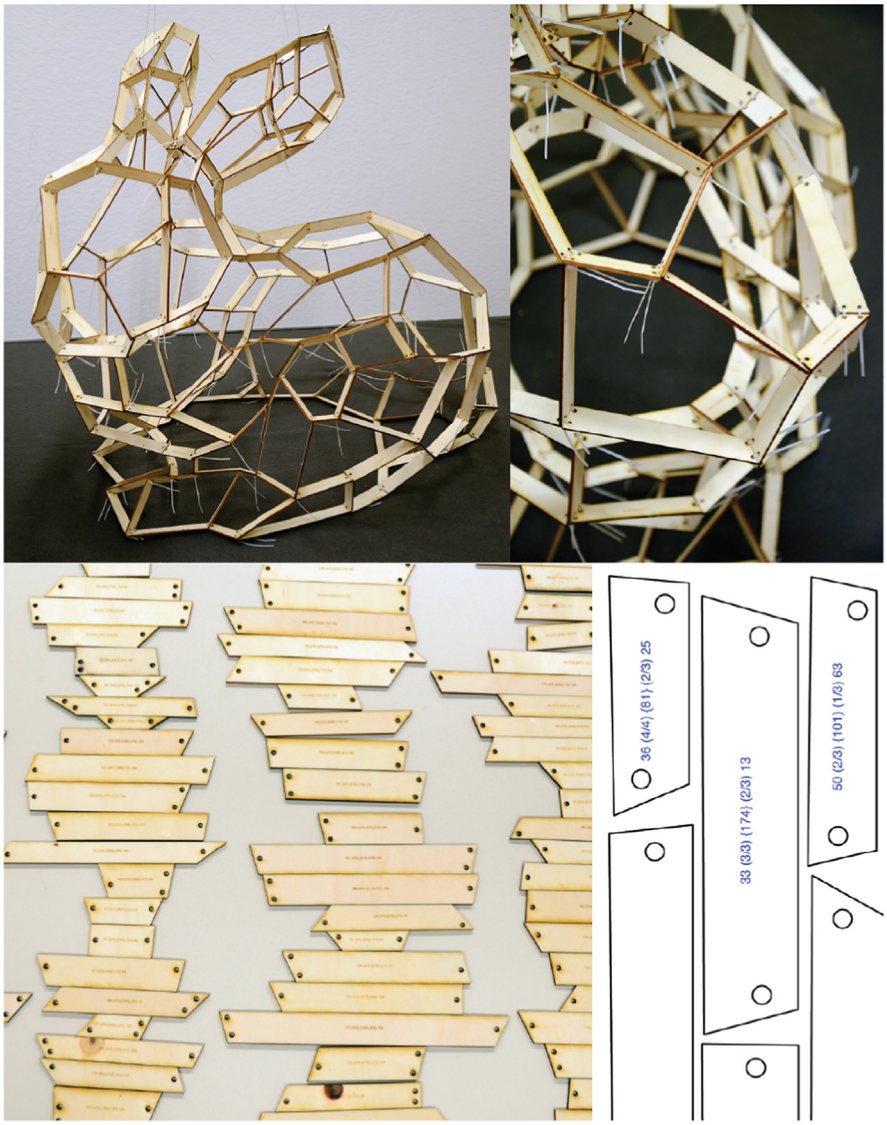

Fig. 12. Model generated and constructed with flexible joints using wooden beams.

We represent it based on the extent in the two directions as a vector $\mathbf{r}_{ij} \in \mathbb{R}^2$.

The available material is similarly represented. Let the initial sheet be $\mathbf{r}_{\max}$, of which we may have multiple pieces. For packing the bounding boxes of the beams, we use a greedy bin-packing strategy [13] that packs axis-aligned boxes in linear time using heuristics.

The available material is represented by a *FIFO* queue Q that contains axis-aligned bounding boxes, marking the (non-overlapping) unoccupied regions of the sheet. The greedy algorithm iteratively takes a box $\mathbf{r}$ from Q and searches for the largest (i. e. maximum area) unplaced beam box $\mathbf{r}_{ij}$ that can be placed inside $\mathbf{r}$. Then $\mathbf{r}$ is subdivided into three sub-boxes: the top left corner $\mathbf{r}^0$ (see Fig. 10a) is occupied by placing $\mathbf{r}_{ij}$. The remaining area $\mathbf{r}^{1,0} \cup \mathbf{r}^{0,1} \cup \mathbf{r}_{1,1}$ is split into two boxes A, B by two possible configurations (10ab). The employed heuristic targets for large boxes. Thus the largest constructible box A and the remaining box B are added to Q. In case there is no unplaced beam that fits into $\mathbf{r}$ the algorithm continues until Q is empty. The algorithm starts by initializing Q with $\mathbf{r}_{\max}$, i. e. the box defining the material size. When the algorithm ends without placing all beams, it starts again with the remaining set of beams and thus starts with a new material sheet.

Finally, we generate a vector graphics file with the (2d) shape of the beams and additional markers to identify each beam (Fig. 7, middle). We base the id's for vertices and edges on the suggested assembly order (see below).

6.3. Assembly

Although many parameters (object size, shape, material, etc.) may affect the best assembly strategy, we consider to build the model in a bottom-up fashion. For this we assume an up vector $\mathbf{u}$ is given. Then we can sort vertices based on the projection $\mathbf{u}^\top \mathbf{v}_i$. In particular, this gives us a lowest vertex (which may be not unique).

Assembly requires connecting a beam with already existing beams. This suggests to traverse the edge graph (rather than, e g, ordering edges based on the sorted vertices). We have made good experience with a *breadth-first* traversal of the graph, with the neighbor vertices sorted according to the order along $\mathbf{u}$, i. e. considering $\mathbf{u}^\top \mathbf{v}_i$.

We found it useful to print an assembly list, indicating vertex and edge id's to be put together.

7. Discussion

We have generated a variety of beam meshes from different input geometry. A set with exactly 100 faces each is shown in

Fig. 11. Computation times on a recent laptop computer: *Bunny* [63*K* faces]: 32 s, *Venus* [47*K* faces]: 22 s, *Teddy* [25*K* faces]: 12 s, *Hand* [101*K* faces]: 41 s

These computation times are negligible compared to the subsequent manufacturing procedure.

Physical realization: We have also physically constructed a few shapes – Fig. 7 already showed the Venus model made out of acrylic using the support disks as connectors. A Stanford Bunny of about 1 m height is shown in Fig. 12 (note that this geometry was created with an earlier version of our algorithm). As connectors we used simple cable ties that go through holes at the end of the beams. As we use a laser cutter to generate the beams from plywood it is easy to add the holes. We wish to stress that the construction time far exceeds the computation time: computation took a few minutes, while construction took several hours.

Limitations: We note that our approach still has limitations. Most importantly, the result of our automatic approach often requires some tuning of the parameters to yield the most aesthetic results. This is somewhat expected, as there is some randomness in the initialization, and aesthetics is no technical concept we optimize for. Still, it would be better if the process could be controlled in a more intuitive manner.

Our approach is so far purely geometric. It might be possible to also consider the static loads and adjust the mesh to better withstand its own weight. Such physical considerations would also be useful during the assembly phase, for example making sure that the structure is in balance.

References

[1] Hildebrand K, Bickel B, Alexa M. crdbrd: Shape fabrication by sliding planar slices. Comput Graph Forum 2012;31(2pt3):583–92. http://dx.doi.org/10.1111/j.1467-8659.2012.03037.x.

[2] Chen D, Sitthi-amorn P, Lan JT, Matusik W. Computing and fabricating multiplanar models. Comput Graph Forum 2013;32(2pt3):305–15. http://dx.doi.org/10.1111/cgf.12050.

[3] Schwartzburg Y, Pauly M. Fabrication-aware design with intersecting planar pieces. Comput Graph Forum 2013;32(2pt3):317–26. http://dx.doi.org/10.1111/cgf.12051.

[4] Cignoni P, Pietroni N, Malomo L, Scopigno R. Field-aligned mesh joinery. ACM Trans Graph 2014;33(1):11:1–12. http://dx.doi.org/10.1145/2537852.

[5] Fu CW, Song P, Yan X, Yang LW, Jayaraman PK, Cohen-Or D. Computational interlocking furniture assembly. ACM Trans Graph 2015;34(4):1–11. http://dx.doi.org/10.1145/2766892.

[6] Pottmann H, Liu Y, Wallner J, Bobenko A, Wang W. Geometry of multi-layer freeform structures for architecture. ACM Trans Graph 2007;26:3. http://dx.doi.org/10.1145/1276377.1276458.

[7] Du Q, Faber V, Gunzburger M. Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev 1999;41(4):637–76. http://dx.doi.org/10.1137/S0036144599352836.

[8] Du Q, Wang D. Anisotropic centroidal Voronoi tessellations and their applications. SIAM J Sci Comput 2005;26(3):737–61. http://dx.doi.org/10.1137/S1064827503428527.

[9] Richter R, Alexa M. Mahalanobis centroidal Voronoi tessellations. Comput Graph 2015;46:48–54. http://dx.doi.org/10.1016/j.cag.2014.09.009 Shape Modeling International 2014.

[10] Bishop CM. Pattern recognition and machine learning (Information science and statistics). Secaucus, NJ, USA: Springer; 2006.

[11] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. In: ACM SIGGRAPH; 2004. p. 905–14.

[12] Bouaziz S, Deuss M, Schwartzburg Y, Weise T, Pauly M. Shape-up: shaping discrete geometry with projections. Comput Graph Forum 2012;31(5):1657–67. http://dx.doi.org/10.1111/j.1467-8659.2012.03171.x.

[13] Coffman Jr EG, Garey MR, Johnson DS. Approximation algorithms for bin packing: a survey. In: Hochbaum DS, editor. Approximation algorithms for NP-hard problems. Boston, MA, USA: PWS Publishing Co.; 1997. p. 46–93.