ALGORITHMS FOR GENERATION OF IRREGULAR SPACE FRAME STRUCTURES

Franz GRUBER, Günter WALLNER

University of Applied Arts, Vienna

ABSTRACT:
Complex space frames with respect to aesthetics and stability are an important factor in contemporary architecture. Obviously there are many different ways to generate spatial structures, especially if randomness affects the generating process. In this work we present two algorithms to generate irregular space frames inside arbitrary (including non-convex) boundary volumes with predefined support areas. The resulting structures are intended as input for a genetic algorithm which optimizes the static stability. The first algorithm uses 3D-Voronoi structures as a starting point, which makes sense in terms of the frameworks load capacity. The second approach uses a repulsive force field for the calculation of curve-skeletons of three-dimensional objects.

Keywords: space frames, structures, 3D Voronoi tessellations, skeletonization, vector field

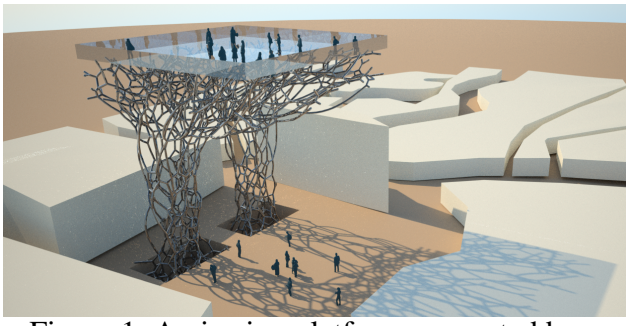

Figure 1: A viewing platform supported by a framework of Voronoi paths.

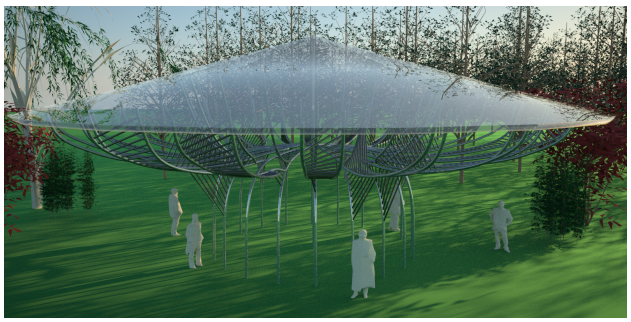

Figure 2: A pavilion were the girders where laid out with the skeletonization algorithm.

1 INTRODUCTION

The current methods of structural design are usually based on structures with a high degree of regularity (see, for example, [5] for numerous case studies). However, there is desire for irregular complex space frames in contemporary architecture. The goal of the project *Algorithmic Generation of Complex Space Frames* is to analyze new and innovative approaches to develop irregular and at the same time effective structures [3]. Part of this project was the development of algorithms to generate irregular space frames inside arbitrary (including non-convex) boundary volumes with predefined support areas. At this point we should stress that it is not our concern to generate structures in regard to statics in the first instance. Instead, the structures are intended as input for a genetic algorithm which optimizes the static stability (see Hofmann et al. [10] for a description of an early version of the system).

Space frames were independently developed by Alexander Graham Bell, who invented space-frames assembled from tetrahedral frames around 1900 and by Buckminster Fuller, who's investigations five decades later led to the creation of the famous geodesic dome. Nowadays, not only regular but also irregular space frames are becoming increasingly popular in architectural design. Notable buildings are for example the Biosphere 2 in Oracle, Arizona, the Eden

Project in the United Kingdom or the Beijing National Aquatics Center.

Obviously there are many different ways to generate spatial structures, especially if randomness affects the generating process. In this work we present two algorithms which, after different experiments, turned out to be promising. The first method uses 3D Voronoi structures as a starting point, which are known to produce statically rigid structures of space-filling tetrahedra [4]. The second method uses a repulsive force field for the calculation of the structure and was influenced by the work of Cornea et al. [7]. Figure 1 and Figure 2 show concepts renderings for buildings where the framework was calculated with the two described algorithms.

The paper at hand is structured as follows: Section 2 reviews related work in the area of irregular space frames. In Section 3 the bounding volume is discussed and Section 4 as well as 5 present the two algorithms. The paper is concluded in Section 6.

2 RELATED WORK

Because contemporary landmark architecture – as pointed out by [8] – continually moves away from economic considerations toward increasing numbers of building elements that are unique both to the individual project as well as within that particular project, research on irregular structures has increased over the past years.

For example, Kanellos [12] addressed a problem similar to ours, where a certain volume has to be filled with a structural space frame network lattice consisting of a given number of nodes. The author employs a particle-spring system where the connectivity between the particles is not predetermined but established dynamically. The system uses only local rules of inter-particle interaction so that the particles are able to generate crystal-like lattices through self-organisation.

Canzarra [4] also starts with a population of points in space but uses a model which is used in bone accretion, which mechanisms are relatively known and simple and produce structures with good static stability. As in our case, the author was not interested in finding optimal solutions but instead challenging and creative ones, as pointed out in [4]. A Delauney triangulation is used as starting structure, which is the dual of the Voronoi tessellation.

Jaworski [11] published a method to grow a structure that supports a building by providing initial seeds and the volumes to be supported. Influenced by the concept of phototropic growth, stems originating at the initial seeds grow vertically upwards and avoid obstacles and therefore entwine existing volumes. During growth the stems are connected to nearby points to ensure static stability. The resulting structures look similar, although denser, than our structures produced by algorithm (1).

Fischer [8] proposed a method to create apparently irregular structures from relatively small sets of identical parts, by combining a highly regular space-filling structure with a bottom-up generative procedure.

3 BOUNDARY VOLUME

Because the space frames have to be generated inside a given (not necessarily convex) polygonal boundary volume and multiple solutions which can be used as input for the genetic algorithm have to be generated, an efficient representation of the boundary volume is essential. In fact, the data structure has to allow fast intersections with a ray and tests if a point is inside or outside. A kd-tree [2] is therefore used as spatial data structure to allow fast traversal of the mesh for intersection tests. Furthermore, it has to be possible to define certain faces as supporting areas, in other words areas where support points of the framework can be placed.

4 VORONOI PATHS

The algorithm starts by distributing points inside the bounding box of the given boundary volume which are used as basis for a 3D Voronoi tessellation. Afterward the tessellation is cropped at the boundary volume. Then a given number of paths is traced along the cropped tessellation between two points from different supporting areas (*Voronoi paths*). Finally, the paths are smoothed

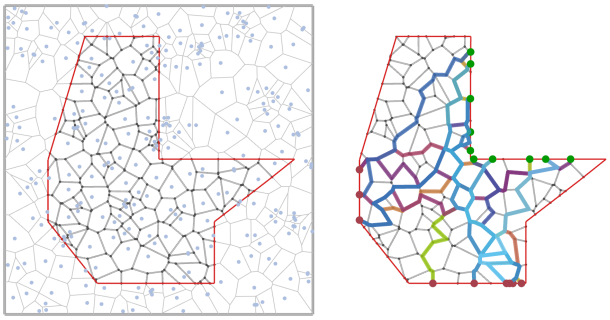

Figure 3: Left: The Voronoi tessellation is cropped at the boundary (red) which serves as *traffic system* for the path finding. Right: Boundary points (red and green dots) from different boundary areas are connected by individual paths.

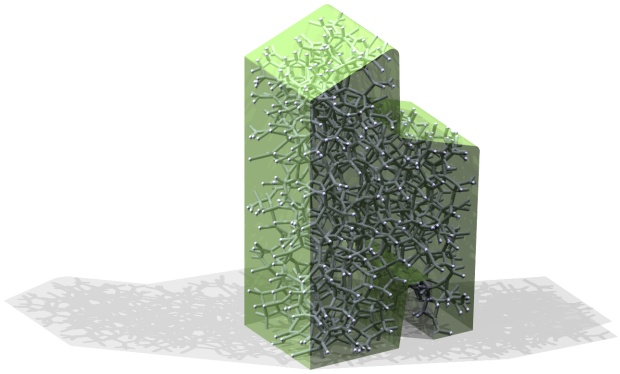

Figure 4: 3D Voronoi tessellation cropped at a non-convex boundary surface.

and yield the irregular structure. The viewing platform in Figure 1 was constructed with this algorithm. For the sake of clarity let us explain the process in two dimensions.

4.1 Preproprocessing

In the first step, points are uniformly distributed inside the bounding box of the given boundary volume, whereas the density is an important parameter for the fineness of the final framework. These points are used as generating points for the software package qhull [1] to calculate a 3D Voronoi tessellation. The results are read back and the edges are cropped at the boundary volume. The intersection points are called boundary points in the following. These steps are depicted in Figure 3 (left) and a three dimensional example is shown in Figure 4.

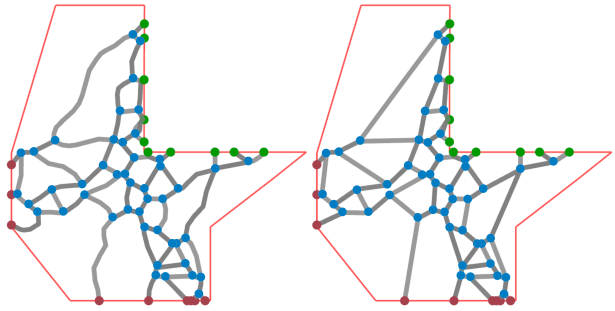

Figure 5: Left: The network from Figure 3 after a few smoothing steps. Right: A fully straightened network. In both cases cross points (blue) were fixed at their initial position.

4.2 Voronoi path finding

Obviously, the cropped tessellation does not meet the condition that its boundary points are solely located at the predefined support areas and from an aesthetical point of view it does not satisfy the required irregularity. However, we use this structure as a kind of *traffic system* to extract Voronoi paths between two randomly chosen support points p_A and p_B on different support areas. Finding a path between p_A and p_B is not a well-defined task, however, in the current implementation we try to connect these points on a preferably short way, by using the following method.

Starting at position p_A the path follows at each crossing point p_C that adjacent edge which has the smallest angle to the target direction $p_B - p_C$. This way we obtain a randomized network of crossing lines which sometimes run partially identical, bifurcate or converge (see Figure 3 (right) and Figure 6 (left) for an example in 3D). Since points and edges can occur multiple times, the network is not suitable for the following force-directed smoothing algorithm which should smooth the network as a whole and not each path separately. Therefore multiple points and edges are merged together.

4.3 Smoothing

At this point the network already has the topology of the final framework. However, the current network is characterized by zigzag lines which do not make much sense neither from a statical nor from an aesthetical point of view.

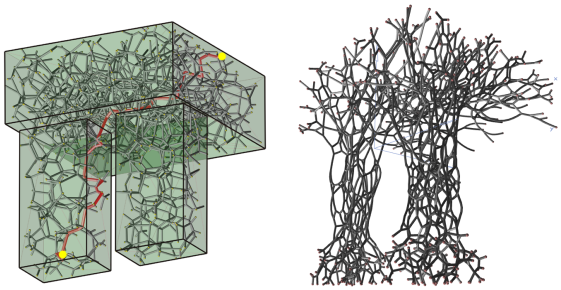

Figure 6: Left: With the help of a random 3D - Voronoi structure we define a Voronoi-Path (red) between two support points located at the top and bottom (yellow). Right: Complex structure composed of many smoothed Voronoi-Paths.

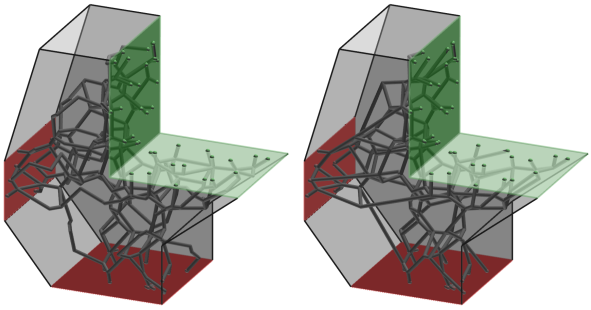

Figure 7: Left: Voronoi paths from the red to the green support area. Right: The fully straightened structure after smoothing with fixed cross points.

For this reason the network is smoothed iteratively by replacing edges with elastic springs. For each vertex v a disposition $\mathbf{v^{disp}}$ is stored which is set to zero at the beginning of each iteration. Afterward, the algorithm loops through each edge with incident vertices v_1 and v_2 and adds $\varepsilon \cdot \mathbf{d_0}$ to $\mathbf{v_1^{disp}}$, respectively subtracts it from $\mathbf{v_2^{disp}}$, where $\mathbf{d_0} = (\mathbf{v_2^{pos}} - \mathbf{v_1^{pos}}) / \left\|\mathbf{v_2^{pos}} - \mathbf{v_1^{pos}}\right\|$ and ε is a small constant. At the end of each iteration the disposition $\mathbf{v^{disp}}$ of each vertex is added to its position $\mathbf{v^{pos}}$. For boundary points the last step is omitted to keep their initial positions fixed. Conventional space frame structures feature completely straight girders and can be best achieved by additionally locking the position of cross points.
In general, the higher the number of iterations, the more straight the network will become. Figure 5 compares a slightly smoothed with a fully straightened network. Further three dimensional examples are shown in Figure 6 (right) and Figure 7.

5 SKELETONIZATION

The second approach follows the work of Cornea et al. [7], who use a repulsive force field for the calculation of curve-skeletons of three-dimensional objects. Although the connection to architecture might not seem obvious at first, because their research was originally targeted to areas like virtual colonoscopy or animation, we found it appropriate for our purposes. The method uses a generalized potential field [6] to generate a discretized vector field inside an object by charging the object's boundary.

5.1 Outline

The algorithm starts by distributing point charges on the surfaces of the boundary volume. Afterward a discretized vector field is calculated within the surface's bounding box. Although this would not be strictly necessary, it accelerates the numerical integration later in the process and eases the computation of critical points. Once the vector field is established, the particle trajectories, starting from the supporting areas toward a critical point, are calculated with numerical integration. Because these paths are running partially parallel they are merged together to avoid unaesthetic clutter. Finally, cross-links depending on an angle-threshold are inserted into the existing space frame structure. Figure 8 illustrates these steps.

5.2 Preprocessing

After the boundary volume has been defined n points $p_1 ... p_n$ with charges $q_1 ... q_n$ are placed in a small distance orthogonal to the surfaces, as opposed to Cornea et al. [7] who place them at the center of cells which are intersected by the volume. This is necessary in our case because the particles start at the boundary surfaces and should not move outside the boundary volume right away. The point charges are placed at the vertices of the the bounding volume and on the center between the barycenter v_{bc} and each triangle vertex v_1, v_2, v_3. The placement is re-

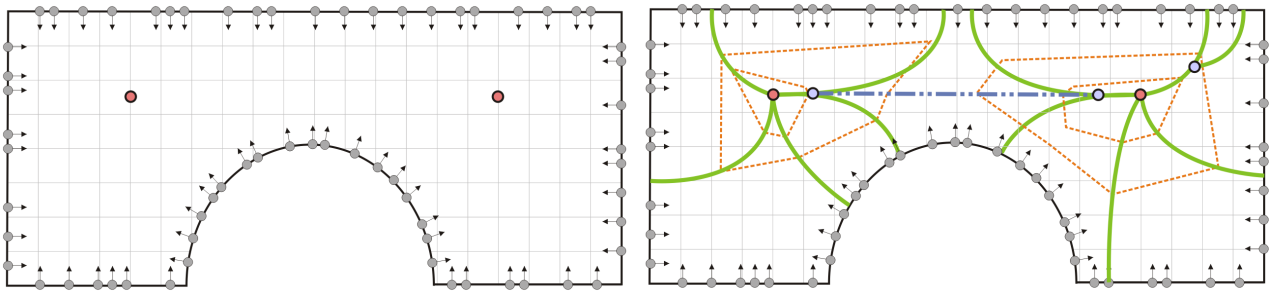

Figure 8: Left: After point charges (circles with arrows) have been uniformly distributed on the object's boundary, a discretized vector field (shown schematically as gray grid) is derived which at least contains one critical point (red circles). Right: The trajectories of particles (green) which originate on the object's boundary are the core of the final framework. Separate components are joined by straight lines (blue, dot-dashed) which connect the two closest crossings (blue circles) to ensure that the entire structure is a coherent whole. Cross-links (orange, dashed) are inserted to ensure better stability.

peated recursively for each triangle (v_{bc}, v_1, v_2), (v_{bc}, v_2, v_3) and (v_{bc}, v_3, v_1) until a given subdivision level is reached.

Based on the bounding box of the volume a discretized vector field is constructed where the number of division in each direction depends on the associated side length. At each cell center a vector

$$\mathbf{v} = c \cdot \sum_{i=0}^{n} \left(\frac{q_i}{\|\mathbf{d}_i\|^m} \right) \cdot \mathbf{d}_i \qquad (1)$$

is calculated, where $\mathbf{d_i}$ is the vector pointing from the cell center to the position of point charge i. m is a quantity for the descent of the vector field and c a small constant.

In a discretized vector field a critical point may only occur in cells where all three components of $\mathbf{v}$ pass through 0, which in case of trilinear interpolation can be found with a simple heuristic as described in [9]. If for each component of the force vector at each cell vertex both negative and positive values exist then the components must change sign somewhere inside the cell and the cell is a potential candidate for containing a critical point. If the condition is fulfilled then the cell is recursively subdivided and the test is repeated for each sub-cell until either the test fails or a maximum number of subdivisions is reached. As pointed out by Globus et al. [9] this is only a necessary condition and the cell must not contain a critical point. They therefore use Newton's method to better estimate the location of the critical point after a fixed number of subdivisions. However, in our case the exact location of the critical point is not necessary and incorrect classifications do not interfere with the correct working of the algorithm. Locating the critical points is the most time consuming step in the preprocessing step. However, the pre-process must only be performed once and does not need to be repeated to generate different space frames for a given boundary volume. Figure 8 (left) shows the status of the system after the pre-process is finished.

5.3 Particle Trajectories

Once the pre-process has finished, paths through the vector field are traced which build the foundation of the final framework. This process starts by placing particles randomly on the supporting areas. For each particle the trajectory is calculated by explicit Euler integration[1]. Because all particle trajectories have to end at a critical point (or to be more specific at an attracting node), and there must be at least one critical point in the closed boundary volume, the integration is

[1] More precise integration schemes, like Runge Kutta 4th order integration can also be used but the additional effort may not be necessary because the accurate path is not required for the matter in hand.

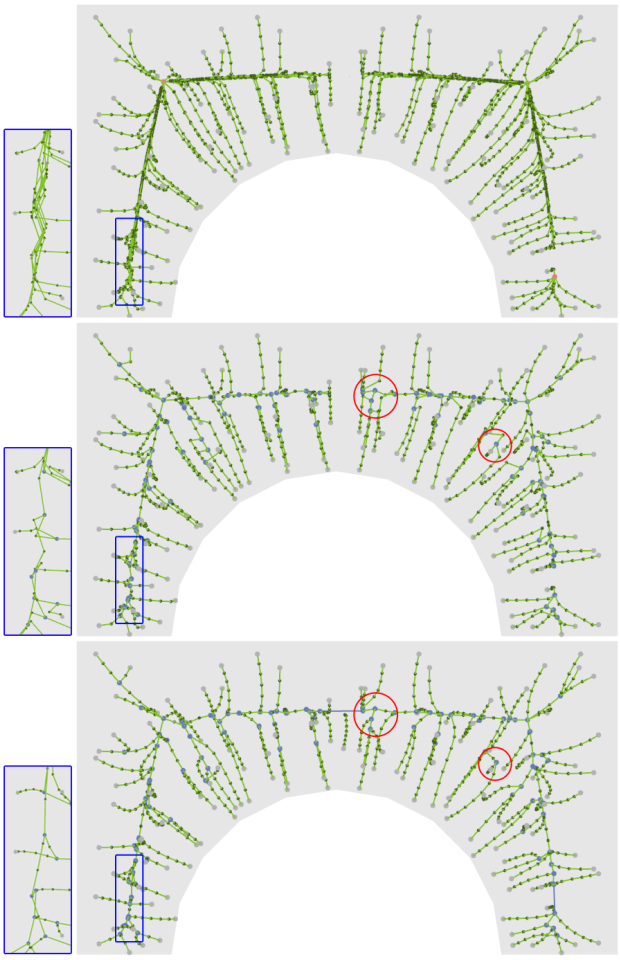

Figure 9: Top: Calculating each particle trajectory independently from each other results in clutter, since paths frequently run partially parallel to each other. Middle: The same example after merging the paths during integration which circumvents the cluttering. Bottom: The result after the components have been connected and the paths have been smoothed. The red circles show two areas where unaesthetic spikes have been removed by the smoothing algorithm.

aborted if the last position is in proximity of such a critical point. This point is added as the last point to the particle trail. These paths are shown in green in Figure 8 (right).

Because the time step Δt for numerical integration is usually relatively small it is not practical to add each position to the final path. Therefore a minimum distance $\varepsilon_d > \Delta t$ between two points must be fulfilled.

As shown in Figure 9 (top) these paths run frequently in parallel which makes the result look cluttered and not suitable for a space frame structure. To circumvent this problem the integration stops if the current location is in proximity of an already existing position, which eventually becomes the final point of the current path. This is implemented with a simple space partitioning scheme which divides the bounding volume into small cuboids. Each time a new position p is added to the path it is inserted into the appropriate cuboid c by mapping its coordinates to indices. Then the distances between p and all other positions of different paths in c are calculated and p will be connected with the position which is closest to it. Positions on which multiple paths converge will be called crossings henceforth and critical points are automatically considered as crossings regardless of the number of incident edges. The merged paths are shown in Figure 9 (middle).

5.4 Postprocessing

Depending on the vector field, the emerging structure may consist of multiple non-connected components. These are linked to each other by connecting the two closest crossings between them. Once a single component exists the smoothing algorithm as described in Section 4.3 is applied. Figure 9 (bottom) shows the result after the components are connected and the paths had been smoothed. Afterward cross-links – which are shown as orange slashed lines in Figure 8 (right) – are added to the structure as follows.

First, for each crossing c and every pair of adjacent branches with vertices $(u_0 = c, u_1, ..., u_m)$ and $(v_0 = c, v_1, ..., v_n)$ a cross-link is added between u_i and v_i provided that none of the following conditions is fulfilled:

1. the angle $\alpha = \angle(\overrightarrow{u_{i-1}u_i}, \overrightarrow{v_{i-1}v_i})$ is larger than a definable threshold α_{max}
2. either u_i or v_i is a crossing or a boundary point
3. the distance between u_i and v_i is larger than a definable maximal length l_{max}
4. a cross-link has already been added between

u_i and v_i

5. $m > n$ or $n > m$

The process is then repeated recursively between u_{i+1} and v_{i+1}[2] until one of the above mentioned conditions is violated.

Although the basic appearance depends on the vector field which in turn depends mostly on the geometry of the bounding volume the results can be altered to a certain degree by changing the number and starting position of the particles as well as the parameters ε_d, α_{max} and l_{max} and the size of the cuboids (used for merging). Figure 2 shows a pavilion which was constructed with this algorithm.

6 CONCLUSIONS

Among the infinite number of possibilities of generating spatial structures inside a given volume, we described two methods: one is based on Voronoi tessellation and the other on repulsive force fields.

Regarding the former we currently use a uniformly distributed point cloud to generate the Voronoi tessellation. For future work, the density of the point cloud could depend on geometric properties of the boundary volume. For example, the density could be higher in critical areas inside the volume, like constrictions or bottlenecks. Furthermore, different topologies can be generated by replacing the Voronoi tessellation with alternative patterns like a regular grid.

The main disadvantage of the latter method is that the resulting structures – despite randomly choosing the support points – look quite similar because the underlying vector field is defined by the bounding volume. If the vector field leads to statically unfeasible space frame structures then altering the support points will not have much effect and one cannot expect that the genetic algorithm will create a good solution. For example, a simple box only has one critical point and all particle trails will converge to this point. Therefore, the vector field should be disturbed by placing point charges randomly inside the volume which will influence the number and location of the critical points.

[2] u_i and v_i can each only have one successor because otherwise they would either be a crossing (two or more) or a boundary point (one) which would terminate the process.

ACKNOWLEDGMENTS

This work was supported by grant L358 of the Austrian Science Foundation (FWF).

REFERENCES

[1] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996. `http://www.qhull.org/`.

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[3] Klaus Bollinger, Arne Hofmann, and Clemens Preisinger. Algorithmic generation of complex space frames (Austrian Science Fund project), 2009.

[4] Pablo Miranda Canzarra. Self-design and ontogenetic evolution. In *Proceedings of the Generative Art International Conference*, 2001.

[5] John Chilton. *Space Grid Structures*. Architectural Press, 2000.

[6] Jen-Hi Chuang, Chi-Hao Tsai, and Min-Chi Ko. Skeletonization of three-dimensional object using generalized potential field. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1241–1251, 2000.

[7] Nicu D. Cornea, Deborah Silver, Xiaosong Yuan, and Raman Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. In *The Visual Computer*, volume 21, pages 945–955. Springer-Verlag, 2005.

[8] Thomas Fischer. Generation of apparently irregular truss structures. In *Computer*

Aided Architectural Design Futures 2005, pages 229–238, 2005.

[9] Al Globus, Creon Levit, and Tom Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 33–40, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[10] Arne Hofmann, Klaus Bollinger, and Manfred Grohmann. Generating geometry of irregular frameworks algorithmically. *Proceedings of Advances in Architectural Geometry*, pages 21–23, 2008.

[11] Przemyslaw L. Jaworski. Using simulations and artificial life algorithms to grow elements of construction. Master's thesis, University College London, 2006.

[12] Anastasios Kanellos. Topological self-organisation: Using a particle-spring system simulation to generate structural space-filling lattices. Master's thesis, University College London, 2007.

ABOUT THE AUTHORS

1. Franz Gruber studied Technical Mathematics at the Johannes Kepler University in Linz. He is currently working as Research Scientist at the University of Applied Arts in Vienna, where he did his dissertation about the *Application of classical geometric methods in architecture and computational aesthetics*. His main interests are the numerical development of algorithms in geometric applications and surface design. He can be reached by e-mail: `franz.gruber@uni-ak.ac.at`. His web site is: `www1.uni-ak.ac.at/geom/staff_fg.php`

2. Günter Wallner received his Diploma degree in Computer Science from the Technical University Vienna in 2005. He is currently a Research Assistant at the Department of Geometry (Institute for Art and Technology) at the University of Applied Arts Vienna where he finished his doctoral thesis about *GPU Enhanced Algorithms for Radiosity and Shadow Volume Rendering* in 2009. His research interests include global illumination, photorealistic rendering, GPU programming as well as procedural methods in computer graphics. He can be reached by e-mail: `wallner.guenter@uni-ak.ac.at`