FAST DELAUNAY-TRIANGULATED IMPORTANCE-SAMPLED POINT SETS

Charles Donohue, Victor Ostromoukhov
University of Montreal, Canada
ostrom@iro.umontreal.ca

Keywords: Importance Sampling, Delaunay Triangulation, Blue Noise, Penrose Tiling, Terrain Rendering, Geometry Processing.

Abstract: In this paper, we propose a novel method to generate Delaunay-triangulated point sets from a given density function in 2D. In order to accomplish this, we employ a Penrose-tiling-based importance-sampling strategy, which not only provides a good sampling point pattern with a local blue-noise distribution, but also provides a balanced base geometric structure from which we can efficiently derive a Delaunay triangulation of the underlying point set. We observe linear execution time with respect to the number of points. There are many areas in computer graphics that can benefit from our fast triangulated point set generator. Typical applications include terrain rendering, 3D geometry processing, and image compression.

1 INTRODUCTION

In this paper, we address a general problem which can be stated as follows. Given a 2D density function, we want to tessellate the plane with a triangle mesh which has the following properties: first, the vertices of the mesh should have a local density that is proportional to the input importance function at their position. Furthermore, we expect these vertices to follow a local blue-noise distribution (Ulichney, 1988; Hiller et al., 2001), which is similar to a Poisson-disk pattern. Also, we want to have the triangles of the mesh follow a Delaunay triangulation of its vertices. This constraint imposes that no vertex be contained in the circumcircle of any triangle in the mesh. Finally, we want the whole process to be fast and to scale well with the number of vertices.

This is a problem that arises in many applications in computer graphics. Often, it is desirable to have a mesh in which the triangle density can be modulated according to some importance metric, either for reasons of efficiency and/or visual quality. Although many regular mesh subdivision strategies can be used to this end, such as the quadtree or $\sqrt{3}$-subdivision (Kobbelt, 2000) schemes, these tend to permit only large, discrete steps in density, whereas we would like to have a smooth gradation. Also, the inherent regularity of these methods creates heavy alignments which are often undesirable, whereas a blue-noise distribution of the vertices would preclude such alignments. Another concern that often arises is the problem of narrow triangles, or 'slivers', which are not only inefficient, but can often lead to visual 'glitches'. This is why a Delaunay triangulation can be desirable, for it has the property of maximizing the minimum angle of the triangles. Finally, generating these meshes must be quick, in order to permit the interactive rates that are required in many computer graphics applications.

We propose a novel method of solving this problem. Our method has its foundations in the Penrose-tiling-based importance-sampling system proposed in (Ostromoukhov et al., 2004). The idea is to use a regular triangle subdivision scheme based on the Penrose tiling in order to obtain an overly dense set of initial points, and then to threshold these points against the importance function in order to obtain the required density. To break the structures and alignments in the point set, precalculated correction vectors are applied to each vertex. In our new tessellation system, we exploit the underlying hierarchical triangle subdivision scheme to build the triangulation of the resulting point

set in a very efficient manner. We also harness the distribution properties of the original sampling system in order to enforce a Delaunay constraint on the resulting triangulation, at a minimal computational cost. We observe that our system can build the desired Delaunay triangulated point sets in $O(n)$ with regards to the number of vertices and gives very fast results in practice.

There are a few instances in computer graphics literature in which this specific problem is faced, albeit indirectly. One case is in (Davoine et al., 1996), where an image is tessellated into a Delaunay triangulation with a higher density of triangles in the regions with a higher variance, which serves as a basis for fractal image compression. They use an incremental insertion approach based on weighed barycenters, which is fairly expensive and would not scale well with many vertices. In (Alliez et al., 2003), a similar problem arises in the context of 3D geometry remeshing. Their approach consists of using a weighed Lloyd relaxation process, which is very time consuming. At the other end of the spectrum are methods employed in many terrain visualization algorithms, such as in ROAM (Duchaineau et al., 1997), which rely on regular subdivision strategies in order to be very fast and tend to show strong alignments as a result, as well as only permitting large steps in triangle density.

We expect that the reader is familiar with the abundant litterature on Voronoi diagrams and Delaunay triangulations, which have been heavily studied and surveyed (Fortune, 1992; Bern and Eppstein, 1992). Non-randomized incremental insertion algorithms with unsophisticated point location can exhibit $\Theta(n^2)$ running time, where n is the number of vertices. There are more efficient algorithms that can run in $O(n \log n)$ worst case, such as Clarkson and Shor's algorithm (Clarkson and Shor, 1989), or the Shamos and Hoey's divide-and-conquer approach (Shamos and Hoey, 1975), Fortune's sweepline algorithm (Fortune, 1987), or a randomized incremental algorithm augmented with a search structure, such as in (Devillers, 1998). In some cases, when the vertices exhibit "nice" distribution properties, e.g. as defined in (Talmor, 1997), some general Delaunay triangulation algorithm may run in linear time (Dwyer, 1991). The algorithm presented in this paper belongs to the latter family of Delaunay triangulation of the *a priori* well-distributed point sets.

The rest of the paper is organized as follows. The sampling system which our method extends is briefly explained in Section 2. Our Delaunay triangulation algorithm is presented in Section 3. Results are presented in Section 4. Conclusions and future work follow in Section 5.

2 SAMPLING SYSTEM

In order to explain how our triangulation algorithm works, we must first make a brief review of the sampling system, which is based on *Fast Hierarchical Importance Sampling with Blue-Noise Properties*, as introduced in (Ostromoukhov et al., 2004). The system shall henceforth be referred to as 'Penrose-tiling-based sampling system' or simply 'sampling system'. The three basic steps that the system takes are illustrated in Figure 1.

First, an adaptive tile subdivision scheme is used to build an initial structure (see Figure 1-right). This results in a subdivision tree in which the spatial density of the leafs is modulated by the objective importance function. The subdivision rules are based on the Penrose tiling (Penrose, 1979); the tiles are all triangular ('c' to 'f' in Figure 2-left) save for a pair of infinitesimal pentagonal tiles ('a' and 'b' in Figure 2-right). This makes for a hierarchic structure that can be built only out of triangular subdivisions. Also, the subdivision rules are such that all angles are multiples of $\frac{\pi}{10}$, so the trigonometric operations can be tabulated for speed. Each tile has a number of attributes: a pair of orthogonal vectors shown in Figure 2-left, and a binary code (F-code) that can be interpreted as a number in the Fibonacci number system (Knuth, 1997) and (Graham et al., 1994). Each subdivision left-concatenates two binary symbols to the parent's F-code, according to the following scheme:

$$\mathscr{P}_{Penrose} := \begin{cases} a_* \mapsto \{b_{00*}\} \\ b_* \mapsto \{a_{00*}\} \\ c_* \mapsto \{f_{00*}, c_{10*}, a_{10*}\} \\ d_* \mapsto \{e_{00*}, d_{10*}\} \\ e_* \mapsto \{f_{00*}, c_{10*}, e_{01*}, a_{10*}\} \\ f_* \mapsto \{e_{00*}, d_{10*}, f_{01*}, a_{01*}\}, \end{cases} \quad (1)$$

where x_y means a tile of type x having F-code y. The symbol '$*$' replaces the parent's F-code of a tile before subdivision.

As they are created, the vertices of this structure are numbered using the Fibonacci number system (Knuth, 1997; Graham et al., 1994). In a process akin to digital halftoning, the numbers are used as a threshold against the importance function in order to obtain the desired local density of points. The numbering of the vertices reflects their position in the hierarchy, and the ordinal numbering of the vertices ensures a linear response of point density with regards to the importance values. We call the sampling points *active* when they are selected according to the thresholding process defined in (Ostromoukhov et al., 2004).

Finally, the system applies precalculated correction vectors to the active points. This tends to 'relax' the points with respect to their neighbors and breaks

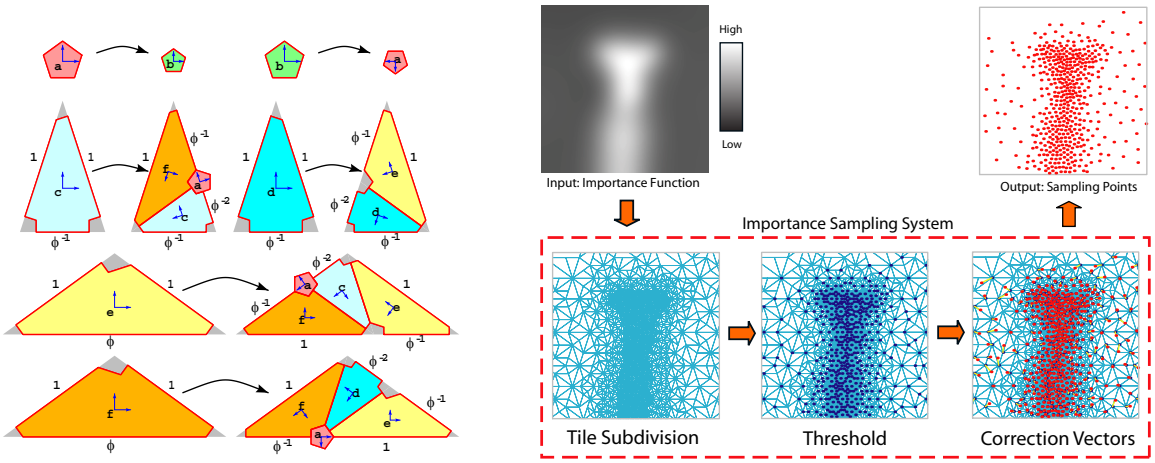

Figure 1: Left: Subdivision rules for modified Penrose tiling. The *Golden Ratio* $\phi = \frac{1+\sqrt{5}}{2} \approx 1.61803$. Pairs of orthogonal vectors form the basis for each tile. Right: Sampling system outline. *Active* (selected) points are shown as blue dots. Red dots are corrected active points. Correction vectors are shown as yellow lines that connect blue and red dots.

the inherent structures in the point set. But no proximity queries are required, as the vectors are applied to each point independently of its neighbors. The correction vectors are stored in a table, which is obtained using an offline optimization process which involves Lloyd's relaxation scheme (Lloyd, 1983). This makes the process very fast and deterministic.

We are left with a discrete sample distribution, in which the local point density is proportional to the importance density function. The resulting tree structure also determines a tringular mesh defined by the edges of the Penrose tiles. As the adaptive subdivision process operates locally, such a tringular mesh may contain T-edges.

The local distributions of points have a blue-noise spectral profile (Ulichney, 1988; Hiller et al., 2001), which equates to a low anisotropy and no principal directions or alignments. This kind of distribution can be very effective in computer graphics, especially considering the fact that the human visual perception system is particularly sensitive to such alignments.

Several existing methods can be used to generate point sets with blue-noise properties. The techniques that give good quality results, such as Lloyd's relaxation-based techniques, tend to be slow, whereas the faster techniques generally fail to meet the blue-noise requirements. Our sampler is a fast approximation, yet it is among the best in terms of quality. The possibility of generating these good distributions at such a high speed opens the door to many applications that require fast generation of high quality triangular meshes. But, as is, the system generates a cloud of points, without the connectivity information that is useful in many applications. We solve the connectivity issue in this paper.

3 Our Triangulation Algorithm

In order to extract connectivity and proximity information from a point set, it is often useful to build a Delaunay triangulation of the set. For a set S of points in the Euclidean plane, the Delaunay triangulation can be defined as the unique triangulation $DT(S)$ of S such that no point in S is inside the circumcircle of any triangle in $DT(S)$. It can also be defined as the dual of the Voronoi diagram of S, as illustrated in Figure 2. The Delaunay triangulation is the target of our algorithm, and can be built very quickly by harnessing certain intrinsic properties of the sampling system. The main ideas behind our triangulation algorithm follow.

This structure can be transformed into a proper triangulation by making sure that no T-edges remain. Since a strict set of rules is used to build this structure, it is possible to create such a triangulation in linear time with regards to the number of triangles. Second, not all vertices in the structure will be considered as *active* sampling points after the thresholding process, so these must be eliminated from the triangulation . We assume that the connectivity of the triangulation of the final point distribution will be very similar to the connectivity of the structure mentioned above. The sampler displaces the points with correction vectors, which can leave us with an invalid topology, but we suppose that this can be corrected in constant time at the local (edge) level. Finally, we can observe that the connectivity information that stems from the original structure is very close to the Delaunay connectivity after the points are displaced. After a finite number of conditional edge flips, every edge is in the Delaunay set.

Let us outline the main steps of our algorithm.

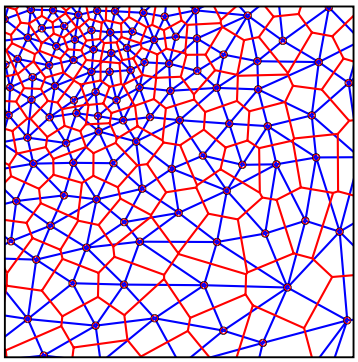

Figure 2: An example of our triangulation shown in blue, with its dual in red. These are respectively equivalent to the Delaunay triangulation and the Voronoi diagram.

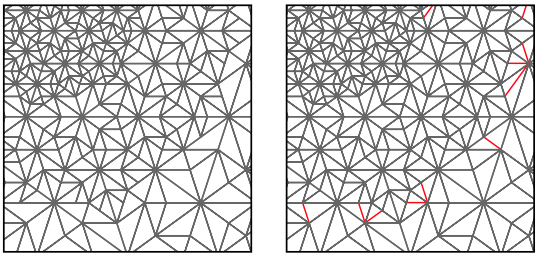

Figure 3: T-edge elimination. Before on the left, after on the right. The edges that have been added are shown in red.

3.1 Initialization

The preliminary step is the initialization of the sampler over the given importance density function, as shown in Figure 1. Instead of simply using the output points, we will use the tile subdivision tree structure which the system employs internally, in the manner described in section 3.2.

3.2 Base Triangulation

The next step is to create a *valid* triangulation from the underlying sampling structure, meaning there should not be any T-edges in the mesh. An efficient way of obtaining such a triangulation is to iterate through the tile subdivision tree of the sampler in a width-first manner; this has the effect of enforcing the following rule: no two adjacent triangles, which are slated to be subdivided at a subsequent level, will ever be at more than one level of subdivision apart, at any time during the traversal of the tree. This way, whenever a new vertex needs to be added to the current triangulation, it is assured that we only need to split two triangles along their common edge, which is a trivial operation. Also, on the borders of areas

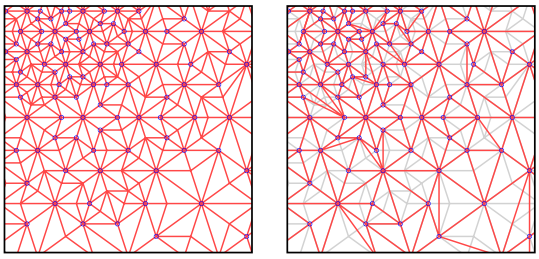

Figure 4: Inactive vertex extraction. Before on the left, after on the right. Blue vertices have passed the thresholding process.

at different levels of subdivision, the triangulation remains valid because the triangles on both sides are split. This holds true no matter how many levels of subdivision this border jumps. An example of this operation is shown in Figure 3.

3.3 Inactive Vertex Removal

The triangulation obtained at this point includes every vertex in the tiling. The second step consists in the extraction of the *inactive* vertices, the potential sampling points that have failed the thresholding step. This process is fairly straightforward; we iterate through all the 'sampling' tiles, and those that have failed the thresholding test are marked for extraction. Finding these vertices in the triangulation is simple, because we have stored reference to the latter. The removal of a vertex from the triangulation involves the re-triangulation of the hole it generates, which can be done with or without enforcing a Delaunay constraint. We have opted for a simple greedy re-triangulation because the end result is the same, while less computationally intensive because we avoid the circumcircle tests. When this greedy approach is chosen, special attention must be brought to collinear points in the re-triangulated area. The original Penrose tiling has alignments in the 10 principal directions but, depending on the numerical precision chosen for the point representation, some collinear points might appear slightly non-collinear, which can result in triangle slivers. These triangles have an unstable orientation, and can be problematic for the predicates used in further operations on the triangulation. Fortunately, a simple collinearity test avoids these situations, using a numerical precision based on the level of subdivision at the offending point.

This decimated triangulation will serve as the foundation for our final triangulation. An example of this step is shown in Figure 4.

Our implementation uses the half-edge data structure (Eastman and Weiss, 1982) in the Computational

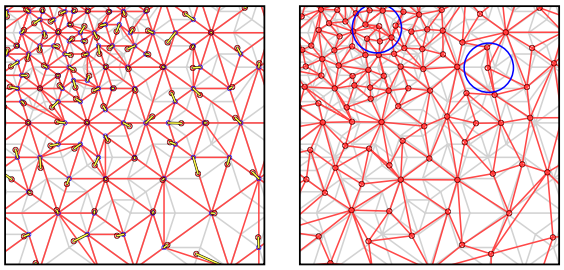

Figure 5: Sampler correction vectors. Before corrections on the left, after on the right. Notice the invalid topology of the displaced triangulation in certain areas (circled in blue).

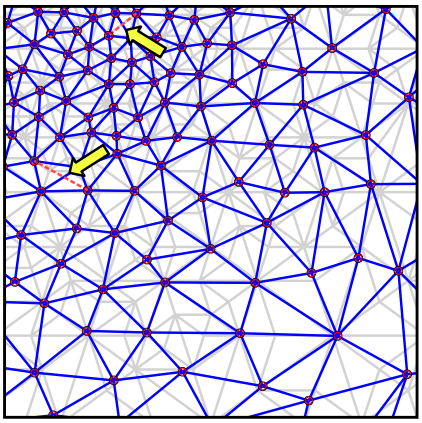

Figure 6: Comparison of our triangulation after two edge-flip operations with a Delaunay triangulation (in blue). Red edges are not in the Delaunay set. After edge-flip operations beeing performed, our triangulation is identical to Delaunay's, in this example.

Geometry Algorithms Library (CGAL) (Boissonnat et al., 2002), which supports triangle splits and edge flips in $O(1)$ time, and vertex removal in $O(d^2)$ time, where d is the degree of the vertex.

The vertices of the resulting triangulation will need to be displaced by the correction vectors provided by the sampling system; this can cause an invalid topology at certain vertices, as shown in Figure 5. This leads us to the next step in the algorithm, which is a finite number of conditional edge flips of the current triangulation. In our implementation, we use a standard edge flip algorithm (Lawson, 1972; Bern and Eppstein, 1992). In order to obtain a proper Delaunay triangulation, a certain number of these edge-flip operations must be made successively. An example of such a process is shown in Figure 6.

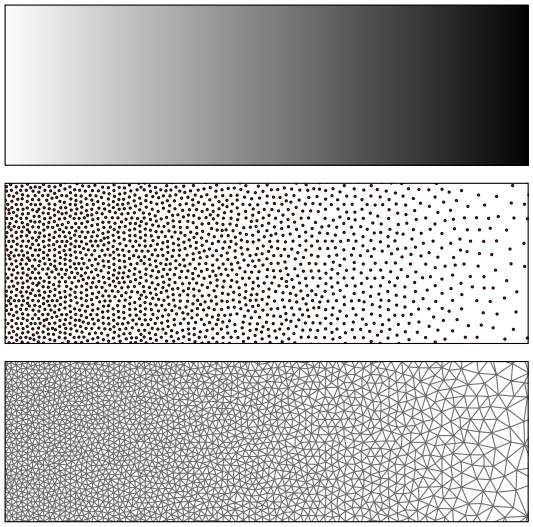

Figure 7: (Top) Gradient ramp importance density function. (Middle) Sampler output points. (Bottom) Triangulation obtained with our algorithm.

4 Results

4.1 Qualitative Results

An example of using our triangulation algorithm on a gradient ramp is shown in Figure 7. In Figure 9, a high dynamic range image is used as the importance density function. The quality of the results of our triangulation are intrinsically tied to the quality of the Delaunay triangulation. Whether this is a 'good' triangulation or not depends of course on the application, but the fact that the Delaunay triangulation maximizes the minimum angles of the triangles gives it many useful properties. Obviously, the quality of the triangulation is also tied to the quality of the distribution of the points generated by the sampler. Given that the points follow a local blue-noise (or Poisson-disk) distribution, the dual of the triangulation, called the Voronoi diagram (see Figure 2), is very close to what is called a centroidal Voronoi tessellation, which confers on it some interesting properties (Du et al., 1999).

4.2 Case Study: Terrain Rendering

In order to illustrate the potential of our system in the field of computer graphics, we present a simple-use case, the rendering of terrain maps. Once reserved for high-end flight simulators and scientific visualization, the rendering of large terrain maps is now a mainstream computer-graphics task, with a prominent place in video games. Because of the sheer size of

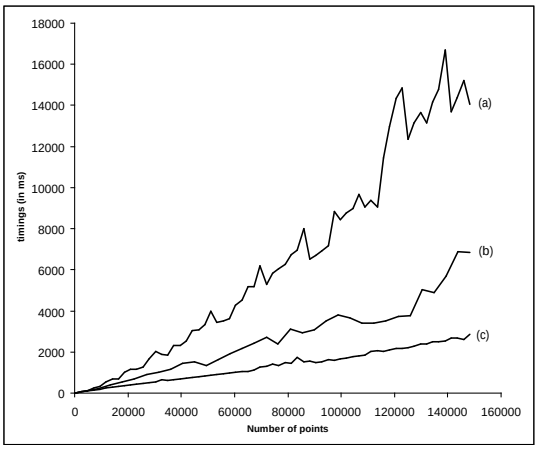

Figure 8: Performance comparison: (a) Incremental insertion. (b) Devillers' algorithm. (c) Our algorithm. The source function is a non-trivial HDR image. Times are on a P4, 2.6 GHz system.

these data sets, the challenge is to quickly generate a set of triangles that can efficiently portray the landscape, given its topological features and the position of the observer. The goal is to obtain better-quality images out of the triangle budget imposed by the rendering sub-system. An outline of how our triangulation system could be exploited in this context is shown in Figure 10.

Most fast terrain rendering algorithms, such as *ROAM* (Duchaineau et al., 1997), rely on regular triangle-subdivision schemes that give rise to highly regular structures, which can be detrimental, especially when these structures are aligned with topological features in the map. Our system has the advantage of generating the vertices of the mesh in a blue-noise distribution pattern, which not only avoids any alignments, but also avoids triangle 'slivers'. Of course, there are many other issues in terrain rendering that transcend the triangulation, most of which are related to the constraints and capabilities of the underlying graphics hardware, but these are well beyond the scope of this paper. Let us simply note that our system could be used in conjunction with many recent advances in the field, such as *Geometry Clipmaps* (Losasso and Hoppe, 2004).

4.3 Quantitative Results

In order to compare our algorithm's performance with others in real-world applications, we have timed the triangulation algorithms on an increasing number of points. The results are shown in Figure 8. For fairness' sake, all three algorithms use the same triangulation data structure. Also, the implementations of the two compared algorithms are provided by the CGAL library (Boissonnat et al., 2002), known for its good performance. As the graph shows, a naive algorithm such as the incremental insertion method, is no match for our algorithm. The brute force approach, in $O(n^4)$, is not even shown because it is substantially slower than all other methods. Devillers' algorithm (Devillers, 1998) uses an efficient search structure, which gives a nearly linear performance on the point sets generated by the sampler, given their blue-noise distribution. This makes for an algorithm that performs in the same order as ours. Nevertheless, our algorithm manages to run at least twice as fast, and this is while adhering to the highest-quality standards.

5 Conclusions and Future Work

We have addressed an important problem in computer graphics, which is to generate well-distributed point sets along with their Delaunay triangulations, given an importance density function in 2D. To this end, we employ a Penrose-tiling-based importance-sampling strategy described in (Ostromoukhov et al., 2004). The sampling system provides a good sampling point distribution with blue-noise property; it also provides a base geometric structure from which we efficiently derive a Delaunay triangulation of the underlying point set.

We have observed that our algorithm runs in linear time. Even in comparison with the best known Delaunay triangulation algorithms which can run in almost linear time in the expected (average) case, we can observe a speedup by a factor of at least two.

As future work, we plan to extend the algorithm in order to generate 3D Delaunay tetrahedrizations, and possibly n-D polyhedrizations. Of course, this depends on whether an appropriate sampler will be available in such dimensions. This is a question that we are looking into. Some operations on triangulations are simple in the 2D case, but become more complex in higher dimensions.

Another extension to explore is the case of a dynamic importance function, where temporal coherence of the function could be exploited to save computation time, as opposed to simply rebuilding the triangulation at each frame. Given that our sampler is expected to return coherent point sets across frames (which is not the case in most other similar systems), the time savings could be considerable. Since the proposed method is hierarchical by construction, a lot of work could be saved between frames by exploiting the previous subdivision and trying to keep changes incremental.

Finally, we plan to develop applications of the algorithm for promising uses in computer graphics. One such application is image compression, where an image would be partitioned into a triangulation which has a local density proportional to the image complexity. Another application is isotropic remeshing of 3D surfaces, in a manner similar to (Alliez et al., 2003), which could be made more interactive with our fast system.

REFERENCES

Alliez, P., de Verdière, É. C., Devillers, O., and Isenburg, M. (2003). Isotropic surface remeshing. In *Proceedings of Shape Modeling International*, pages 49–58.

Bern, M. and Eppstein, D. (1992). Mesh generation and optimal triangulation. In Du, D.-Z. and Hwang, F., editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. World Scientific, Singapore.

Boissonnat, J.-D., Devillers, O., Pion, S., Teillaud, M., and Yvinec, M. (2002). Triangulations in CGAL. *Comput. Geom. Theory Appl.*, 22:5–19.

Clarkson, K. L. and Shor, P. W. (1989). Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4(1):387–421.

Davoine, F., Antonini, M., Chassery, J.-M., and Barlaud, M. (1996). Fractal image compression based on Delaunay triangulation and vector quantization. *IEEE Transactions on Image Processing*, 5(2).

Devillers, O. (1998). Improved incremental randomized Delaunay triangulation. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 106–115. ACM Press.

Du, Q., Faber, V., and Gunzburger, M. (1999). Centroidal Voronoi tessellations: Applications and algorithms. In *SIAM Review*, volume 41, pages 637–676. Society for Industrial and Applied Mathematics.

Duchaineau, M. A., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C., and Mineev-Weinstein, M. B. (1997). ROAMing terrain: real-time optimally adapting meshes. In *IEEE Visualization*, pages 81–88.

Dwyer, R. A. (1991). Higher-dimensional Voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, 6(4):343–367.

Eastman, C. M. and Weiss, S. F. (1982). Tree structures for high dimensionality nearest neighbor searching. *Information Systems*, 7(2):115–122.

Fortune, S. (1987). A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174.

Fortune, S. (1992). Voronoi diagrams and Delaunay triangulations. In Du, D.-Z. and Hwang, F., editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 193–233. World Scientific, Singapore.

Graham, R., Knuth, D., and Patashnik, O. (1994). *Concrete Mathematics: a Foundation for Computer Science*. Addison-Wesley, 2nd edition.

Hiller, S., Deussen, O., and Keller, A. (2001). Tiled blue noise samples. In *Proc. Vision Modeling and Visualization*, pages 265–272.

Knuth, D. (1997). *The Art of Computer Programming, Volume 1, Fundamental Algorithms*. Addison-Wesley, 3rd edition.

Kobbelt, L. (2000). sqrt(3) subdivision. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 103–112.

Lawson, C. L. (1972). Transforming triangulations. *Discrete Math.*, 3:365–372.

Lloyd, S. (1983). An optimization approach to relaxation labeling algorithms. *Image and Vision Computing*, 1(2):85–91.

Losasso, F. and Hoppe, H. (2004). Geometry clipmaps: terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776.

Ostromoukhov, V., Donohue, C., and Jodoin, P.-M. (2004). Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3). Proc. SIGGRAPH 2004.

Penrose, R. (1979). Pentaplexity, a class of non-periodic tilings of the plane. *The Mathematical Intelligencer*, 2:32–37.

Shamos, M. I. and Hoey, D. (1975). Closest-point problems. In *Proceedings of the sixteenth Annual Symposium on Foundations of Computer Science*, pages 151–162. IEEE.

Talmor, D. (1997). *Well-spaced points for numerical methods*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. Available as Technical Report CMU-CS-97-164.

Ulichney, R. A. (1988). Dithering with blue noise. *Proc. of the IEEE*, 76:56–79.

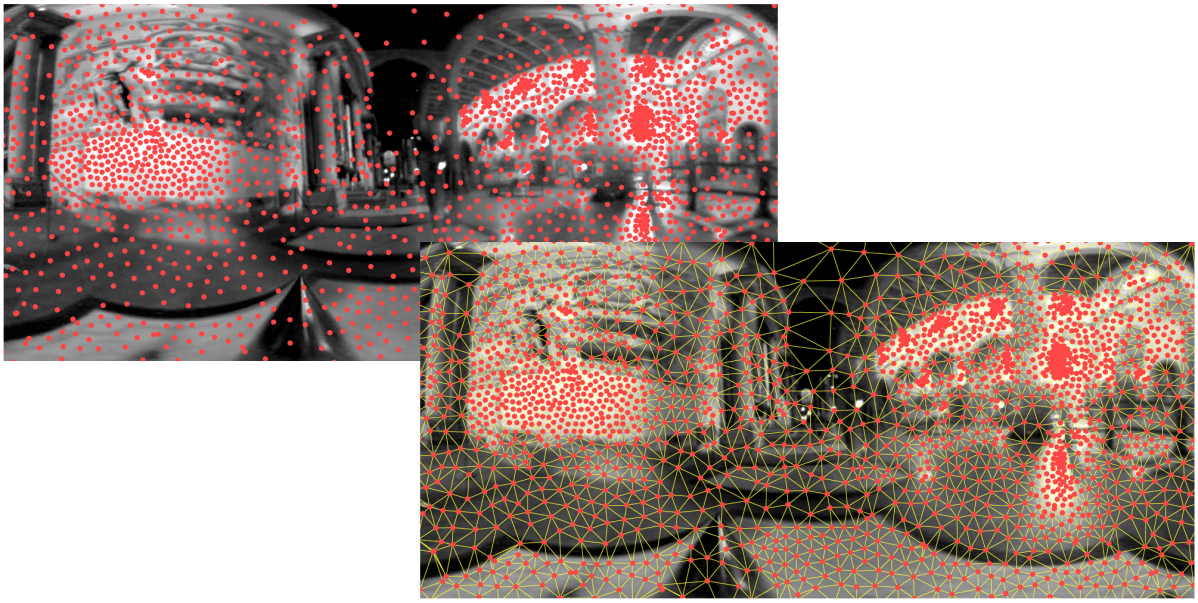

Figure 9: HDR map sampled with the Penrose-tiling-based sampling system (background image), then triangulated using our algorithm (foreground image). The total running time was 0.094 seconds on a P4 at 2.6 GHz. Running time is linearly proportionnal to the number of vertices. HDR image source: Paul Debevec.

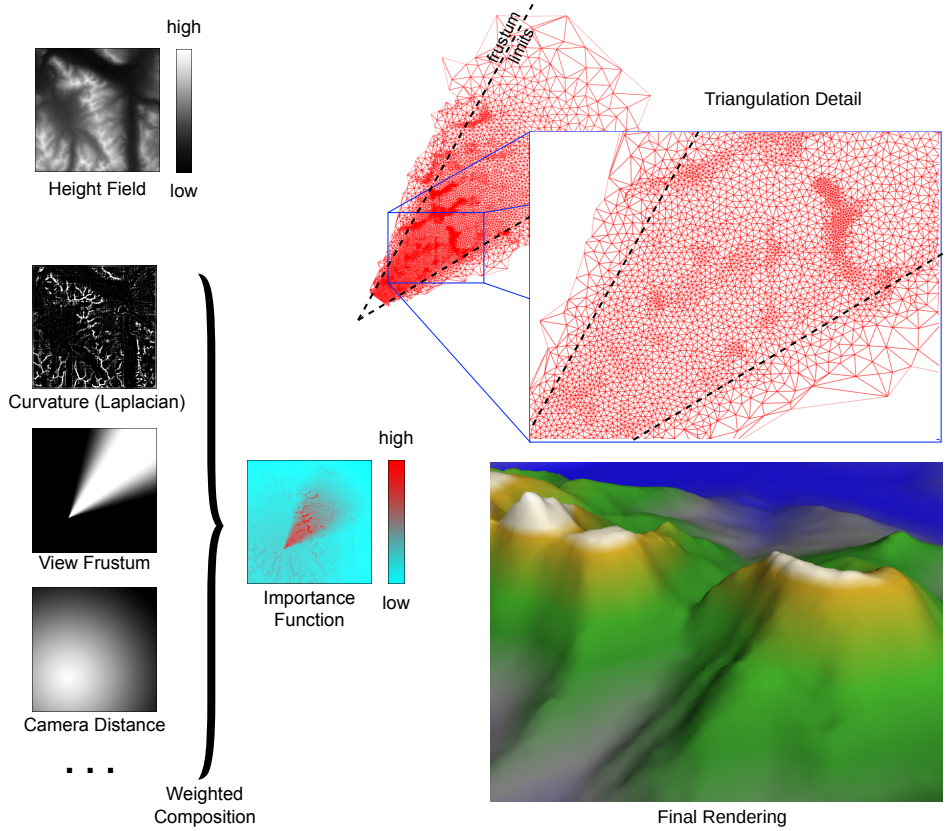

Figure 10: Terrain rendering example. (Height field data source: Yukon Dept. of Renewable Resources)