Incremental PSLG Triangulation for Surface Rendering

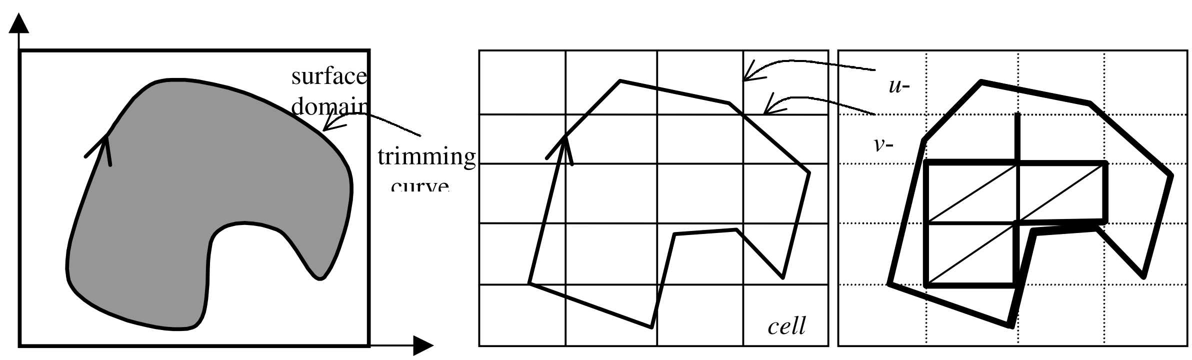
(abstract)

Subodh Kumar
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
subodh@cs.jhu.edu

Trimmed Bézier and Non-uniform Rational B-spline (NURBS) forms are widely used to represent complex models in engineering and other domains. Several techniques have been proposed for sampling and tessellation of trimmed Bézier surfaces [Kum96]. In particular, [Kum96] proposes an incremental triangulation algorithm. Surface triangulation is a popular technique for interactive display and analysis of geometric models. We present a simple and practical, yet fast incremental algorithm for triangulation of trimmed Bézier surfaces. Points on a surface are provided to this algorithm by a sampling algorithm. A planar straight-line graph (PSLG) is formed from these samples, which is then triangulated. The output of the algorithm is a set of triangle strips. The algorithm avoids long and thin triangles. In addition, it also detects if the sampling of the trimming curve forms a non-simple polygon. We have implemented the algorithm and it performs well in practice. It allows *incremental updates* and is comparatively more efficient in worst case complexity, expected complexity, and observed behavior than that used in [KM95]. In addition, in the trapezoidation step, we have incorporated *simplicity detection* at little extra cost. As a result, our implementation is more *robust*. We also avoid producing long and thin triangles. Furthermore, unlike all recent algorithms, we directly generate *triangle strips*, which are more efficient to render than triangles on current graphics systems.

Several asymptotically efficient polygon triangulation algorithms are known [CI84,Cha91,CTV89,FM84,Sei91]; our work draws inspiration from these, but is simpler to implement. Recently several researchers have shown the benefit of using Steiner points for generating triangles with good quality. For example, in two dimensions, it is possible to triangulate a PSLG using triangles with angles at most $7/8\pi$ using $O(n^2 \log n)$ Steiner points in time $O(n^2 \log^2 n)$ [Mit93]. Unfortunately these Steiner points greatly increase the number of triangles generated. Moreover, Steiner points could be added on PSLG edges. In our application such on-edge Steiner points can result in cracks [Kum96] and must be avoided. At the same time our PSLG is populated with strategically placed vertices and an enclosed grid, which helps us generate *good-quality triangles* on average without having to pay the extra cost for it. Note that while these vertices may seem to be Steiner, it can be shown that in theory the problem is still as hard as triangulating a general PSLG without introducing any Steiner points as there is no direct relationship between size of the grid and that of the PSLG.

A tensor product parametric surface, $S(u,v)$, is defined by a vector function over the domain $(u,v) = [0,1] \times [0,1]$. Optionally, a



(a) **(b)** **(c)**

Figure 1

closed sequence of trimming curve, $C_i(t)$, each defined over the domain $t = [0,1]$ may be defined on the domain of S (see Fig. 1). For brevity, we refer to the sequence as a single trimming curve, C. The trimming curve restricts the domain of S. Depending on the orientation of the curve, we either discard the part of domain of S enclosed by it or that enclosing it. In this paper, we only consider the trimming curves that comprise a single connected component (other than the regular 0-1 domain boundary). Reducing multiple components to this case follows the same principle as described in [Kum96]. Before triangulation, the surface and the trimming curves are uniformly sampled. The sampling is performed for every frame in an interactive graphics simulation. Since the sampling density for a surface does not change much in consecutive frames, it is much more efficient to modify the triangulation of the previous frame than to re-generate the whole triangulation anew. For a discussion of sampling techniques, please refer to [Kum96]. This paper produces a triangulation of these samples. In particular:

- The initial input to the algorithm is a grid of points and a closed polygonal chain (Fig. 1(b)) on the *u-v* domain. In addition, an update involving an addition or deletion of one of the grid-lines or a sample on the curve may be performed. While multiple changes on a surface can be simultaneously handled, it is easier to describe the algorithm in terms of a single update at a time.
- The output is a triangulation such that no triangles lie outside the domain restricted by the trimming curve's tessellation. In addition, no edge of the triangles generated may be greater than twice the maximum distance between adjacent input samples.
- Long and thin triangles are undesirable for smooth shading. However, since the input to our triangulation routine is generated by a uniform sampling, the points are usually well distributed. Thus, an exact Delaunay triangulation is not necessary. Using simple heuristics, we are able to obtain fat triangles in practice.
- Since triangle strips can be displayed much more efficiently than a list of triangles, the output of our algorithm is a set of triangle-strips. We follow the OpenGL convention for these strips. We avoid triangle fans, since fans tend to produce higher degree vertices in the triangulation. We keep the number of vertices adjacent to a given vertex in the triangulation small in order to lower the cost of incremental updates to the triangulation.
- No Steiner points may be added on the trimming polygon, since doing it independently of adjacent surfaces can lead to cracks in the tessellation of the model. Additionally, we do not introduce Steiner points elsewhere on the domain either, thus saving the extra triangles and keeping the point location query simple.
- A sparse sampling of the curve can produce non-simple polygons. While uncommon, non-simple polygons do occur and can cause most triangulation algorithms fail. Instead of assuming that the input polygon is simple (and thus inducing high sampling density), we check for such cases at little additional cost and thence make the polygon simple by increasing the sampling when necessary.

In our application domain, a significant number of instances of the algorithm are executed every frame. The size of each instance is relatively small. Thousands of surfaces are re-triangulated per frame. Hence, in addition to simplicity and robustness, the constants of complexity of the triangulation algorithm are crucial to interactive performance. The space requirement per surface is not quite critical, but it is prohibitive to maintain a large data structure for each surface across all frames. To speed up point location operations (required for incremental updates), we use the natural partitioning provided by the grid sampling of the surfaces domain and discard all other auxiliary data structures, which are re-constructed every time they are needed.

References

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.* , 6:485—524, 1991.

[CI84] B. Chazelle and J. Incerpi. Triangulation and shape-complexity. *ACM Trans. Graph.* , 3(2):135—152, 1984.

[CTV89] K. Clarkson, R. E. Tarjan, and C. J. Van Wyk . A fast Las Vegas algorithm for triangulating a simple polygon. *Discrete Comput. Geom.* , 4:423—432, 1989.

[FM84] A. Fournier and D. Montuno. Triangulating simple polygons and equivalent problems. *ACM Transactions on Graphics* , 3:153—174, 1984.

[Kum96] S. Kumar. Interactive display of parametric Spline Surfaces. *Phd Dissertation*, University of North Carolina at Chapel Hill, 1996.

[Mit93] S. Mitchell. Refining a triangulation of a planar straight-line graph to eliminate large angles. *Proc.* 34th *Annual IEEE Symposium on Foundation of Computer Science (FOCS).* Pages 583—591, 1993.

[NM95] A. Narkhede and D. Manocha. Fast polygon triangulation based on Seidel's algorithm. In A. Paeth, editor, *Graphics Gems V*, Academic Press, 1995.

[Sei91] R. Seidel. A simple and fast randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry Theory & Applications* , 1(1):51—64, 1991.