Edge-Constrained Marching Triangles

Neil H. McCormick and Robert B. Fisher
Division of Informatics, The University of Edinburgh.
{nhm,rbf}@dai.ed.ac.uk

Abstract

Marching triangles is a method for producing a polygon mesh surface approximation composed of triangular facets which are approximately equilateral. This paper improves the Marching Triangles algorithm where the inputs are multiple range images of scenes. $\mathcal{C}^1$ discontinuities (fold edges) are detected and used to constrain the final triangulation, thus increasing the accuracy of the mesh at sharp edges and corners and decreasing the number of triangles with a poor aspect ratio.

1. Introduction

Efficient surface triangulation is desirable for representing and rendering three-dimensional object models such as historic sculpture [11] or machine parts to be reverse-engineered [15]. Surface triangulation is also of much interest to the computer graphics community and to those involved in terrain or seabed mapping. Tessellation of a surface may be performed using polygons of any order, but triangulation is often used because the vertices of a triangle are guaranteed to be coplanar. So what constitutes an efficient triangulation? Two conflicting considerations are *representational efficiency*, that there should be as few triangles as possible, and *representational accuracy*, that there should be as close an approximation to the original surface as possible. There is the related issue of *geometrical optimisation* of each triangle, with suggested criteria including maximising either the minimum angle or the minimum height of a vertex above its corresponding baseline [2]. For a set of vertices in two dimensions, the well-known Delaunay triangulation connects points whose Voronoi regions are adjacent, thus supposing that geometrically optimal triangles are within a *minimum containment circle*. For points in three dimensions there is an equivalent Delaunay tessellation involving tetrahedra and a minimum containment sphere criterion. The exterior faces of this tessellation are then called the Delaunay triangulation of the surface in three dimensions. This all leads to the widely-accepted idea that the ideal surface triangle should be equilateral.

Three-dimensional surface triangulation is also commonly performed using *volume-intersection* methods such as Marching Cubes [9]. These intersect the surface with cubic or tetrahedral volumetric cells, producing triangles of a non-uniform nature and thereby limiting representational efficiency. Moreover the spatial resolution of the triangles in such schemes is limited above by the size of the cell, whilst increasing the size of the cell reduces the accuracy of the resulting representation. Until recently [10] such methods have also been characterised by noise and artefacts around fold edges.

The Marching Triangles (MT) algorithm was introduced by Hilton *et al.* [7] to overcome such limitations. Starting with an initial ‘seed’ triangle, the vertices of which lie on the surface, MT adds triangles incrementally to the current frontier of the mesh. New triangles are added such that each is as close to equilateral as possible and that each conforms to a condition which is an approximation of the Delaunay minimum containment sphere. This *Delaunay surface constraint* says that if the three vertices of a proposed triangle lie on a circle, then the sphere defined by the centre and radius of this circle should not contain any other vertices which are part of the mesh [7]. This is a necessary approximation since we only have three of the four points needed to define the Delaunay minimum containment sphere uniquely and so we choose to constrain its centre to be the centroid of the triangle. An alternative to constraining the centre is to fix the radius of the Delaunay sphere and pivot the sphere around a triangle edge until it touches another point [3]. This requires prior determinination of the final mesh points and successive passes of the algorithm if point density is non-uniform, while MT in one pass determines mesh points as it progresses. Both methods have the advantage common to *advancing front* triangulations [6, 13] that they allow the triangulation of surfaces that are not closed. MT is also not limited by the sampling resolution of the mesh points. However a problem with MT occurs when it is applied to range data acquired from built environments where fold edges between planes abound. These cause the algorithm to stop at the fold edge, to cut across the fold edge or (at best) to complete the triangulation with poorly-formed thin triangles. To improve this, we show in Section 2 how to constrain the algorithm to commence from the fold edges. Our implementation of the MT algorithm is given in Section 3, where some other problems with the Marching Triangles method are solved. These include its behaviour at depth discontinuities, triangulation over small gaps in the data, and some limitations of the Delaunay surface constraint.

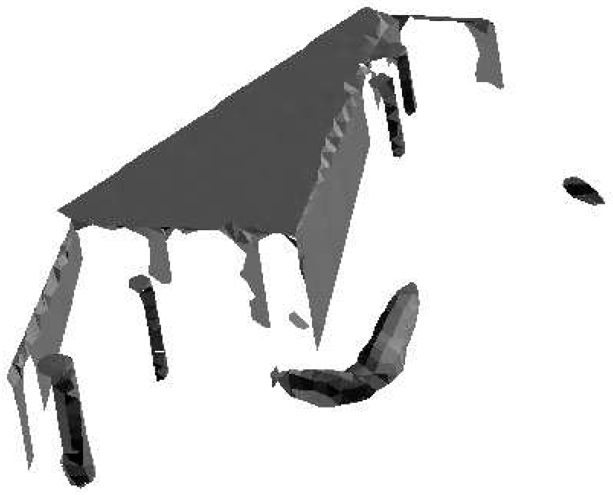

Figure 1. The basic Marching Triangles algorithm applied to data from a factory scene. Fold edges including the one in the centre of this model should be straight but have been 'cut' by some of the triangles.

2. Edge Constraints

Constraining a range image mesh to edge data has been suggested by Chen and Schmitt [4] and, in the context of environment modelling, by Wolfart *et al.* [16]. In [4], both $\mathcal{C}^0$ (depth) and $\mathcal{C}^1$ (fold edge) discontinuities are detected by fitting curves to a two-sided neighbourhood and comparing the limit or gradient of these curves respectively. In [16], a Canny-like operator is used on the range data to find depth discontinuities and local plane fitting is used to find the fold edges. Sappa and Devy [12] refine edges detected by a scan line method on a single range image, triangulate the image in two dimensions from these edges and then refine the triangulation using the three-dimensional length of each triangle side. Our approach extends these ideas in two respects. Firstly, the detected discontinuities from multiple registered range images are amalgamated to improve accuracy. Secondly, lines are fitted to the set of fold edge data. Mesh vertices may then be placed arbitrarily along these fitted lines and so need not be constrained to the original data points. This is necessary for 'seeding' the Marching Triangles algorithm by placing triangle sides along fold edges.

We are interested in real-world environmental reconstruction where the algorithm's input is a set of registered range images. In order to approximate the theoretical surface, an *initial* triangulation of each of these range images is performed by connecting adjacent points subject to a distance threshold criterion [8]. We then only need to be able to find the nearest point on each range image to any given arbitrary point p. The function to do this will be invoked during the marching triangulation to find the nearest point on each range image, the closest of which is returned. The multiple range images are thus fused seamlessly into a single mesh without the need to employ techniques such as zippering [14].

Before MT commences, we pre-segment the range images to find fold edges, then form a set of seed triangles that lie along these edges. This prevents 'corner-cutting', avoids having to explicitly navigate around 90^o-plus edges and enables placement of more-uniform triangles along fold edges. To find the range data points that lie on each fold edge we threshold the angle between each pair of triangles in the initial range image mesh. The methods of [4] or [16] could also be used, but these suffer from the same limitation as our simpler method, namely scale-dependency. The size of the neighbourhood in each method affects the fold edges detected and we have found that very detailed range images (the factory scene is 1394×2752 range measurements) may still be smooth enough at the 'obvious' fold edges whilst detecting what seem to be spurious ones. Therefore each range image is decimated to the scale of the marching triangulation and the fold edge points are re-detected. Then the fold edge points from the whole set of range images are clustered into lines using a RANSAC [5] algorithm (see Figure 2). The lines found by RANSAC are used to define a new set of mesh vertices, equally spaced as far as possible by the length of a marching triangle edge. On each line, each successive pair of points then becomes a new edge in the Marching Triangles mesh. An advantage of this method is that, in line with the MT technique as a whole, these vertices are placed with arbitrary accuracy and are not limited to range image point positions. Further, edge information from multiple range images (which individually could be noisy) is fused in a natural way.

3. The Algorithm

Once the seed mesh has been produced, the triangle edges at the frontier of the mesh are placed on a linked edge list. The algorithm itself then proceeds as follows. Each triangle edge on the list is considered in turn. For each, let p_1 and p_2 be its endpoints. Then to form a new triangle:

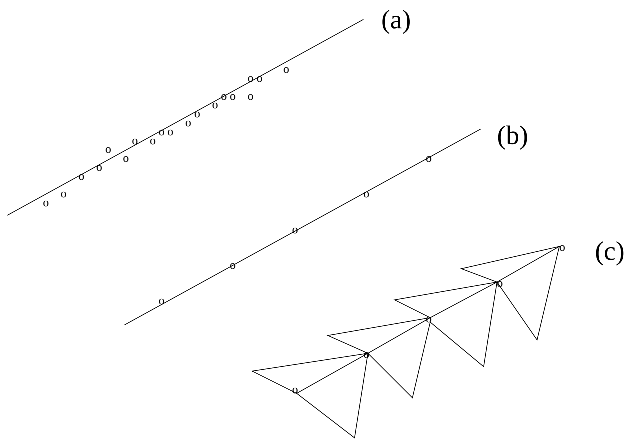

Figure 2. Triangle seeding. Lines are fitted to edge points using RANSAC (a), then mesh vertices are positioned equally along the line (b) before triangles are attached to both sides of the fold edge (c).

1. Edge $p_1 p_2$ is tested to make sure it is a frontier edge.

2. Connection from the edge $p_1 p_2$ to the endpoints of either of the two adjacent frontier edges p_{next} and p_{prev} is considered. One of the points p_{prev} or p_{next} will be connected if: (a) the point is on the opposite side of the edge from the third point p_3 of the existing triangle $p_1 p_2 p_3$; (b) the point is reasonably close to *both* points p_1 and p_2 on the current edge; and (c) the point satisfies the Delaunay surface constraint. For condition (b) we take a multiple of the normal triangle side length l (say $1.2l$) as the maximum distance the point may be from either of the current edge points, in order to prevent formation of an excessively obtuse new triangle.

3. If it is not appropriate to connect either of the two adjacent edge points, a new point is projected out from the midpoint of the frontier edge $p_1 p_2$ perpendicular to the edge and in the same plane as the triangle. This projection length is an input to the algorithm and determines the size of the triangles in the mesh. Use of an adaptive projection length is possible [1]. The third vertex of the new triangle is the nearest point p_{new} on the surface to this projected point. The new triangle is accepted if (a) it satisfies the Delaunay surface constraint and (b) neither of the two new proposed edges cross the two adjacent frontier edges $p_1 p_{\text{next}}$ and $p_2 p_{\text{prev}}$ considered in step 2 above.

4. When the Delaunay surface constraint is not satisfied in step 3 above, we check for an overlap of the new proposed point onto any existing triangle. In this case, the current edge is connected to the nearest frontier edge vertex on that triangle, where 'nearest' means nearest to the midpoint of the current edge.

Illustrations of steps 2 to 4 are contained in [7]. If none of the above steps produces a new triangle, the algorithm moves on to consider the next edge on the list. Each time a new triangle is added to the mesh, any new vertices are added to the vertex list and any new edges are added to the end of the edge list. The algorithm continues until it exhausts the edge list.

Note that steps 2 and 3 are reversed as compared with previous implementations [1, 7]. There are two reasons for this. Firstly, we wish to triangulate right to every depth discontinuity, whereas the algorithm in [7] would appear to stop the triangulation short of the surface boundary. Triangulating to the boundary normally requires non-standard shapes of triangle for complete coverage, but this can lead to a situation where the Delaunay surface constraint is not enough to prevent triangles crossing (Figure 3(a)). Secondly, we can prevent thin triangles when the proposed point is just next to an existing point which is just outside the Delaunay sphere (Figure 3(b)). Another concern is with the Delaunay surface constraint itself. Situations arise occasionally where the constraint does not prevent a point from crossing onto an existing triangle (Figure 3(c)). By considering a connection before a projection, we overcome such situations without compromising the algorithm.

We also introduce a heuristic check on the fit of each proposed triangle to the range data. This is desirable since otherwise the MT algorithm can only be guaranteed to be near the surface at mesh vertex points. This means a marching triangle may cross a small gap in the surface. Each range image is composed of a set of discrete locations $\{(i, j)\}$, with corresponding range measurement r_{ij}. We compare the number of range measurements N_r which fall within the projection $\triangle$ of the marching triangle on to the plane of the range image with the number N_e of possible locations. If the ratio N_r/N_e is too small (less than 95%) then the triangle is rejected (and a different strategy, such as a smaller projection length, may be tried).

Figure 4 shows the constrained MT mesh model of the factory scene. The central fold edge is sharper when compared with the corresponding mesh in Figure 1. The fold edge along the bottom left wall is also much improved.

4. Conclusions

The Marching Triangles algorithm is an elegant method for triangulating surfaces obtained from multiple range images, fusing the data in a straightforward manner. We have extended the method to cope with fold edges by pre-segmenting the range images to detect the fold edges and

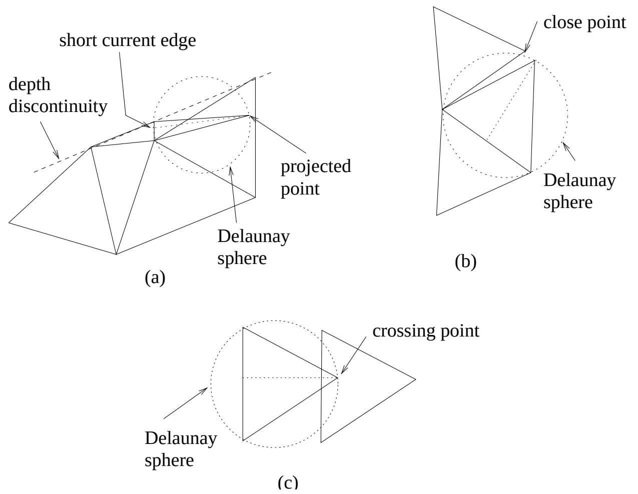

Figure 3. Problems with the Delaunay surface constraint. (a) At depth discontinuities, short edges may allow a new triangle to cross an existing one. (b) A good point to connect on an existing triangle may lie just outside the sphere. (c) Triangle crossing may occasionally occur even without short edges.

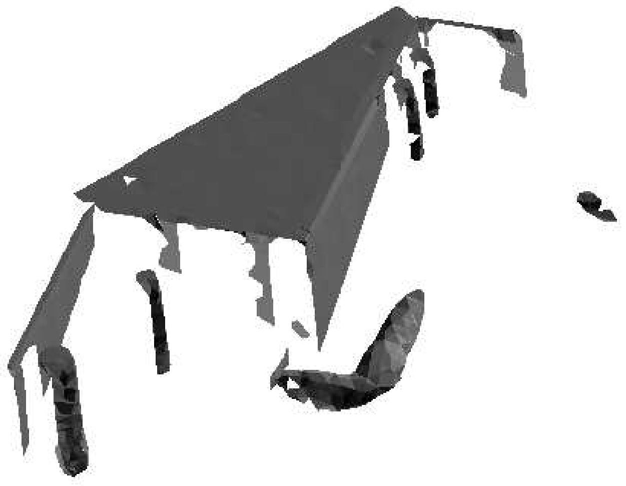

Figure 4. The constrained mesh model for the factory scene. Note how the central fold edge is straighter than in Figure 1.

then seeding the MT algorithm by triangulating from these fold edges. This prevents triangles from cutting off the fold edges and also improves the fit of the mesh to the data.

References

[1] S. Akkouche and E. Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics Forum*, 20(2):67–80, 2001.

[2] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. Technical Report CSL–92–1, XEROX Palo Alto Research Centre, March 1992.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[4] X. Chen and F. Schmitt. Surface modelling of range data by constrained triangulation. *Computer-Aided Design*, 26(8):632–645, 1994.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Technical Report 213, Artificial Intelligence Centre, SRI International, 1980.

[6] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized Delaunay triangulation. *Computer Graphics Forum*, 19(3):467–478, 2000.

[7] A. Hilton and J. Illingworth. Marching triangles: Delaunay implicit surface triangulation. Technical Report CVSSP 01, University of Surrey, January 1997.

[8] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Implicit surface-based geometric fusion. *Computer Vision and Image Understanding*, 69(3):273–291, March 1998.

[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics: Proc. SIGGRAPH*, 26(2):71–77, 1992.

[10] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H. Seidel. Feature sensitive surface extraction from volume data. *Computer Graphics: Proc. SIGGRAPH*, pages 57–66, 2001.

[11] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J.Ginsberg, J. Shade, and S. Fulk. The Digital Michelangelo Project: 3D scanning of large statues. In *Computer Graphics: Proc. SIGGRAPH*, pages 131–144, 2000.

[12] A. D. Sappa and M. Devy. Efficient contour extraction in range image segmentation for building modelling. In *Proceedings of the International Symposium on Virtual and Augmented Architecture*, pages 57–67. Springer, 2001.

[13] C. T. Silva and J. S. B. Mitchell. Greedy Cuts: an advancing front terrain triangulation algorithm. In *ACM Symposium on Geographical Information Systems*, pages 137–144, 1998.

[14] G. Turk and M. Levoy. Zippered polygon meshes from range images. *Computer Graphics: Proc. SIGGRAPH*, pages 311–318, 1994.

[15] N. Werghi, R. B. Fisher, A. Ashbrook, and C. Robertson. Object reconstruction by incorporating geometric constraints in reverse engineering. *Computer-Aided Design*, 31(6):363–399, 1999.

[16] E. Wolfart, V. Sequeira, K. Ng, S. Butterfield, J. Goncalves, and D. Hogg. Hybrid approach to the construction of triangulated 3D models of building interiors. In *Lecture Notes in Computer Science 1542: Computer Vision Systems (Proceedings ICVS '99)*, pages 489–508, 1999.