Modeling Range Images with Bounded Error Triangular Meshes without Optimization

Angel Domingo Sappa †

LAAS-CNRS †
7, Avenue du Colonel Roche
31077 Toulouse, Cedex 4, France
asappa@laas.fr

Miguel Angel García ‡

Department of Computer Science and Mathematics‡
Rovira i Virgili University
Ctra. Salou s/n. 43006 Tarragona, Spain
magarcia@etse.urv.es

Abstract

This paper presents a new technique for approximating range images by means of adaptive triangular meshes with a bounded approximation error and without applying optimization. This new approach consists of three stages. In the first stage, every pixel of the given range image is mapped to a 3D point defined in a reference frame associated with the range sensor. Then, those 3D points are mapped to a 3D curvature space. In the second stage, the points contained in this curvature space are triangulated through a 3D Delaunay algorithm, giving rise to a tetrahedronization of them. In the last stage, an iterative process starts digging the external surface of the previous tetrahedronization, removing those triangles that do not fulfill the given approximation error. In this way, successive fronts of triangular meshes are obtained in both range image space and curvature space. This iterative process is applied until a triangular mesh in the range image space fulfilling the given approximation error is obtained. Experimental results are presented.

1. Introduction

As range sensors are becoming more efficient and affordable, the use of range images for 3D computer vision applications is significantly increasing. Dense range images are highly redundant representations in the sense that, for instance, large planar areas can occupy thousands of pixels in the image. In order to accelerate further processing, range images can be approximated by more efficient representations, such as triangular meshes [1]. Subsequent processing can then be done at a higher abstraction level in the geometric domain, dealing with triangles instead of with individual pixels (e.g., [2], [3]).

In order to obtain triangular approximations with a predefined bounded error, two main approaches have been proposed. The first approach (*fine-to-coarse*) starts with a dense triangular mesh containing all the points of the original range image and, at each iteration, removes the point that introduces the lowest error (e.g., [4][5]). The second approach (*coarse-to-fine*) starts with a few triangles and, at each iteration, adds the point that produces the largest error reduction (e.g., [6]). Since optimization is applied at each iteration, those methods become very costly when they are applied to large range images.

A fine-to-coarse algorithm that does not apply optimization at each iteration and ensures a bounded error was proposed in [7]. This technique starts with the dense triangular mesh that contains all the original points in the image and successively removes points until the resulting surface intersects with one of two offset surfaces which are initially defined above and below the original triangular mesh, at a distance of half the maximum allowed error.

This paper presents a coarse-to-fine technique for generating an adaptive triangular mesh with a bounded approximation error without applying optimization. The proposed technique is described in section 2. Section 3 presents experimental results and finally, conclusions and further improvements are given in section 4.

2. Generation of triangular meshes with bounded approximation error

The proposed algorithm computes an adaptive triangular mesh **M** that approximates a given range image **R** such that the approximation error (tolerance) between **M** and **R** is below a specified threshold ξ.

The algorithm consists of three stages. In the first stage, the original range image is mapped to a 3D *image space* whose reference frame is attached to the range sensor. Thus, each pixel of the range image is converted to a 3D point **P**. The set of all those 3D points will be referred to as the *3D range image*. Afterwards, each point **P** is associated with a curvature value obtained by estimating the surface that would pass through that point in the 3D range image. Taking these curvatures into account, all the points **P** are mapped to a 3D *curvature space*.

The second stage of the algorithm triangulates the points contained in the curvature space through a 3D

0-7695-0750-6/00 $10.00 © 2000 IEEE

Delaunay algorithm. The effect accomplished by applying the 3D triangulation in curvature space instead of in the 3D image space is that the coarsest (outest) triangles in the tetrahedronization tend to join distinctive points (points with high curvature), which define the shape of the objects being approximated.

Finally, in the third stage, an iterative process starts digging the triangular mesh that constitutes the external surface of the previous tetrahedronization, removing those triangles that do not fulfill the approximation error. Each removed triangle is substituted for the three triangles that form its corresponding tetrahedron. In this way, the external mesh of the tetrahedronization is successively refined until all its triangles satisfy the approximation error. These stages are further described below.

2.1. Mapping to a 3D curvature space

A range image is a rectangular sampling of a scene surface. Its usual representation is a two dimensional array **R**, where each element $\mathbf{R}(r, c)$ is a scalar that represents a 3D point **P** of coordinates: $\mathbf{P} = (f_x(c), f_y(r), f_z(\mathbf{R}(r, c)))$ referred to a local reference frame. The set of points $\mathbf{P} = (x, y, z)$ defined above constitute the 3D range image. The *viewing direction* is aligned with the Z axis of the local reference frame and it is pointing downwards: $\mathbf{D} = (0, 0, -1)$.

The surfaces of the objects contained in the range image can be approximated through a trivial triangulation of the points in the 3D range image, by joining them along the rows, columns and diagonals associated with their corresponding pixels in the range image. This triangular mesh will be referred to as the *original triangular mesh.*

For each point $\mathbf{P} = (x, y, z)$ belonging to the 3D range image, an estimation of its curvature is computed as $\mathbf{K} = \left|8z - \sum z_i\right|$, where z_i denotes the z coordinate of each of the eight neighbors of **P**.

After obtaining the 3D range image and computing all the curvatures, each point **P** associated with $\mathbf{R}(r, c)$ is mapped to a point $\mathbf{P}' = (x', y', z')$ in the 3D curvature space as: $x' = \alpha r$, $y' = \alpha c$, $z' = \log(1 + \Re + \mathbf{K}^2)$, with $\Re$ being a random real number ranging between zero and one, and α being a constant necessary to prevent degenerated tetrahedra due to precision errors. The random component is necessary to avoid degenerated tetrahedra in areas of constant curvature, in which all the points tend to lie on the same plane when they are considered in the curvature space.

2.2. 3D Delaunay triangulation of points in curvature space

At this stage, the points in the curvature space are triangulated by using a 3D Delaunay algorithm that produces a list of tetrahedra. Each of the four triangles that form every tetrahedron is defined by three identifiers, (t_0, t_1, t_2), which represent three differents points. In order to speed-up further operations, all the triangles obtained above are inserted into a *hash table.*

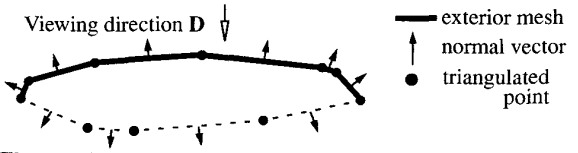

Figure 1. 2D section of the convex-hull and the exterior mesh associated with it for a set of points represented in the 3D curvature space.

A triangle **T** belongs to two tetrahedra at most. Thus, it has associated two opposite points, (op_1, op_2), which constitute the apices of those tetrahedra. As a particular case, triangles that belong to the external mesh of the tetrahedronization are only contained in one tetrahedron. Those triangles are referred to as *exterior triangles.* After applying the 3D Delaunay triangulation, the set of all the exterior triangles constitutes the convex-hull of the tetrahedronization.

An exterior triangle $\mathbf{T} = (t_0, t_1, t_2)$ has associated a unitary normal vector **N** obtained as the cross-product of the 3D points that form **T**, reoriented in such a way that it is pointing out of the positive half-space determined by the plane of **T**. The positive half-space is the half-space delimited by the plane containing **T** and which does not contain the vertex opposite to **T**. The latter vertex is the apex of the tetrahedron whose base is **T**. Since exterior triangles only belong to one tetrahedron, it is very efficient to determine whether a certain triangle is exterior or not given the identifiers of its vertices.

As we are dealing with range images obtained from a certain viewing direction **D**, only those exterior triangles whose normal vectors have an angle with respect to **D** higher than 90 degrees belong to the surfaces of the objects present in the image. The exterior triangles that fulfil the previous condition form the *exterior mesh* (Fig. 1).

2.3. Digging process

Let **M** be the exterior mesh obtained as the union of the exterior triangles determined as described in Section 2.2. We will consider that this mesh is validated and, therefore, that a valid solution to the approximation problem has been found, when all its triangles, considered in the image space, have an approximation error below or equal to the given error ξ.

The iterative digging process goes over all the triangles of the exterior mesh **M**. If a triangle has some of its control points at a distance farther than ξ, this triangle is marked for removal. After all triangles have been considered, each triangle marked for removal is substituted for the other three triangles that belong to its tetrahedron. Thus, a new exterior mesh **M** is obtained and the process is repeated. These two steps are detailed below.

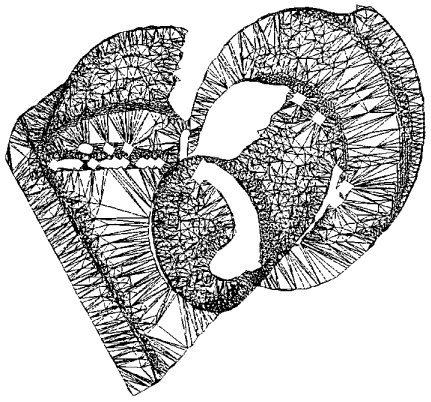

Figure 2. Final exterior mesh in image space containing 6,787 triangles obtained after 31 digging iterations (original image contains 26,028 triangles).

2.3.1. Triangle validation. The validation process is carried out in the image space and consists of measuring the approximation error associated with each triangle. The approximation error is the maximum distance, measured along the viewing direction D, among a triangle and its control points. The control points utilized to validate a triangle **T** are those points of the 3D range image whose projection, along the viewing direction, is contained in **T**.

At the beginning of the digging process, all the triangles of the exterior mesh are invalid. During the digging process, only invalid triangles are considered.

The validation of an invalid exterior triangle **T** consists of verifying that the set of 3D range image points **P** that project onto that triangle are located at a distance from **T** below or equal to ξ. If any of these control points are located at a distance from **T** farther than ξ, the triangle **T** is left as invalid. That triangle will be removed after all the current exterior triangles have been validated.

On the other hand, if all the control points that project onto **T** are located at a distance below or equal to ξ, **T** is labeled as valid. This implies that **T** belongs to the final solution and it will not be considered during the following iterations of the digging process.

2.3.2. Invalid triangle removal. At this point, the triangles of the current exterior mesh M are valid or invalid. If all triangles are valid, M is already a solution to the problem and the algorithm concludes (Fig. 2). Otherwise, all the invalid triangles are removed.

Given an invalid exterior triangle **T** with indices (t_0, t_1, t_2) (see Section 2.2.), the identifier of its opposite point op_i is extracted from the hash table. That identifier corresponds to the apex of the tetrahedron that contains **T**. Triangle **T** is then substituted in the exterior mesh for the other three triangles that form its tetrahedron. If any of those new triangles was already contained in the exterior mesh, that new triangle is not inserted and the existing one removed from the mesh. The new inserted triangles are labeled as invalid.

If after having removed all the invalid triangles, no new exterior triangles have been included in the exterior mesh **M**, the latter is already a solution to the problem and the algorithm concludes. Otherwise, if new triangles have been included, the digging process starts over from stage 2.3.1..

3. Experimental results

The proposed technique has been tested with different real range images. Fig. 2 shows a final triangular mesh obtained after 31 digging iterations. This mesh approximates the original range image by means of triangles of different size. Smaller triangles are located in high curvature regions and bigger triangles in low curvature regions (e.g., planar regions). The final triangular mesh contains 6,787 triangles while the original 3D range image contains 26,028 triangles. CPU times have been measured on a SGI Indigo II with a 175 MHz R10000 processor. The CPU time to compute the curvature space mapping was 0.95 sec. The 3D Delaunay triangulation took 31.56 sec. and the loading of the hash table 3.28 sec. Finally, the 31 digging iterations took 23.07 sec. The maximum approximation error ξ for this example correspond to a 0.041% of the maximum z value of the given 3D range image.

Fig. 3 shows different triangular approximations from another real range image. The original triangular mesh contains 27,798 triangles. The CPU time to compute the curvature space mapping was 1.29 sec. The 3D Delaunay triangulation was computed in 44.01 sec. and the loading of the hash table took 4.44 sec. Fig. 3 (*top*) shows the final triangular mesh obtained after 44 digging iterations. It contains 2,628 triangles. Its maximum approximation error corresponds to a 0.13% of the maximum z value of the given 3D range image. The CPU time to compute these iterations was 5.17 sec. Fig. 3 (*middle*) shows another approximation of the same range image. It has a maximum approximation error of 0.039% of the maximum z value of the given 3D range image, and contains 5,630 triangles. This final representation was obtained after 52 digging iterations and the CPU time was 11.79 sec. Fig. 3 (*bottom*) shows an approximation with a maximum error of 0.013%. This representation was obtained after 58 digging iterations. The final triangular mesh contains 9,545 triangles and was obtained in 24.01 sec. More experimental results can be found in [8].

Finally, the 3D range image corresponding to the example of Fig. 3 has been utilized to compare the proposed technique with two public iterative decimation algorithms: *JADE* [5] and *Simplification Envelopes* [7]. In both algorithms, the approximation error was set to the same value utilized for the proposed technique. *JADE* generates a triangular mesh that approximates the 3D

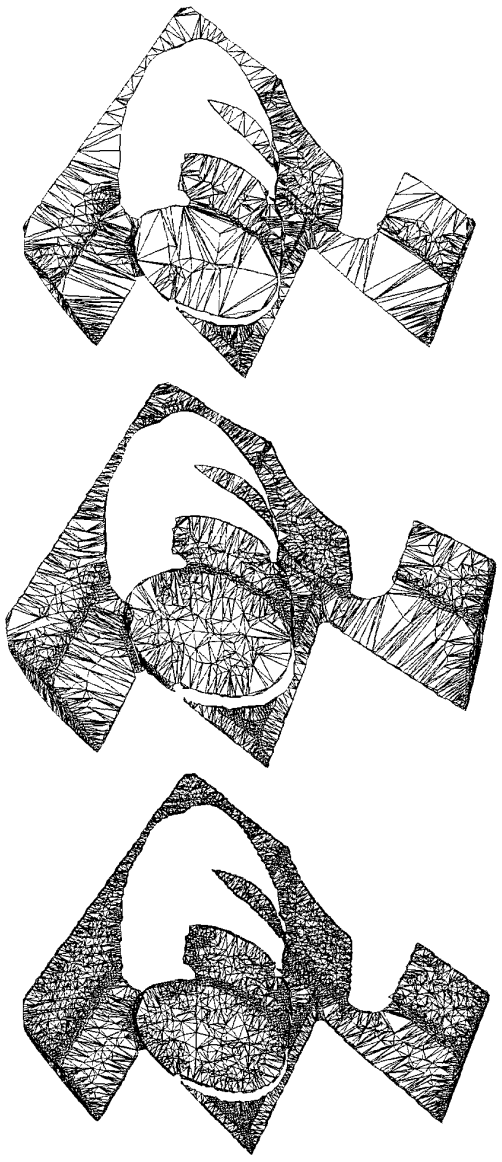

Figure 3. Three different approximations of a real range image. The 3D range image contains 27,798 triangles. (*top*) Final exterior mesh containing 2,628 triangles, approximation error 0.13%. (*middle*) Final exterior mesh containing 5,630 triangles, approximation error 0.039%. (*bottom*) Final exterior mesh containing 9,545 triangles, approximation error of 0.013%.

range image with 5,205 triangles. The result is comparable to the one obtained with the proposed technique but it takes 142.7 sec., more than twice the time spent by the proposed technique (58.86 sec.). *Simplification Envelopes* generates a comparable approximation with 5,495 triangles in 1,315 sec.

4. Conclusions and further improvements

A new approach for generating bounded error triangular meshes from range images with no optimization has been presented. This technique maps the pixels of the given range image to a 3D curvature space. After triangulating those points, a digging process starts eroding the external surface of the resulting tetrahedronization until a mesh fulfilling the desired tolerance is obtained. The approximation error of that external surface is verified in the 3D image space.

The proposed technique does not apply any optimization step and is more advantageous than other approaches, such as [7], whenever a coarse resolution mesh is sought, since the iterative process starts with a coarse resolution triangular mesh, the convex hull, and progressively refines it until the desired mesh is obtained.

We are currently studying the application of discontinuity-preserving filtering techniques [9] in order to reduce the oversampling of planar regions due to noise. Comparisons of the proposed technique with hierarchical triangulation approaches (e.g., [10]) will also be studied.

5. References

[1] M. A. García, A. Sappa and L. Basañez, Efficient approximation of range images through data dependent adaptive triangulations, *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Puerto Rico, 628-633, June 1997.

[2] M. A. García and L. Basañez, Fast extraction of surface primitives from range images, *13th IAPR Int. Conf. on Pattern Recognition, Vol. III: Applications and Robotic Systems*, Vienna, Austria, August 1996, 568-572.

[3] M. A. García, S. Velázquez and A. Sappa, A two-stage algorithm for planning the next view from range images, *British Machine Vision Conf.*, Southampton, UK, 720-729, 1998.

[4] M. Soucy and D. Laurendeau, Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 1-14, January 1996.

[5] A. Ciampalini and others, Multiresolution decimation based on global error. *The Visual Computer, Springer-Verlag*, 13(5), June 1997.

[6] L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications*, pp. 67-78, March 1989.

[7] J. Cohen and others, Simplification envelopes. *SIGGRAPH'96*, 119-128.

[8] A. Sappa, *Automatic generation of 3D geometric models from range images*, Ph.D thesis, Polytechnic University of Catalonia, 1999.

[9] S. Li, Close-Form Solution and parameter selection for convex minimization-based edge-preserving smoothing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 916-932, September 1998.

[10] L. De Floriani and E. Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, vol. 14, no. 4, pp. 363-411, October 1995.