Efficient Approximation of Range Images Through Data-Dependent Adaptive Triangulations

Miguel Angel García
Department of Software (SIG-LSI)
Polytechnic University of Catalonia
Diagonal 647, planta 8. 08028 Barcelona, Spain
garcia@lsi.upc.es

Angel Domingo Sappa Luis Basañez
Institute of Cybernetics
Polytechnic University of Catalonia
Diagonal 647, planta 2. 08028 Barcelona, Spain
{sappa, basanez}@ic.upc.es

Abstract

This paper presents an efficient algorithm for generating adaptive triangular meshes from dense range images. The proposed technique consists of two stages. First, a quadrilateral mesh is generated from the given range image. The points of this mesh adapt to the surface shapes represented in the range image by grouping in areas of high curvature and dispersing in low-variation regions. The second stage splits each quadrilateral cell obtained before into two triangles. Between the two possible flips, it is chosen the one whose diagonal's direction is closest to the orientation of the discontinuities present in that cell. Both stages avoid costly iterative optimization techniques. Results with real range images are presented. They show low CPU times and accurate triangular approximations of the given images.

1. Introduction

Range images are gaining popularity in computer vision since allow the efficient acquisition of 3D information. A range image is a two-dimensional array of pixels. Each pixel represents the distance from a point that lies on the surface of a 3D object to a virtual plane referred to the range sensor utilized to acquire the image.

The processing associated with range images can be significantly reduced by working with data representations able to keep the same shapes defined by the range images but with fewer data points. Triangular meshes are such a data representation since can efficiently adapt to intricate shapes. Then, further processing algorithms (e.g., segmentation, integration, recognition) can directly work on triangular meshes obtained from range images performing more efficiently. For instance, [1] presents a fast technique for segmenting range images approximated by triangular meshes.

Besides the speed-up of further processing algorithms, triangular meshes are a convenient representation in order to integrate the surfaces described by the range image in a world model [2] or to include them in CAD packages.

Different techniques have been proposed for the approximation of dense range images with triangular meshes (e.g., [5][6]). However they are based on costly iterative optimization algorithms that take into account all pixels of the original range image. This may make their application to systems with real-time constraints difficult.

Trying to overcome that efficiency problem, [3] presents a fast algorithm for generating triangular meshes from range images avoiding optimization techniques. However, the randomized nature of the previous technique produces two effects that may be inconvenient for some applications. First, the number of selected points and triangles cannot be specified a priori. Second, the obtained meshes tend to be quite irregular and it is also frequent to find degenerated triangles.

This paper presents a non-randomized technique to obtain triangular meshes from range images with no optimization. Since a deterministic algorithm is applied, the number of points and triangles can be specified a priori. Moreover, triangle shapes tend to vary more gently by construction. Obviously, these new requirements lead to a certain time penalty with respect to the simpler randomized technique. However, the proposed method is still very efficient since avoids iterative optimization and it is inherently parallel.

This paper is organized as follows. The proposed technique is described in section 2. Section 3 presents experimental results with real range images. Finally, conclusions and further improvements are given in section 4.

This work has been partially supported by the Government of Spain under the CICYT project TAP96-0868. The second author has been supported by the Spanish Agency for International Cooperation and the National University of La Pampa (Argentina).

1063-6919/97 $10.00 © 1997 IEEE

2. Non-randomized adaptive triangulation of range images

Adaptive triangular meshes are generated from range images in two stages. In the first stage, an adaptive quadrilateral mesh approximating the original range image is obtained with no optimization.

The second stage divides each previously obtained quadrilateral cell into two triangles by applying a new technique for choosing the diagonal (flip) that best agrees with the discontinuities present in that cell. Both stages are described below.

2.1. Generation of adaptive quadrilateral meshes from range images

This section describes a technique for obtaining quadrilateral meshes that adapt to the shape of the surfaces contained in a given range image. A complete mathematical description can be found in [4].

Let $\mathbf{R}(r, c)$ be a range image with R rows and C columns, $r \in [0, R)$, $c \in [0, C)$, where each valid pair (r, c) denotes a pixel located at row r and column c. Pixels belonging to the background of the image are given a constant value β.

The generation of an adaptive quadrilateral mesh consists of three steps. First, an estimation of curvature is computed for every pixel of the given range image. Second, the range image is tessellated into a user-defined number of tiles that overlap along one line of pixels. Finally, an adaptive quadrilateral mesh is separately computed for each tile based on the range image and its curvature estimation. These steps are described next.

2.1.1. Curvature estimation

Let $\mathbf{R}(r, c)$ be the input range image. The objective of this step is the generation of a *curvature image* $\mathbf{K}(r, c)$ that gives, for every pixel, an estimation of its curvature. First horizontal $\mathbf{K}^R_{rc}$ and vertical $\mathbf{K}^C_{rc}$ curvature estimations are obtained as:

$$\mathbf{K}^C_{rc} = |\mathbf{R}(r, c-1) - 2\mathbf{R}(r, c) + \mathbf{R}(r, c+1)|$$

$$\mathbf{K}^R_{rc} = |\mathbf{R}(r-1, c) - 2\mathbf{R}(r, c) + \mathbf{R}(r, c+1)|$$

The sought curvature is finally obtained as the logical addition of both estimations: $\mathbf{K}(r, c) = \mathbf{K}^R_{rc} \vee \mathbf{K}^C_{rc}$.

Background pixels are given a constant curvature value (e.g., 15) that avoids unnecessary concentration of points along the boundary between the background and valid regions of the image.

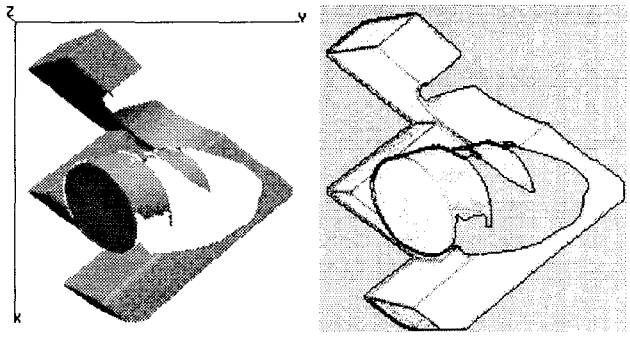

Figure 1. (*left*) Original range image (rendered with perspective). (*right*) Curvature image of the given range image.

Fig. 1 shows an example of a real range image and the curvature image obtained by applying this technique. Dark regions represent areas of high curvature.

2.1.2. Range image tessellation

In order to obtain a quadrilateral mesh adapted to the different regions of the range image taking into account their particular characteristics, and also in order to favor parallelism, both the given range image and its associated curvature image are partitioned into a user-defined number of rectangular tiles. Each tile is considered to be a small range image upon which further processing is applied.

In particular, each range image is divided into H horizontal and V vertical stripes giving rise to HxV different tiles. This partition is not disjoint. A tile shares one row or column of pixels with each of its adjacent tiles.

Hereafter, $\mathbf{R}_{vh}(r, c)$ will represent one tile identified by the number of both the vertical and horizontal stripe it belongs to, $v \in [0, V)$, $h \in [0, H)$. The row and column coordinates (r, c) are local to the tile: $r \in [0, R/V]$, $c \in [0, C/H]$. $\mathbf{K}_{vh}(r, c)$ represents the curvature image corresponding to $\mathbf{R}_{vh}(r, c)$.

2.1.3. Generation of adaptive quadrilateral meshes from range image tiles

Given a range image tile $\mathbf{R}_{vh}(r, c)$ and its corresponding curvature image $\mathbf{K}_{vh}(r, c)$, the objective of this step is to sample the given tile at $\mathcal{R}$ x ζ positions covering the whole tile and adapting to the shape of the surfaces comprised in it with the condition that these positions form a quadrilateral although not necessarily uniform grid. The outcome of this step will be an array of $\mathcal{R}$ x ζ points, each point consisting of the row and column coordinates of one pixel contained in the tile.

In order to generate this array, each row of the given tile is adaptively sampled at ζ different positions so that points tend to concentrate in high-curvature areas and to disperse in low-variation regions.

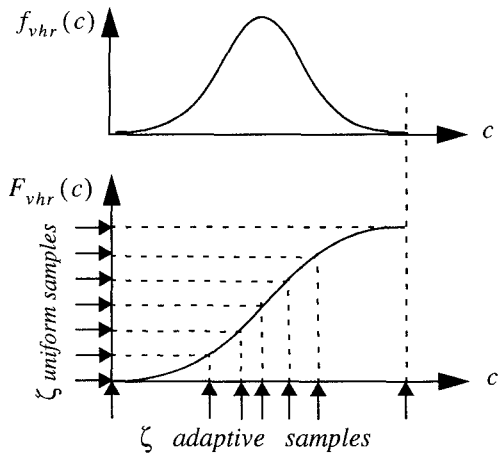

Figure 2. (*top*) Unnormalized probability density function $f_{vhr}(c)$ representing curvature associated with every pixel c. (*bottom*) Uniform sampling of the image space of the corresponding unnormalized probability distribution function $F_{vhr}(c)$ gives a set of points whose density varies according to $f_{vhr}(c)$.

This is done as follows. Let $\mathbf{R}_{vhr}(c)$ be a row of pixels and $\mathbf{K}_{vhr}(c)$ its corresponding curvature profile. This profile is converted to an unnormalized probability density function f_{vhr} that denotes the probability of selecting a given pixel based on its curvature.

Then, an unnormalized probability distribution function $F_{vhr}(c)$ is obtained by "integrating" $f_{vhr}(c)$:

$$F_{vhr}(c) = \sum_{i=0}^{c} f_{vhr}(i) - f_{vhr}(0) \qquad (1)$$

$F_{vhr}(c)$ is unnormalized since it does not range between zero and one but between zero and a maximum value $\mathcal{M}$.

If the image space of $F_{vhr}(c)$ is sampled at ζ uniformly distributed points, the application of the inverse distribution function $F^{-1}{}_{vhr}(y)$ to those points leads to a set of ζ points adaptively distributed according to $f_{vhr}(c)$. This principle is illustrated in Fig. 2. In our case, since the probability distribution function corresponds to the curvature estimation, the density of points will depend on that curvature and, hence, on shape variations.

When this procedure is applied to all rows of the given tile and, for each row, the ζ selected points are displayed in their original positions in the tile, a set of "vertical" curves is obtained, Fig. 3(*left*). These curves determine the columns of the final mesh. Note that they tend to approach in areas of high curvature and to disperse in low variation regions. Moreover, in those areas without curvature information, such as the background, curves are uniformly distributed like in uniform sampling.

Owing to the overlap between adjacent tiles, the same distribution of selected points is obtained at the common boundary. Hence, curves belonging to adjacent patches join smoothly. The final result is a complete range image sampled in vertical curves that adapt to the shape of the individual tiles that make up the whole range image.

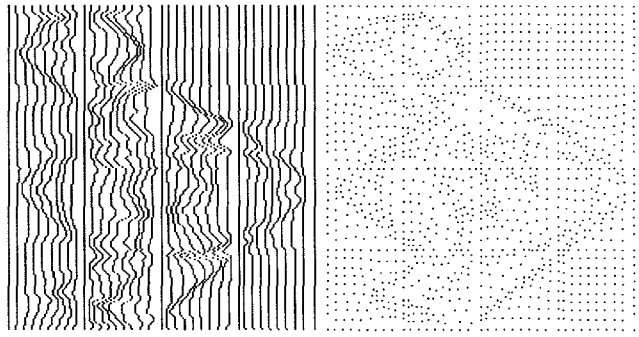

Figure 3. (*left*) Vertical curves after adaptive horizontal sampling, with $H = V = 4$ and $\zeta = 10$. (*right*) Horizontal and vertical sampled points with $\mathcal{R} = \zeta = 10$.

The next step samples each of those curves at $\mathcal{R}$ different positions for every tile. The principle is similar to the one applied before, but now the curvature profiles utilized to obtain the unnormalized probability density function f are not obtained from rows of pixels but from the pixels that make up one of the curves. Each vertical curve inside a tile is traversed and the curvature values associated with its pixels are stored in a vector utilized as a new curvature profile.

Given this curvature profile, an unnormalized probability density function and its corresponding unnormalized probability distribution function are computed similarly to (1) but now considering that the input parameter is the row number instead of the column number used before.

In the end, each vertical curve of each tile is adaptively sampled at $\mathcal{R}$ positions that adapt to the shapes contained in the tile with independence from the other tiles. Again, the overlap between adjacent tiles guarantees the continuity of rows of pixels between horizontal adjacent tiles.

Fig. 3(*right*) shows the set of points obtained in this way. Notice how points tend to concentrate in areas of high curvature and to disperse in low variation regions. Uniform sampling is obtained in no variation regions such as the background. The final adaptive quadrilateral mesh is trivially computed by joining all those points horizontally and vertically.

Fig. 4 shows the uniform (*top*) and adaptive (*bottom*) quadrilateral meshes that approximate the given range image with 37x37 points each. Notice that the uniform mesh misses details that are captured by the adaptive mesh—take a look at the jump and crease edges marked on the figure. In order to display these images, each quadrilateral cell has been split up into two triangles by applying the technique described below.

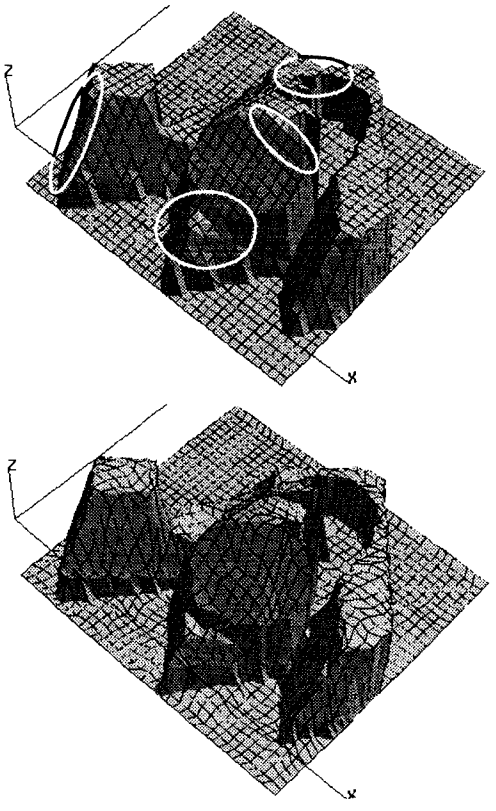

Figure 4. (*top*) Uniform quadrilateral mesh with 37x37 nodes. (*bottom*) Adaptive quadrilateral mesh with 37x37 nodes.

2.2. Data-dependent triangulation of quadrilateral cells

The outcome of the previous stage is a quadrilateral mesh with $(\mathcal{R}-1)V+1$ rows and $(\zeta-1)H+1$ columns, H and V being the number of horizontal and vertical partitions of the range image into tiles and $\mathcal{R}$ and ζ the number of rows and columns in which every tile is sampled. Each node of the quadrilateral mesh contains the row and column of a pixel of the range image and its corresponding value (depth measure). The objective now is to divide each of those cells into two triangles in order to obtain a triangular approximation of the given range image.

Several approaches have been proposed in the literature to perform the only apparently trivial operation of triangulating a quadrilateral cell. The problem consists of choosing the right diagonal used to split that cell—sometimes referred to as choosing the right *flip*.

2.2.4. Previous approaches

A popular method consists of choosing the diagonal that maximizes the minimum of all angles of the resulting triangles. This heuristic is known as the *Lawson's max-min optimality criterion* and is used to incrementally generate *Delaunay triangulations* (see [7] for a recent survey on triangulation algorithms). However the problem of this technique is that the original surface (range image in our case) is not considered for the diagonal choice. That may lead to incorrect behaviors when either surface or orientation discontinuities are present in the cell, since edges can be cut off by wrongly chosen diagonals.

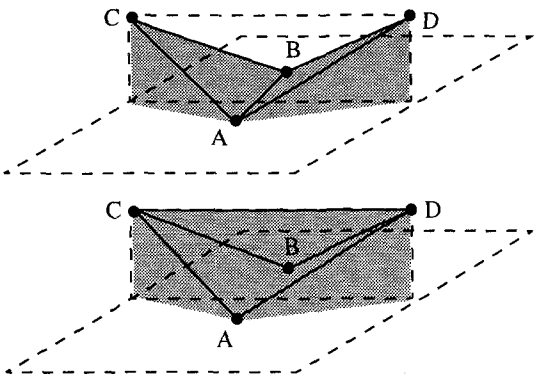

Figure 5. Possible triangulation of four points and their respective approximation error volumes.

The solution to this problem involves the application of a data-dependent triangulation [8] that takes into account the underlying shape of the surfaces contained in the cell. Two different criteria are proposed in [8]. The first one takes into account the normals of the triangles and chooses that combination of triangles with a minimum angle between their normals. This heuristic will not preserve discontinuities though and, therefore, it is not useful in our context, since edges will be cut off.

The second approach leads to a method that computes the difference (vertical distance) between each of the two possible triangulations and the pixels *covered* by those triangles. The objective then is to choose the triangulation that produces the lowest difference and, hence, error.

However, this technique does not always work properly. Suppose we want to approximate a wall perpendicular to a planar surface. Only four points are considered: two points on top of the wall (C,D) and two more points on the planar surface (A,B) as shown in Fig. 5. Those points admit two possible triangulations. Following the aforementioned technique, we would chose the flip that produces the lowest error volume. Error volumes are shown as dark regions in the figure. Therefore, the flip on top would be chosen as best although it does not preserve the surface discontinuity. This means that in order to keep discontinuities and thus the shape of the underlying surfaces, the error criterium is not always reliable.

Experimental results show that this error criterium tends to produce artifacts along edges, leading to flips that

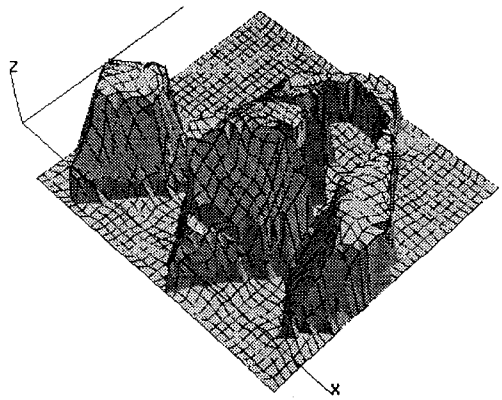

Figure 6. Triangulation of an adaptive quadrilateral mesh through an error-based diagonal selection algorithm (quadrilateral cells are shown in wireframe and triangles are shaded).

do not always agree with the orientation of discontinuities (see Fig. 6 and compare it with the result obtained with the proposed technique shown in Fig. 4[*bottom*]).

A similar problem worsened by a downfall in performance is obtained by applying techniques based on energy functions [9]. A different approach to the diagonal selection problem proposed in [10] and based on choosing the diagonal whose 3D length is smallest will also choose the top flip given the four points we have used for this example.

In this paper a different technique is proposed to triangulate a quadrilateral cell taking into account the discontinuities contained in it. This technique is general in the sense that can be applied to all kinds of quadrilateral meshes: adaptive and uniform.

The proposed technique is composed of three steps. First, the 2D gradient vector associated with each pixel of the given range image is estimated. Second, for each quadrilateral cell, an estimation of the overall gradient associated with the pixels traversed by each of its two possible diagonals is computed. Finally, the diagonal whose direction is closest to the perpendicular to that gradient vector is chosen as the one which best keeps the underlying discontinuities in the cell. These steps are described next.

2.2.5. 2D gradient estimation

Let $\mathbf{R}(r, c)$ be the given range image. The objective now is the generation of a *gradient image* $\mathbf{G}(r, c)$ that keeps, for every pixel of the range image, a 2D vector that represents the projection onto the range image reference plane of the normal to the surface corresponding to that pixel. The norm of this vector is proportional to the magnitude of the derivative of the underlying surface at that pixel. The gradient image is obtained as follows:

$$\begin{aligned} \mathbf{G}(r,c) &= \{G_h(r,c), G_v(r,c)\} \\ G_h(r,c) &= \mathbf{R}(r,c-1) - \mathbf{R}(r,c+1) \\ G_v(r,c) &= \mathbf{R}(r-1,c) - \mathbf{R}(r+1,c) \end{aligned}$$

The gradient image is filtered to reduce fast changes of orientation of the gradient vectors associated with neighboring pixels. Specifically, the 2D gradient vector at each pixel is substituted for the *composition* of the gradient vectors associated with that pixel and its eight neighbors. The composition of two 2D vectors, $v = (v_1, v_2)$, $w = (w_1, w_2)$, is done as follows:

$$v \oplus w = \begin{cases} (v_1 + w_1, v_2 + w_2) & v \cdot w > v \cdot (-w) \\ (v_1 - w_1, v_2 - w_2) & \text{otherwise} \end{cases} \quad (2)$$

where $v \cdot w$ is the dot product between both vectors.

2.2.6. Resultant gradients associated with the diagonals of a quadrilateral cell

Let us consider each quadrilateral cell of the adaptive quadrilateral mesh obtained above. Each cell is composed of four points: $\{\mathbf{C}_{i,j}, \mathbf{C}_{i,j+1}, \mathbf{C}_{i+1,j}, \mathbf{C}_{i+1,j+1}\}$, where $\mathbf{C}_{i,j}$ is the top-left cell's node and $\mathbf{C}_{i+1,j+1}$ the bottom-right cell's node. The components of $\mathbf{C}_{i,j}$ are the row and column of a pixel and its corresponding value: $\mathbf{C}_{i,j} = \{r_{i,j}, c_{i,j}, \mathbf{R}(r_{i,j}, c_{i,j})\}$.

The goal of this stage is to obtain, for each of the two possible diagonals that can split up that cell, a resultant gradient vector obtained by composing (2) the gradients associated with the pixels traversed by that diagonal. That resultant gradient will describe the orientation of the discontinuities crossed by the given diagonal.

Two diagonals are considered for each cell: $\mathbf{D}^+_{i,j} = \{\mathbf{C}_{i,j}, \mathbf{C}_{i+1,j+1}\}$ and $\mathbf{D}^-_{i,j} = \{\mathbf{C}_{i,j+1}, \mathbf{C}_{i+1,j}\}$. In both cases, the set of pixels traversed by a diagonal is determined by applying Bresenham's algorithm [11] to "draw" a straight line between the pixels corresponding to the initial and final nodes of the cell. In order to take into account aliasing effects due to the discretization of that straight line, the latter is thickened by adding, for each pixel, its neighbors above and below.

In summary, given a certain diagonal, a set of pixels is selected. Then, the gradient vector associated with that diagonal is obtained by composing, using (2), the 2D gradient vectors $\mathbf{G}(r, c)$ corresponding to each selected pixel from the set. Finally two resultant gradients are obtained, $\mathbf{G}^+_{i,j}$ and $\mathbf{G}^-_{i,j}$, one for each diagonal.

2.2.7. Diagonal selection

From the resultant gradients, the one with largest norm is chosen as the final representative of the orientation of the discontinuities that are potentially involved in the diagonal selection process. Let $\mathbf{G}_{i,j} = \{G_{[i,j]h}, G_{[i,j]v}\}$ be that resultant gradient

associated with cell (i, j). Next a tangent vector is computed as: $\mathbf{T}_{i,j} = \{T_{[i,j]h}, T_{[i,j]v}\} = \{G_{[i,j]v}, -G_{[i,j]h}\}$.

Since this tangent is orthogonal to the resultant gradient vector, it agrees with the orientation of the discontinuities present in the cell. The objective now is to choose as splitting diagonal the one whose direction is the most similar to the direction of $\mathbf{T}_{i,j}$.

Let $\beta_{i,j}$ be the orientation of $\mathbf{T}_{i,j}$ obtained as $\beta_{i,j} = \text{atan}(T_{[i,j]v}/T_{[i,j]h})$. The orientations of both diagonals are obtained as

$$\alpha^{+}_{i,j} = \text{atan}\left(\frac{r_{i+1,j+1} - r_{i,j}}{c_{i+1,j+1} - c_{i,j}}\right) \quad \alpha^{-}_{i,j} = -\text{atan}\left(\frac{r_{i+1,j} - r_{i,j+1}}{c_{i+1,j} - c_{i,j+1}}\right)$$

Let us consider $\delta^{+} = \left|\alpha^{+}_{i,j} - \beta_{i,j}\right|$. The minimum angle between $\mathbf{T}_{i,j}$ and diagonal $\mathbf{D}^{+}_{i,j}$ is defined as

$$\mu^{+} = \begin{cases} \delta^{+} & |\delta^{+}| \le \pi/2 \\ \pi - \delta^{+} & \text{otherwise} \end{cases}$$

The minimum angle μ^{-} between $\mathbf{T}_{i,j}$ and $\mathbf{D}^{-}_{i,j}$ is defined analogously. Finally, diagonal $\mathbf{D}^{+}_{i,j}$ is selected to split up the given quadrilateral cell if $\mu^{+} \le \mu^{-}$. Otherwise, diagonal $\mathbf{D}^{-}_{i,j}$ will be selected.

3. Experimental results

The proposed technique has been applied to real range images containing both surface and orientation discontinuities (jump and crease edges).

Since this technique is devised to improve the quality of approximation of edges and they usually cover a small percentage of the overall range image, improvements are qualitative rather than quantitative and, hence, they are better appreciated through images of the final result.

The example used so far corresponds to a 197x187 (rows x columns) range image. This image has been split up into 4 by 4 tiles ($H = V = 4$) and 10 by 10 points have been adaptively sampled at each tile. After a 37x37 adaptive mesh is generated, its quadrilateral cells are split up using the technique proposed above. Fig. 4(*bottom*) shows the final result of applying the proposed technique. Quadrilateral cells are shown in wireframe while triangles are shaded. In contrast, Fig. 6 shows the same quadrilateral mesh triangulated using the error-based diagonal selection method described in section 2.2. Notice the artifacts along edges in the latter. Finally, Fig. 4(*top*) shows the approximation of the original range image from a uniform quadrilateral mesh with the same number of points and applying our diagonal selection algorithm. The CPU time to compute the final triangulation was 0.60 sec. on a 200 MHz R4400 SGI Indigo II. From this time, 0.25 sec. were used by the diagonal selection algorithm.

4. Conclusions and further improvements

This paper presents an efficient technique for generating adaptive triangular meshes from range images. Initially, the algorithm generates an adaptive quadrilateral mesh from the range image. This mesh is more accurate than a uniformly sampled one since it distributes the same number of points considering the curvature of the surfaces contained in the range image. Then, each quadrilateral cell is divided into two triangles by choosing one of the two possible diagonals. This choice is based on a selection algorithm that analyzes the orientation of the gradient at the pixels traversed by those diagonals. The proposed diagonal selection technique is general in the sense that can be applied to any kind of quadrilateral meshes (adaptive or uniform).

The current technique only tries diagonal flips inside quadrilateral cells. An immediate improvement consists of trying to flip edges that originally define the quadrilateral cells. A further line of research will consist of automatically determining the number of rows and columns of the quadrilateral mesh and thus the number of points and triangles of the final triangular mesh.

An implementation of this technique in C is available by contacting the authors.

5. References

[1] 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.

[2] M. A. García and L. Basañez, Efficient free-form surface modeling with uncertainty. *IEEE Int. Conf. on Robotics and Automation*, Minneapolis, USA, April 1996, 1825-1830.

[3] M. A. García, Fast approximation of range images by triangular meshes generated through adaptive randomized sampling. *IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, May 1995, 2043-2048.

[4] M. A. García, A. D. Sappa and L. Basañez, Fast Generation of Adaptive Quadrilateral Meshes from Range Images. *Accepted for publication in IEEE Int. Conf. on Robotics and Automation*, Albuquerque, USA, April 1997, 6 pages.

[5] M. Soucy, A. Croteau and D. Laurendeau, A multiresolution surface model for compact representation of range images, *IEEE Int. Conf. on Robotics and Automation*, 1992, 1701-1706.

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

[7] S. Kumar, Surface Triangulation: A Survey. *Technical Report, Department of Computer Science, University of North Carolina*, January 1996.

[8] L.L Schumaker, Triangulations in CAGD. *IEEE Computer Graphics and Applications*, pp. 47-52, January 1993.

[9] Hoppe et al., Mesh Optimization. *SIGGRAPH '93*, 19-26.

[10] M. Rutishauser, M. Stricker and M. Trobina, Merging Range Images of Arbitrarily Shaped Objects. *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 1994, 573-580.

[11] T. Pavlidis, *Algorithms for Graphics and Image Processing*. MD Computer Science Press, 1982.