3.3.2 Approximation of Digital Images with Discontinuity-Preserving Adaptive Triangular Meshes

In the previous technique, image contours (edges) induce an oversampling of the original image. To avoid this drawback, this section proposes a preprocessing stage to detect and separately approximate the edges present in the given image. Then, the adaptive sampling technique described in Section 3.3.1.1 is applied to the regions comprised among the detected edges. The proposed technique differs from previous techniques in which both image discontinuities are explicitly modeled and iterative optimization is avoided. The algorithm consists of two main stages.

In the first stage, the original image is adaptively sampled at a set of pixels, taking into account both image discontinuities and curvatures. In the second stage, the sampled pixels are triangulated by applying a constrained 2D Delaunay algorithm (Section 2.3.1.2). The obtained triangular meshes are compact representations that model the regions and discontinuities present in the original image with many fewer points. Thus, further image processing operations applied upon those meshes can perform faster than upon the original images.

3.3.2.1 Image Adaptive Sampling

The aim of this stage is to sample a given image, obtaining a set of pixels that are distributed according to the shades and discontinuities present in it. Those pixels are obtained by applying two different adaptive sampling processes. In the first process, the image contours (edges) are detected, adaptively sampled and approximated by polylines. In the second process, the internal regions comprised between image edges are adaptively sampled.

Edge Adaptive Sampling

Gray level images usually contain sudden changes in gray level due to region boundaries. The edge sampling stage approximates those boundaries with adaptive polylines. First, the edges present in the original image are found by applying Canny's edge detector (Canny, 1986), and then by thresholding the result such that all the pixels whose value is above zero

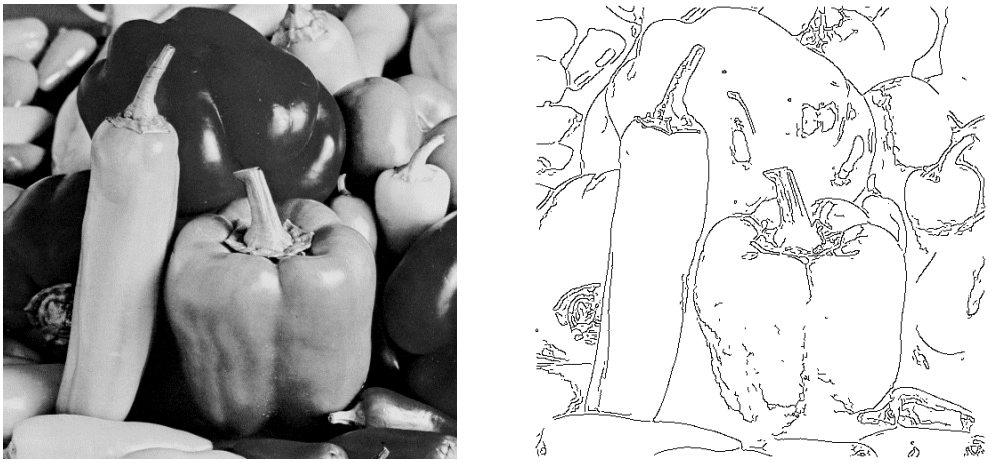

Figure 3.15. (*left*) Original gray level image with 262,144 pixels (512x512). (*right*) Edge image generated from the previous image.

are set to gray level 0 (black) while the other pixels are set to gray level 255 (white). Thus, an *edge image* is generated, such as it is shown in Figure 3.15(*right*).

Afterwards, every edge in the edge image is adaptively approximated by a collection of segments that constitute a *polyline*. The points that define the segments of a polyline are obtained through the following iterative procedure.

First, the edge image is scanned from left to right and from top to bottom until a pixel belonging to an edge is found. This pixel is chosen as the *starting point, A*. The chosen edge is traversed from the starting point, and a second pixel contained in the same edge and placed at a user defined number of pixels away from the starting point is selected. This second pixel is the *reference point, C*. Both, the starting point and the reference point generate an *approximating segment,* $\overline{AC}$. The pixels belonging to the chosen edge, which are comprised between the starting point and the reference point constitute the *approximated points*. The distance in image coordinates between each approximated point and the approximating segment is the approximation error, ξ. If all the current approximation errors are below a given threshold d, a new reference point is selected by advancing the previous reference

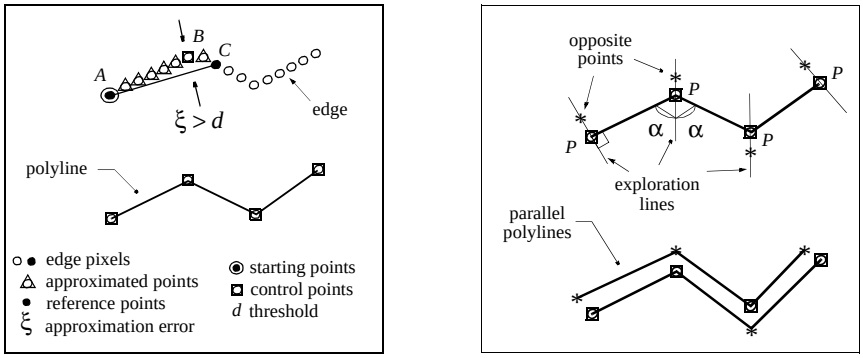

Figure 3.16. Edge sampling process.(*left*) Edge extraction. (*right*) Edge unfolding.

point *C* a fixed number of pixels along the chosen edge. Then, a new segment, joining the starting point and this new reference point is defined and the previous procedure is iterated.

When an approximated point is found to have an error above d, that point is chosen as the new starting point, *B*. The edge is traversed in this way until either one of its extremes is reached or a bifurcation is found. The polyline that approximates the previous edge with an error bounded by d is the set of segments that join all the starting points found during the exploration of the edge, plus the final point in the edge (extreme or bifurcation). The points that define the polyline constitute the *control points*. Figure 3.16(*left*) shows an example of the previous procedure.

When an edge has been successfully approximated by a polyline, all the points traversed during the process are removed from the edge image so that they do not intervene in further edge approximations. This polyline extraction procedure is applied until all edges have been approximated and, therefore, the edge image is white. Since the starting points of the polylines are found by applying the aforementioned scan-line algorithm, different executions of the edge sampling process upon a same image will produce the same polylines.

Each polyline obtained above delimits a boundary between two neighboring regions, indicating a discontinuity in the gray level values. The points that form the polyline (control points) correspond to pixels that can be located at both sides of the discontinuity. However,

the final $2^1/_2$D triangular mesh requires that these discontinuities be modeled as vertical "walls". These walls can only be produced by unfolding each obtained polyline into two parallel polylines, each at a different gray-level (height). This process consists of computing an *opposite point* for each polyline's control point. Each opposite point will be located at the other side of the discontinuity in which its corresponding control point lies.

Given a control point *P*, its corresponding opposite point is obtained as follows. First, an *exploration line* that passes through *P* and bisects the polyline's segments that meet at *P* is computed, Figure 3.16(*right*). The pixels traversed by this line are explored in both directions starting with *P*. The first pixel along that line where a significative change of gray level occurs is chosen as *P*'s opposite point. The opposite points corresponding to the extremes of the polyline are determined by considering as exploration lines the lines perpendicular to the segments that abut at those extremes, and then by applying the previous criterion. Figure 3.16(*right*) shows an example of this procedure.

A new polyline is obtained for each original polyline by linking its corresponding opposite points. Since the control points that define the original polyline may not be located at the same side of the discontinuity, the new polyline and the original one may not be parallel and, therefore, may have some segments that self-intersect. To avoid this problem, the two polylines are traversed exchanging corresponding pairs of control and opposite points, such that each polyline only contains the points that are located at the same side of the discontinuity (all the points of a polyline must have a similar gray level). Thus, two parallel polylines are finally generated from each original polyline: one completely lying at one side of the discontinuity (at the region with the highest gray level) and the other completely lying at the other side (at the region with the lowest gray level), Figure 3.16(*right*). Figure 3.17 shows the set of control points and opposite points corresponding to the current example. Figure 3.17(*middle*) illustrates the edges approximated with adaptive parallel polylines.

Region Adaptive Sampling

This stage aims at choosing a set of pixels adaptively distributed over the image, such that they concentrate in high-curvature regions and scatter over low variation regions. The sam-

Figure 3.17. (*left*) Set of both control and opposite points obtained by the edge sampling process (4,900 points). (*middle*) Parallel polylines obtained from the previous points. (*right*) Detail of parallel polylines.

pling process must be applied by taking into account that all the edges in the image have already been considered by the previous step and do not have to be resampled.

This process consists of applying the non-optimization adaptive sampling technique previously described in Section 3.3.1.1, excluding from the image those pixels that belong to contours already detected in the edge sampling stage (the ones that belong to the edges detected by applying Canny's edge detector and its neighbors). Figure 3.18(*right*) illustrates the pixels obtained after applying the region adaptive sampling process.

The pixels corresponding to the points that have been selected after the edge sampling process, as well as the pixels obtained after the region sampling process, are merged, giving rise to the final result of the adaptive sampling stage. Figure 3.19(*left*) shows the set of sampled pixels that approximate the given original image.

The proposed image approximation algorithm differs from previous approaches (Altunbasak, 1997; Wolberg, 1997; Gevers, Smeulders, 1997; Davoine, Svensson, Chassery, 1995; Terzopoulos, Vasilescu, 1991) in two main features. On the one hand, no iterative optimization is applied, since there is no attempt to maximize or minimize any measure. On the other hand, image discontinuities are modeled by means of parallel polylines that separately capture both the upper and lower edges of the discontinuity. Previ-

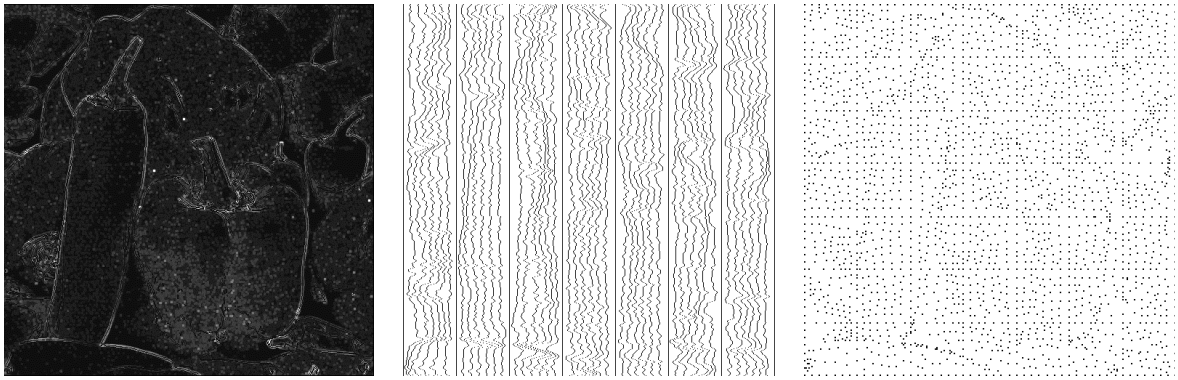

Figure 3.18. (*left*) Curvature image. (*middle*) Adaptively sampled vertical curves. (*right*) Set of pixels obtained by region adaptive sampling whit $H = V = 7$, and $\mathcal{R} = \zeta = 8$, (2,324 points).

ous approaches only find a single polyline per discontinuity. Hence, this polyline may traverse pixels belonging to both sides of the discontinuity, leading to incorrect slashes in the final model.

3.3.2.2 Triangular Mesh Generation

The final aim of this stage is the generation of an adaptive triangular mesh from the set of pixels obtained after the adaptive sampling stage. The generated triangular mesh preserves the shades and discontinuities present in the given image. That mesh is obtained by applying a constrained 2D Delaunay triangulation algorithm (Shewchuck, 1996a) to the row and column coordinates of the sampled pixels. The constraints of the triangulation are the parallel polylines that approximate the detected contours of the image. These polylines can be either open or closed. In this way, it is guaranteed that discontinuities of the gray level values are preserved as edges in the generated triangular mesh. Figure 3.19(*middle*) shows the final adaptive triangular mesh obtained for the current example. The *z* coordinates of the vertices of that mesh correspond to the gray levels associated with them in the original image. The approximating image obtained from the previous triangular mesh displayed in Figure 3.19(*middle*) is presented in Figure 3.19(*right*). It was obtained by applying the technique presented in Section 3.4.

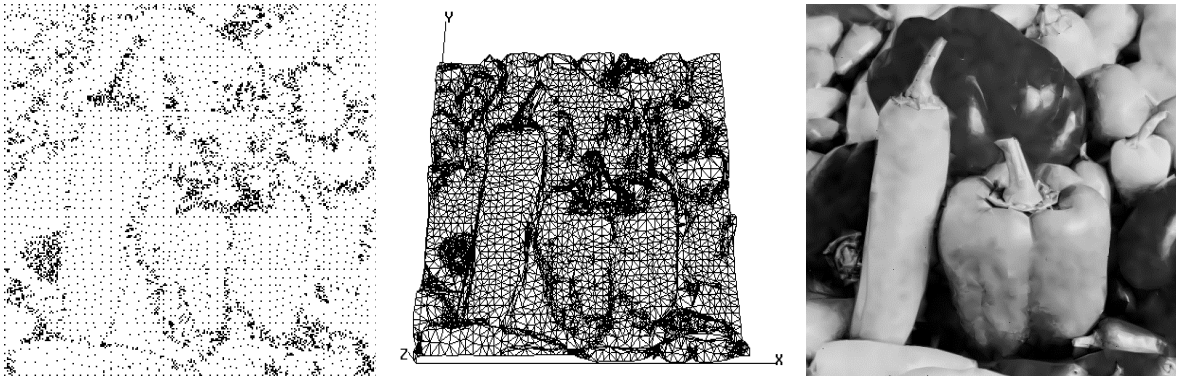

Figure 3.19. (*left*) Final set of pixels obtained after the adaptive sampling process (7,224 points). (*middle*) Adaptive triangular mesh generated from the previous sampled pixels. (*right*) Approximating image obtained from the previous triangular mesh through z-buffering in 0.31 sec.

The RMS error of the image shown in Figure 3.19(*right*) is 12.1. This RMS error can be bounded to a desired tolerance by applying the refinement steps of the previously developed algorithm (Section 3.3.1), which approximates a given image through bounded error triangular meshes.

3.3.2.3 Experimental Results

Two sets of real digital images have been used to test the algorithm described above. The first set includes range images and the second one gray level images. All the CPU times presented in this section were measured on a SGI Indigo II with a 175 MHz R10000 processor.

Figure 3.20(*top row*) shows the first test image corresponding to a range image with 28,182 pixels. The initial parameters specified by the user were experimentally set to the following values: for the edge adaptive sampling, the distance between the starting point and the reference point (approximating segment length) was set to 4. The distance between each approximated point and the approximating segment (approximating error), was set to 1; furthermore, for the region adaptive sampling, the original image was partitioned in 4(H) x 5(V) tiles, considering 4($\mathcal{R}$) x 4(ζ) pixels per tile. Figure 3.20(*middle row, left*) shows the final set of sampled pixels after applying the whole image adaptive sampling process. The

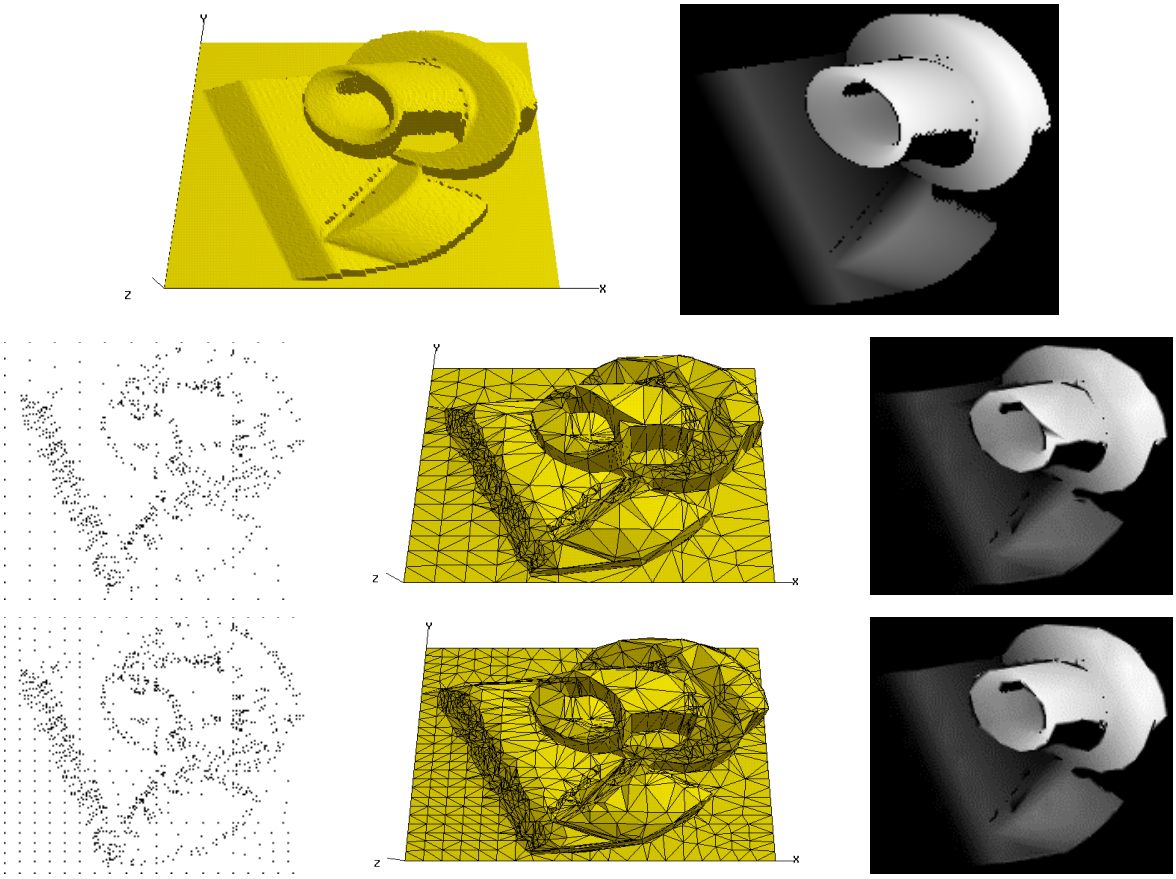

Figure 3.20. (*top row*) Original range image with 28,182 pixels (154x183) rendered in the 3D space (*left*) and in the image space (*right*). (*middle row*) Image approximation with 918 sampled pixels, obtained by applying the image adaptive sampling process, RMS = 18. (*bottom row*) Image approximation with 1,222 sampled pixels, obtained by applying the adaptive sampling process, RMS = 16.2.

result contains 918 points. The triangular mesh generated from these points is shown in Figure 3.20(*middle row, middle*). This mesh contains 1,780 triangles. The total CPU time to generate the previous mesh was 0.39 sec. Figure 3.20(*middle row, right*) shows the approximating image generated from the previous adaptive triangular mesh, obtained through z-buffering in 0.04 sec. The RMS error is 18.

Figure 3.20(*bottom row*) shows the results after applying the proposed approximation algorithm with different parameters. The image was divided into 4(H) x 5(V) tiles, considering 6($\mathcal{R}$) x 6(ζ) pixels per tile. The approximating segment length was set to 4 and the

approximating error to 1. According to those parameters, 1,222 points were obtained. The adaptive triangular mesh generated from the previous points contains 2,352 triangles. The total CPU time to generate those results was 0.42 sec. The approximating image was generated through z-buffering in 0.04 sec. The RMS error between the approximating image and the original image is 16.19.

The second test range image is shown in Figure 3.21(*top left*). This image contains 262,144 pixels. The image was divided into 6(H) x 6(V) tiles, considering 9($\mathcal{R}$) x 9(ζ) pixels per tile. The approximating segment length was set to 4 and the approximating error to 1. The image was sampled at 3,926 pixels, Figure 3.21(*top right*). The adaptive triangular mesh generated from previous set of pixels is shown in Figure 3.21(*bottom left*). This mesh contains 7,568 triangles. The total CPU time to obtain the adaptive mesh from the given image was 5.44 sec. The approximating image was obtained in 0.24 sec by applying z-buffering (Section 3.4.2). The obtained RMS error is 9.1.

Additionally, the proposed approximation algorithm has been tested with gray level images of different size and also compared to both a uniform (non-adaptive) sampling technique and a mesh decimation technique based on iterative optimization (JADE), described in (Ciampalini et at., 1997).

The uniform sampling technique consists of choosing one pixel out of a predefined number of pixels along the rows and columns of the image. On the other hand, the optimization-based technique (JADE) starts with a dense triangular mesh containing all the pixels from the image, and decimates it until either a certain number of points is obtained or the approximation error is above a threshold. In order to be able to compare these techniques with the proposed one, JADE and the uniform sampling process were run to produce triangular meshes with a similar number of pixels as the ones obtained with the proposed technique.

Figure 3.22 shows two of the test images. The triangular meshes generated from these images with the proposed technique were obtained in 7.64 (*Lenna*) and 1.26 (*House*) seconds. The Lenna image was divided into 6(H) x 6(V) tiles, considering 9($\mathcal{R}$) x 9(ζ) pixels

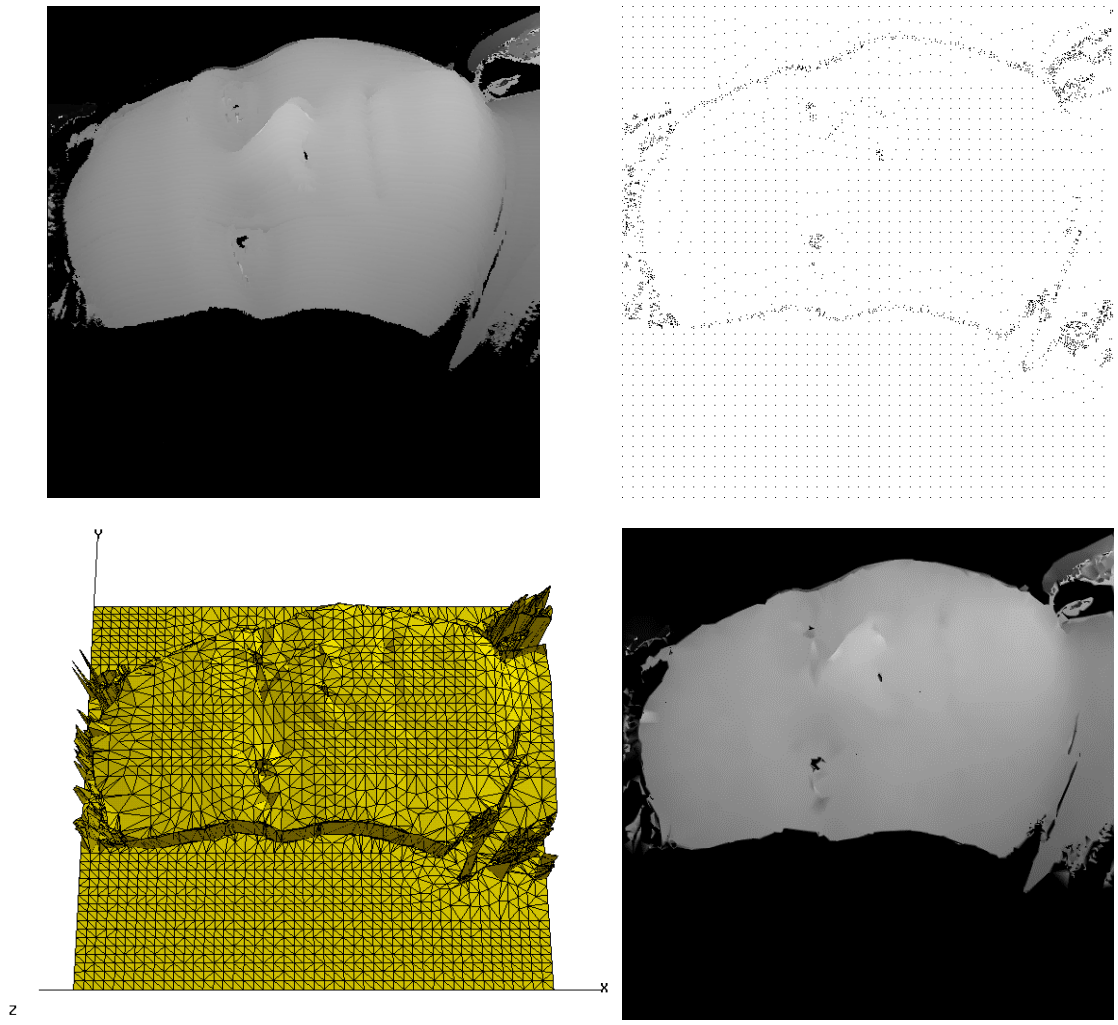

Figure 3.21. (*top left*) Original range image with 262,144 pixels (512x512). (*top right*) Set of pixels obtained with the proposed discontinuity-preserving adaptive sampling technique, (3,926 pixels). (*bottom left*) Adaptive triangular mesh generated from the previous pixels. (*bottom right*) Approximating image generated through z-buffering in 0.24 sec., RMS = 9.1.

per tile. The approximating segment length was set to 4 and the approximating error to 1. The House image was divided into 6(H) x 6(V) tiles, considering 6($\mathcal{R}$) x 6(ζ) pixels per tile. The approximating segment length was set to 4 and the approximating error to 1. These meshes contain 7,492 and 1,649 points, and 14,790 and 3,176 triangles respectively, Figure 3.22(*middle column, Lenna*) and Figure 3.22(*middle column, House*). The RMS errors of

Figure 3.22. Approximating images. (*left column*) Uniform (non-adaptive) sampling: (*2nd row*) RMS=14.3 with 7,569 points and (*4th row*) RMS=18.3 with 1,681 points. (*middle column*) Proposed technique. (*2nd row*) RMS=9.8 with 7,492 points and (*4th row*) RMS=10.6 with 1,649 points. (*right column*) Optimization-based technique (JADE). (*2nd row*) RMS=11.7 with 7,559 points and (*4th row*) RMS=15.3 with 1,623 points.

the approximating images are 9.8, Figure 3.22(*middle column, Lenna*), and 10.6, Figure 3.22(*middle column, House*).

The proposed technique produced better image approximations than both the uniform sampling technique and the optimization based technique (JADE). For example, given a similar number of points, the proposed technique always produced lower RMS errors (e.g., 9.8 and 10.6 in Figure 3.22(*middle column*)) than the uniform sampling technique (e.g., 14.3 and 18.3 in Figure 3.22(*left column*)) and than the optimization based technique (e.g., 11.7 and 15.3 in Figure 3.22(*right column*)). The reason is that the proposed technique explicitly models the discontinuities present in the image, while optimization-based techniques, such as JADE, are only able to keep those discontinuities by concentrating large numbers of points. Moreover, the CPU times necessary to generate the adaptive triangular meshes corresponding to the two previous examples were two orders of magnitude faster with the proposed technique (e.g., 7.64 sec. [*Lenna*] and 1.26 sec. [*House*]) than with JADE (e.g., 2,635 sec. [*Lenna*] and 675 sec. [*House*]).

If subsequent image processing operations were applied upon the resulting triangular meshes, the sampled pixels (mesh points) would be the only ones to be processed, whereas the same operations applied upon the original images would require the processing of all the image pixels. Additionally, if the mesh refinement steps proposed in Section 3.3.1 are applied upon the resulting triangular meshes, the previous RMS errors obtained from the approximating images could be bounded to a specified tolerance by the user.

3.4 Image Generation from Adaptive Triangular Meshes

Any compression or coding algorithm requires a corresponding decompression or decoding counterpart that allows the recovery of data in the original format. Likewise, a tool for generating digital images from adaptive triangular meshes is necessary. In this section, two techniques for generating images from adaptive triangular meshes are presented.

The first technique uniformly samples the given triangular mesh at as many positions as pixels the original image has. In this technique, each triangle of the given mesh is pro-

cessed by generating its bounding box and then by determining the pixel values which are inside it. In the second technique, the previous process is speeded up by taking advantage of graphics hardware acceleration.

The adaptive triangular meshes considered for this process are supposed to be image representations by assuming that the first two dimensions of the vertices of the mesh correspond to row and column image coordinates, and the third dimension to a property (e.g., gray level, range, elevation, ...). Those triangular meshes model a given image with the constraint that the triangles do not overlap when they are projected onto the image reference plane. Hence, each image element (pixel) has a unique corresponding value in the given triangular mesh.

3.4.1 Generation of the Approximating Image: Geometric Uniform Sampling

This section presents an algorithm for generating approximating images from $2^1/_2$D adaptive triangular meshes. The proposed algorithm uniformly samples every mesh triangle at as many positions as pixels that triangle covers. This process is described below.

A digital image (e.g., gray level image) can be generated from a given triangular mesh by considering that each triangle of the mesh is a portion of a plane that models a set of pixels. The *z* coordinates of the points contained in that plane represent, for example, gray level values in the resultant image.

The image generation process is done as follows. First, every triangle t_j of the mesh is projected onto the *xy* (image) reference plane in order to compute its bounding box. That bounding box covers a rectangular region of the sought approximating image. The pixels from that region that are contained in the projection of t_j are obtained. Finally, the pixel values (*z* coordinates) corresponding to those pixels are obtained by using the plane equation of t_j. The previous process is applied to all the triangles of the given mesh. The pixels found inside a triangle t_j are labeled such that they are not analyzed twice. The previous process is illustrated in Figure 3.23.

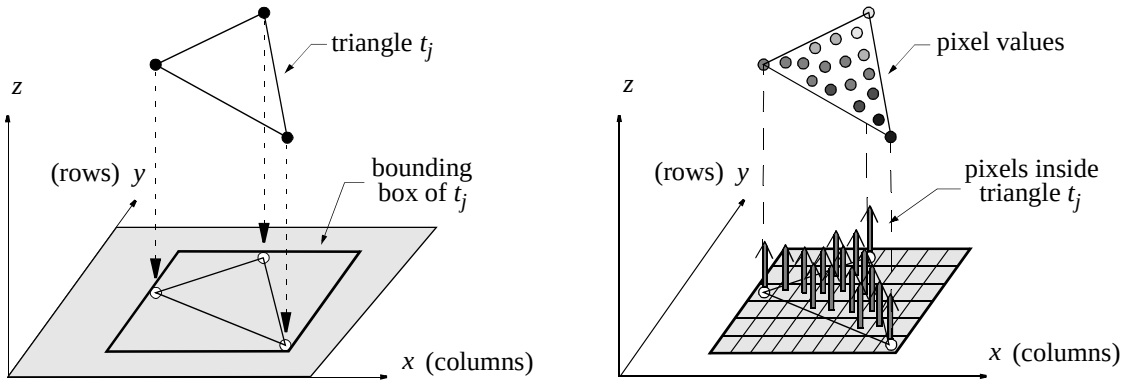

Figure 3.23. Illustration of the image generation process from $2^1/_2$D triangular meshes by using geometric uniform sampling. (*left*) Project triangle t_j onto the *xy* reference plane in order to compute its bounding box. (*right*) Find the pixels of the bounding box which are inside t_j and compute the pixel values (*z* coordinate) by using the plane equation corresponding to t_j.

The previous process was applied to the triangular meshes obtained in Section 3.3.1.5 and Section 3.3.2.3. For example, the CPU times to compute the approximating images corresponding to Lenna were: 1.78 sec. for the triangular mesh obtained through uniform sampling, Figure 3.22(*left column, Lenna*), 2.87 sec. for the triangular mesh generated with the proposed adaptive sampling technique, Figure 3.22(*middle column, Lenna*), and 3 sec. for the triangular mesh generated with JADE, Figure 3.22(*right column, Lenna*). The number of triangles corresponding to the previous triangular meshes are: 14,792, 14,790 and 14,657 respectively. Additionally, the same process was applied to the triangular meshes approximating *House*. In this case, the CPU times were: 0.41 sec. for the mesh obtained through uniform sampling (3,200 triangles), Figure 3.22(*left column, House*), 0.81 sec. for the mesh generated with the proposed technique (3,176 triangles), Figure 3.22(*middle column, House*), and 0.79 sec. for the mesh generated with JADE (3,077 triangles), Figure 3.22(*right column, House*).

The CPU times obtained with the previous process can be speeded up if hardware acceleration is used. The next section presents a second technique to generate digital images from adaptive triangular meshes by applying a z-buffering algorithm.

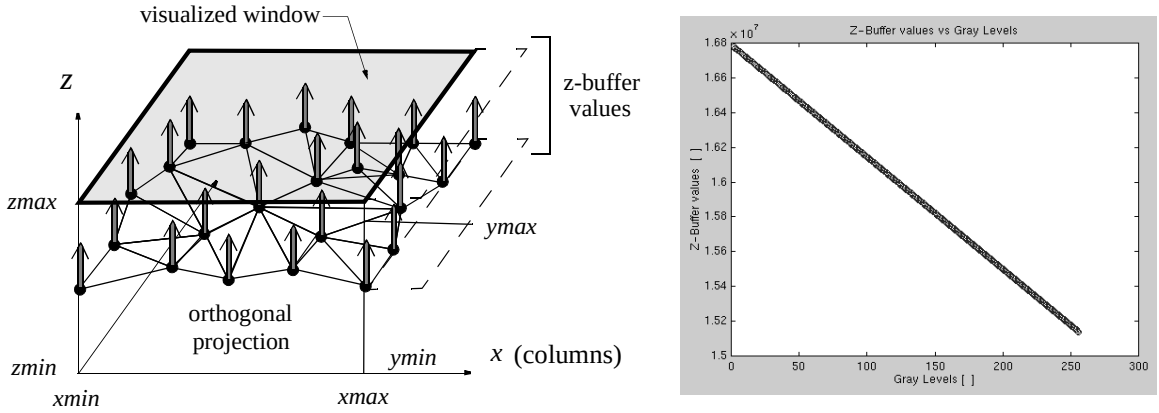

Figure 3.24. Illustration of the image generation process from triangular meshes by applying z-buffering. (*left*) Orthogonal projection of the given triangular mesh upon a specified window, in order to read the z-buffer values. (*right*) linear mapping of the z-buffer values to values in the appropriate range (e.g., [0-255] for gray level images).

3.4.2 Generation of the Approximating Image: Z-buffering

By taking advantage of the 3D nature of the processed triangular meshes, the computational cost of the previous algorithm can be reduced almost by half by applying a well-known algorithm, *z-buffering*, which is extensively utilized in computer graphics. Thus, the image generation stage can also be implemented as follows. First, the $2^1/_2$D triangular mesh is orthogonally projected and visualized in a window with the same size as the desired image through functions of the standard 3D OpenGL library. The coordinates corresponding to the clipping planes that delimit the aforementioned window are defined by the planes that define the bounding box of the given mesh in the *xy* plane (e.g., the maximum and minimum *x* and *y* coordinates of the mesh), and by the planes that delimit the maximum and minimum values of the *z* coordinates (pixel value) of the sought approximating image (e.g., 0 and 255 for 8-bit gray level images). Next, the z-buffer obtained after this visualization is read with another OpenGL function (*glReadPixels*).

Finally, the digital image is obtained by linearly mapping the values of the z-buffer to values in the appropriate range (e.g., [0-255] for 8-bit gray level images). Figure 3.24 illustrates an example of the previous process.

Since the implementations of the OpenGL library take advantage of hardware acceleration in most current computers (including PCs), the whole process turns out to be very fast. For example, the approximating images shown in Figure 3.22 were obtained by applying the proposed z-buffering process. The CPU times to obtain the approximating images corresponding to Lenna were: 0.32 sec. for the triangular mesh obtained with uniform sampling (Figure 3.22(*left column, Lenna*)), 0.32 sec. for the triangular mesh generated with the proposed technique (Figure 3.22(*middle column, Lenna*)), and 0.31 sec. for the triangular mesh generated with JADE (Figure 3.22(*right column, Lenna*)). The number of triangles corresponding to each triangular mesh were: 14,792, 14,790 and 14,657 respectively. Additionally, the CPU times to generate the approximating images for the triangular meshes representing the House were: 0.08 sec. for the mesh obtained through uniform sampling (3,200 triangles), Figure 3.22(*left column, House*), 0.08 sec. for the mesh generated with the proposed technique (3,176 triangles), Figure 3.22(*middle column, House*), and 0.07 sec. for the mesh generated with JADE (3,077 triangles), Figure 3.22(*right column, House*).

3.5 Summary

Two new techniques for generating adaptive triangular meshes from digital images have been proposed in this chapter. The obtained adaptive triangular meshes are considered to be coarse geometric representations of the given images, which model the shapes and discontinuities present in them. Additionally, two techniques to generate digital images from adaptive triangular meshes have also been presented.

The first mesh generation technique approximates a given digital image with an adaptive triangular mesh. This technique applies a coarse-to-fine approach for obtaining the required adaptive mesh. In this way, an iterative mesh refinement algorithm has been implemented. This algorithm guarantees that the maximum RMS error of the generated mesh with respect to the original image is below a specified tolerance. Thus, the proposed technique can be used in applications where the approximation error is an important parameter to be considered for the generation of the triangular mesh. The proposed technique (coarse-to-fine approach) has been compared to an optimization-based algorithm (fine-to-coarse

approach). In this case, when a triangular mesh with a high resolution (for example, a fine mesh containing all the pixels of the original image) is generated, the CPU time with the optimization-based algorithm is much lower than with the proposed technique. However, when a triangular mesh with a lower resolution (for example, a coarse mesh containing a few pixels of the original image) is generated, the proposed technique is much faster than the fine-to-coarse technique. The reason is that, if a fine mesh is required, the sought final mesh is similar to the original dense mesh used by the fine-to-coarse technique, whereas if a coarse mesh is required, the sought final mesh is similar to the original coarse mesh utilized by the coarse-to-fine technique.

The proposed approximation technique has also been compared to a uniform sampling technique. The meshes obtained with the proposed adaptive approach are more accurate than the meshes obtained with the uniform (non-adaptive) approach since the former distributes the same number of points by taking into account the curvatures of the regions contained in the given image. A drawback of the proposed technique is that image edges induce an oversampling of the original image every time the mesh refinement algorithm is applied.

The second mesh generation technique approximates a digital image with an adaptive triangular mesh by preserving the shapes and discontinuities present in the image, avoiding optimization criteria. This technique utilizes a preprocessing stage in order to detect and separately approximate the edges present in the original image. Hence, it avoids the drawback of the first technique. Initially, the algorithm generates an adaptive sampling of the given image by choosing a set of pixels from it. The obtained pixels are chosen by taking into account the edges contained in the image and regions comprised among them. Each image edge is approximated by a pair of polylines. Finally, the sampled pixels are triangulated, using the obtained polylines as constraints for the triangulation. Although the proposed technique does not ensure a maximum approximation error with respect to the original image, the quality of the generated triangular meshes is good enough. Moreover, this second mesh generation technique can be complemented with the first one if a maximum error must be guaranteed.

The proposed technique has also been compared to the previous fine-to-coarse optimization-based technique. The adaptive triangular meshes generated with the proposed algorithm are obtained much faster than with optimization-based algorithms and, since image discontinuities are explicitly handled, the results are better than the ones obtained through both uniform (non-adaptive) sampling and optimization-based algorithms.

Besides being able to generate triangular meshes from digital images, it is also necessary to proceed on the other way round in order to determine the accuracy with which the obtained meshes approximate the original images. Hence, two techniques to generate digital images from adaptive triangular meshes have also been proposed. The first technique uniformly samples every triangle of the given mesh at as many positions as pixels that triangle covers. The result is an approximating digital image. The second technique generates a digital image from a given adaptive triangular mesh by applying functions of the OpenGL graphics library, which are implemented by taking advantage of hardware acceleration. In this way, the image generation stage turns out to be significantly faster than the previous geometric uniform sampling technique.