# Chapter 5

# *Geometric Processing of Adaptive Triangular Meshes: Enhancement Operations*

Chapter 5 presents several techniques to apply typical image enhancement operations upon $2^1/_2$D adaptive triangular meshes. These techniques allow to process a triangular mesh in order to obtain a certain improvement of that mesh. The proposed enhancement operations are introduced in Section 5.1. Section 5.2 describes a set of gray level modification operations upon adaptive triangular meshes. Section 5.3 describes two techniques to apply mean and median filters to adaptive triangular meshes. The contents of the chapter are finally summarized in Section 5.4.

As it was mentioned above, the $2^1/_2$D triangular meshes processed with the proposed techniques are considered to be projectable onto their *xy* reference planes.

## 5.1 Introduction

Image enhancement techniques are intended to modify digital images in order to improve their visualization or to allow the extraction of more information from them. This is done by removing blurring and noise, increasing contrast or revealing details that may be hidden in the original images.

The types of techniques typically used for image enhancement are *point* and *neighbor operations*. Point operations allow to modify every image pixel according to a particular equation that is not dependent on other pixel values. On the other hand, neighbor operations allow to modify each image pixel according to the values of its neighbors.

This chapter describes a set of techniques to perform typical image enhancement operations upon $2^1/_2$D triangular meshes. Taking advantage that the processed triangular meshes may be compact representations of digital images, those operations can be performed more efficiently in the geometric domain than if they were performed in the image domain, by sequentially processing all the pixels contained in the digital images.

## 5.2 Modification Operations upon Triangular Meshes

This section describes several techniques to apply image modification operations to $2^1/_2$D triangular meshes. These operations belong to the category of point operations and proceed by applying a *mapping equation* that is typically linear and modifies the $z$ coordinates of the points that constitute the triangular mesh.

The triangular meshes shown in this section were generated with the algorithm described in Section 3.3.2. The developed techniques include: gray-scale and histogram modification operations.

### 5.2.1 Gray-Scale Modification

Within the image processing field, the gray-scale modification operation allows to compress or stretch the gray-scale of an image by using a specified mapping equation. Typically, a gray-scale range is compressed when it is of little interest, whereas a gray-scale range is stretched in order to highlight imperceptible details present in the original image.

The proposed gray-scale modification technique consists of three main stages. The first stage dissects a given triangular mesh with two planes parallel to the mesh *xy* reference plane. The points of the given mesh that are contained between those planes define the range of *z* values which must be mapped. The second stage maps that range of *z* values by means of a specific mapping equation. Finally, the resulting triangular mesh is obtained in the third stage.

#### *5.2.1.1 Double Dissection of a Triangular Mesh*

Given a triangular mesh *M*, a double dissection process is applied. Each mesh dissection is performed with a plane parallel to the $z = 0$ reference plane, by using the dissection algorithm proposed in the thresholding operation (Section 4.2.2). Each dissection plane intersects the *z* axis at a value that is specified by the user. Those planes represent the *inferior dissection plane* and *superior dissection plane* respectively.

Although the order is indifferent, the dissection algorithm is first applied with the superior dissection plane and then with the inferior dissection plane. In both cases, a set of intersection points is obtained. This dissection procedure is similar to the one applied by the quantization algorithm (Section 4.2.3.1) but, in this case, two dissection planes are only considered.

#### *5.2.1.2 Mapping of z Coordinate Values*

Once the previous dissection process has been applied, the algorithm labels the points of the given mesh which are contained between the inferior and superior dissection planes. Those points correspond to the range of *z* values that must be modified. The *z* coordinate of every labeled point is changed by using a linear equation. This equation is specified by the user. An example that shows how the *z* values are changed is illustrated in Figure 5.1(*left*).

Additionally, the vertices with *z* coordinates that are above the superior dissection plane and below the inferior dissection plane can also be modified by applying a similar

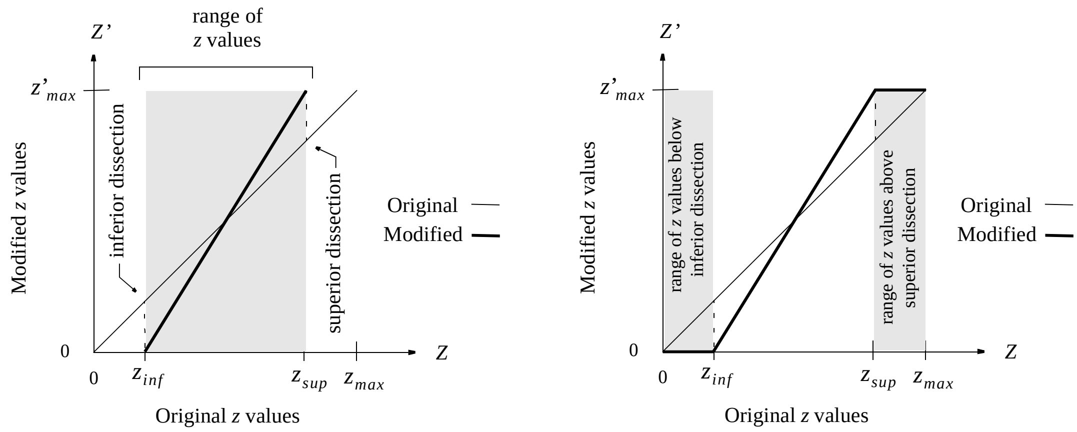


Figure 5.1. Gray-scale modification. (*left*) Mapping of the range of $z$ values that are contained between the inferior and superior dissection planes to modified $z'$ values (thick line). (*right*) Additionally, the range of $z$ values that are above and below the superior and inferior dissection planes can also be mapped to specified $z'$ values.

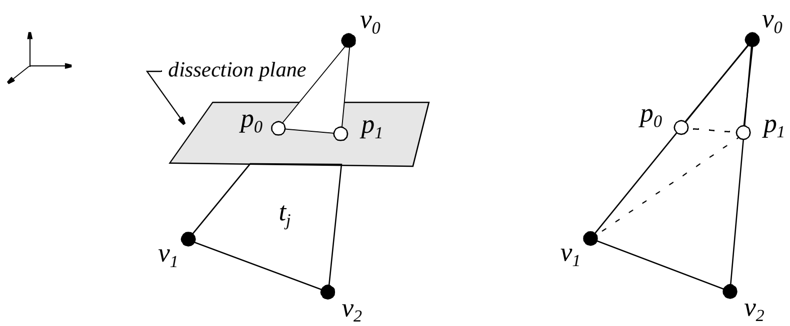


Figure 5.2. Retriangulation process of a dissected triangle. (*left*) Triangle $t_j$ is dissected at the points $p_0$ and $p_1$ by a horizontal plane. (*right*) Retriangulation of triangle $t_j$ after it has been dissected.

procedure. In this case, both ranges of $z$ values are mapped using two new specified linear equations. Figure 5.1(*right*) shows an example of this procedure.

#### *5.2.1.3 Triangular Mesh Generation*

The procedure applied in this stage is similar to the one utilized for the quantization operation (Section 4.2.3.2). The objective consists of applying a retriangulation process to the dissected triangles, such as it is shown in Figure 5.2. The topology of the non-dissected triangles must be kept.

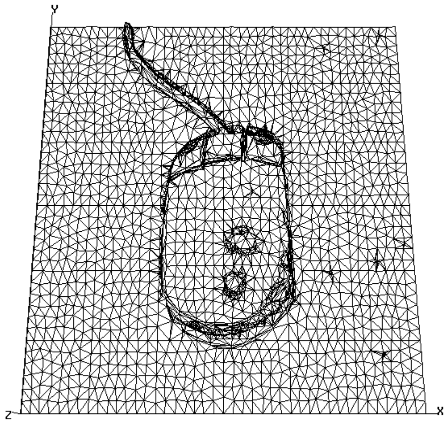


Figure 5.3. (*left*) Triangular mesh to be processed with the proposed gray-scale modification algorithm. This mesh contains 2,624 points and approximates an 8-bit image. (*right*) Approximating image with 262,144 pixels (512x512) generated from the previous mesh.

Since only the dissected triangles are divided and retriangulated, the topology of the non-dissected triangles is kept.

#### *5.2.1.4 Experimental Results*

This section presents experimental results obtained with the proposed gray-scale modification operation. This operation was run upon various adaptive triangular meshes which were generated with the algorithm described in Section 3.3.2. The CPU times to perform this operation were measured and compared with the times to perform similar operations with CVIPtools, a conventional image processing software which is publicly available (Umbaugh, 1998). The CPU times were measured on a SGI Indigo II with a 175MHz R10000 processor.

Figure 5.3(*left*) shows a triangular mesh that approximates an 8-bit image. The range of *z* values between [0, 50] of this mesh was mapped to a single value equal to 20, such as it is illustrated in Figure 5.4(*top left*). The CPU time to perform that mapping in the geometric domain was 0.12 sec., while the same operation applied to the original 8-bit image by using CVIPtools (Umbaugh, 1998) took 0.16 sec. in the image domain. Figure 5.4(*top right*)

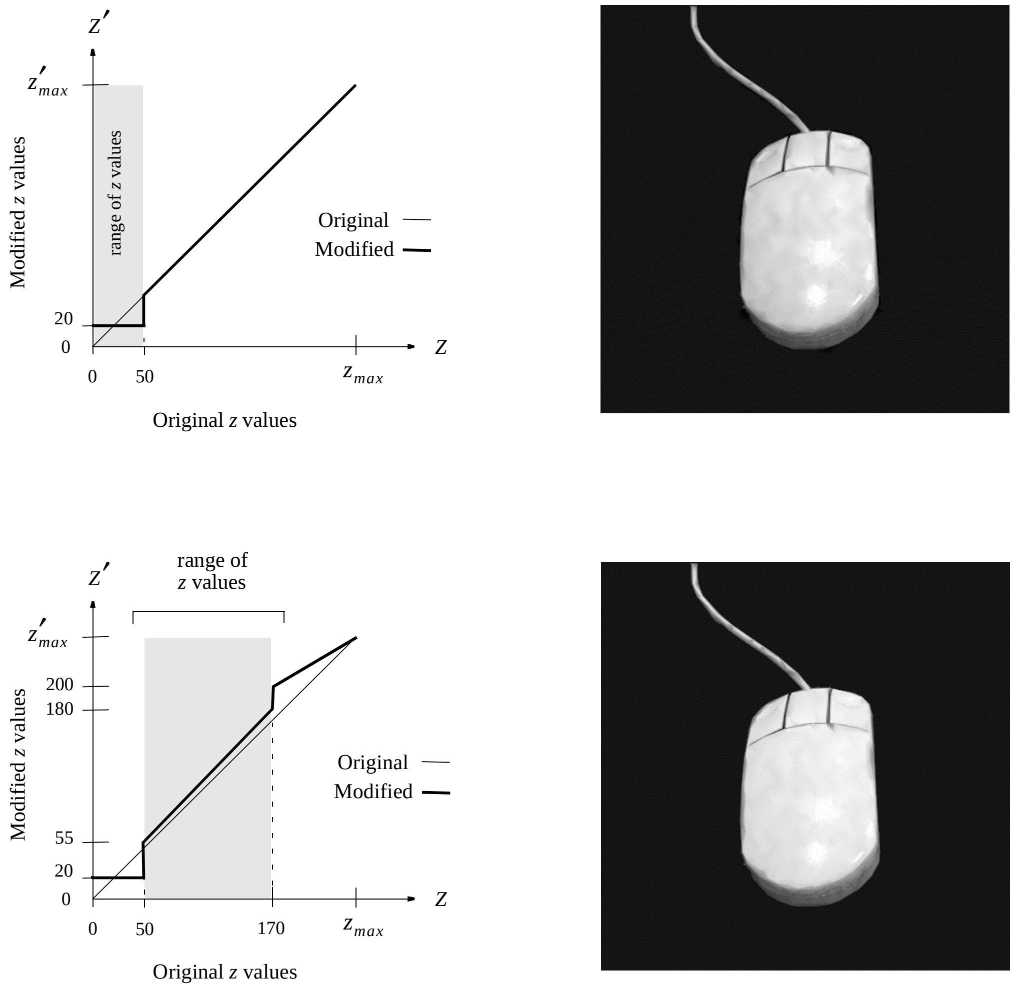


Figure 5.4. (*left column*) Mapping the range of *z* values to specified $z'$ values: (*top*) only the range [0, 50] is mapped to the $z' = 20$, (*bottom*) the values outside that range are also mapped: range [0, 50] is mapped to 20, range [51, 170] to range [55, 180] and range [171, 255] to range [200, 255]. (*right column*) Approximating images generated from the resulting triangular meshes obtained after applying the gray-scale modification.

shows the approximating image obtained from the final triangular mesh generated after applying the proposed gray-scale modification algorithm.

We have also considered the case of mapping the range of *z* values that are above and below the superior and inferior dissection planes respectively. In this case, the range of *z* values between the dissection planes was [51, 170] and was mapped to range [55, 180]. On

Figure 5.5. Gray-scale modification with CVIPtools. (*left*) Mapping the range of intensity corresponding to [0, 50] to 20. (*right*) Mapping both the range of intensity [50, 170] and the intensity values outside that range to the values specified in Figure 5.4(*bottom left*).

the other hand, the values outside that specified range were modified as follows: range [0, 50] was mapped to a single value equal to 20, and range [171, 255] was mapped to range [200, 255]. Figure 5.4(*bottom left*) shows those values. The CPU times were 0.24 sec. with the proposed technique and 0.44 sec. with CVIPtools. The approximating image generated from the resulting mesh is shown in Figure 5.4(*bottom right*). Figure 5.5 shows equivalent modification operations applied to the original image with CVIPtools (Umbaugh, 1998), which processes the given image pixel by pixel.

### 5.2.2 Histogram Modification

This section describes three enhancement algorithms that modify the distribution of the $z$ values of a given triangular mesh by applying a mapping function that either stretches, shrinks or slides the histogram corresponding to that mesh. The histogram of a mesh can be computed with the technique described in Section 4.6.

#### *5.2.2.1 Histogram Stretching*

The histogram stretching process computes a new $z$ coordinate for every vertex of the given triangular mesh by applying the following equation (Umbaugh, 1998):

$$z' = \frac{z - Z_{min}}{Z_{max} - Z_{min}}(Z_{MAX} - Z_{MIN}) + Z_{MIN} \tag{5.1}$$

where $Z_{max}$ and $Z_{min}$ are the largest and smallest *z* coordinates of the given mesh, while $Z_{MIN}$ and $Z_{MAX}$ are the maximum and minimum values corresponding to the current pixel depth, for example, 255 and if the given triangular mesh represents an 8-bit image.

Figure 5.7(*left column*) shows the histogram and approximating image obtained after applying the stretching procedure to the triangular mesh shown in Figure 5.6(*top row, left*).

#### *5.2.2.2 Histogram Shrinking*

The histogram shrinking process consists of applying the same previous equation to all the vertices of the triangular mesh, although, in this case, $Z_{MIN}$ and $Z_{MAX}$ are substituted for both the minimum and maximum values desired for the compressed histogram, $Shrink_{max}$ and $Shrink_{min}$ respectively:

$$z' = \frac{z - Z_{min}}{Z_{max} - Z_{min}}(Shrink_{max} - Shrink_{min}) + Shrink_{min} \tag{5.2}$$

An example that shows the final histogram and its approximating image after applying the shrinking procedure is given in Figure 5.7(*middle column*), considering the triangular mesh shown in Figure 5.6(*top row, middle*).

#### *5.2.2.3 Histogram Sliding*

Finally, the mapping function for the histogram sliding process is defined as (Umbaugh, 1998):

$$z' = z + offset \tag{5.3}$$

with the *offset* being the displacement applied to the histogram associated with the original triangular mesh.

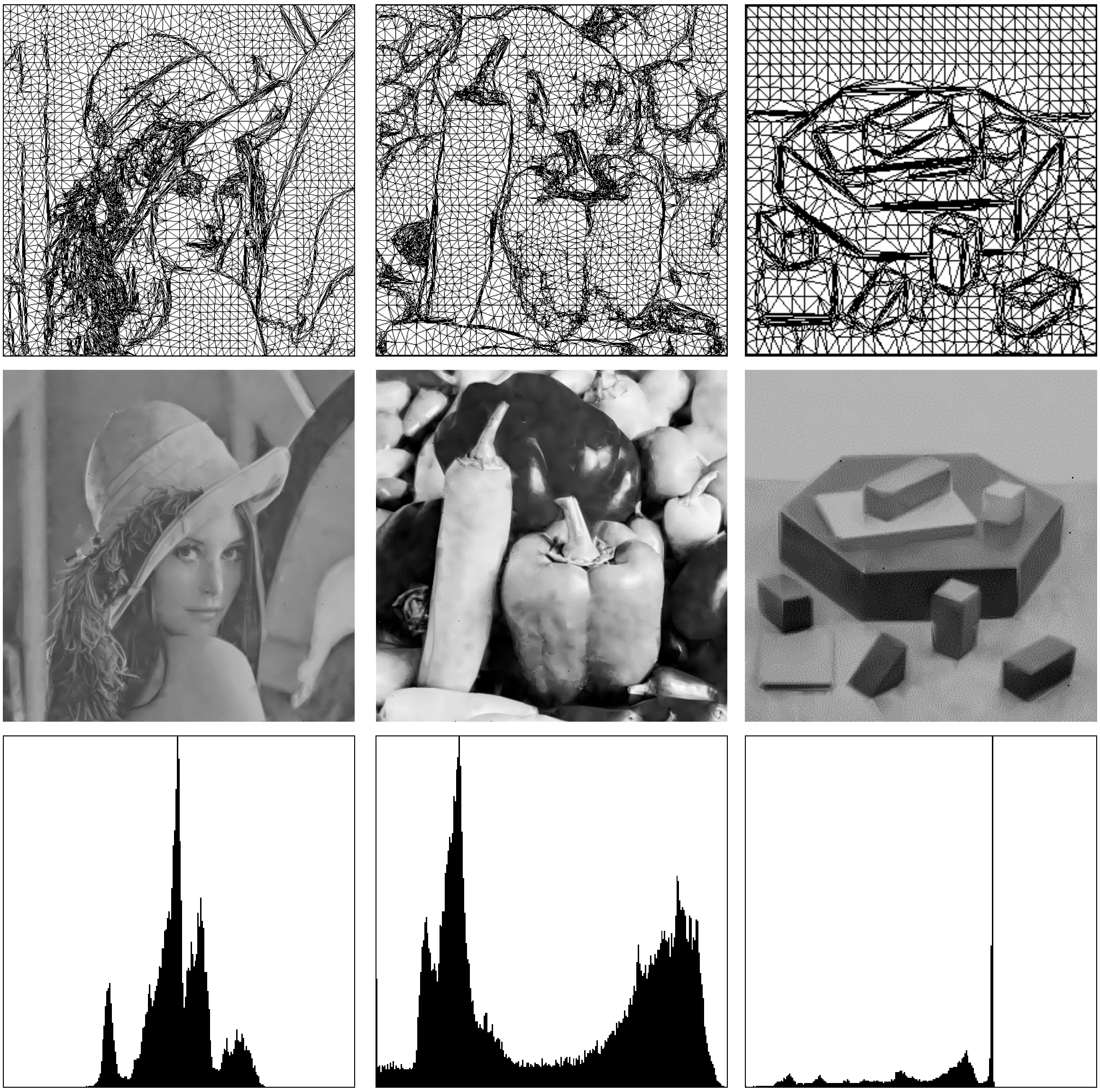

Figure 5.6. (*top row*) Adaptive triangular meshes: (*left*) 7,492, (*middle*) 7,370 and (*right*) 1,593 points. (*middle row*) Approximating images from the previous meshes: (*left*) 512x512 (*middle*) 512x512 and (*right*) 256x256 pixels. (*bottom row*) Histograms computed from the given meshes.

Figure 5.7(*right* column) shows the histogram and final approximating image corresponding to the application of the sliding process to the triangular mesh shown in Figure 5.6(*top row, right*).

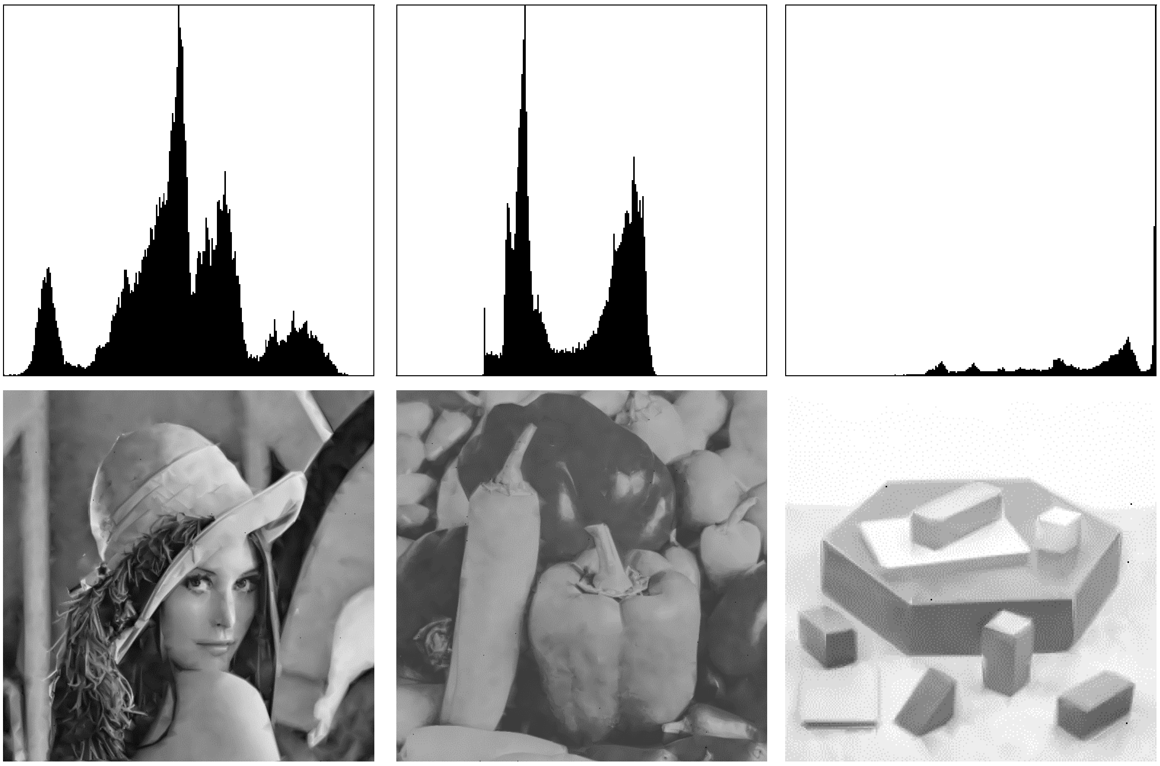

Figure 5.7. Histogram modification with the proposed technique. (*top row*) Histograms after applying: (*left*) stretching (*middle*) shrinking and (*right*) sliding. (*bottom row*) Approximating images after: (*left*) stretching, (*middle*) shrinking to the range [60,180] and (*right*) sliding.

The CPU time for both the stretching and shrinking operations in the previous examples was 1.3ms. Qualitatively similar results were obtained with CVIPtools in 190ms. The sliding process took 0.1ms with the proposed technique and 10ms with CVIPtools. Figure 5.8 shows the results obtained by applying the three image enhancement processes with CVIPtools.

#### *5.2.2.4 Experimental Results*

The proposed operations have been tested upon several triangular meshes obtained from 8-bit images. The CPU times have been measured on a SGI Indigo II workstation with a 175MHz R10000 processor. Results for four of these images are presented in this section: House (256x256x8), Lenna (512x512x8), Peppers (512x512x8) and Blocks (256x256x8).

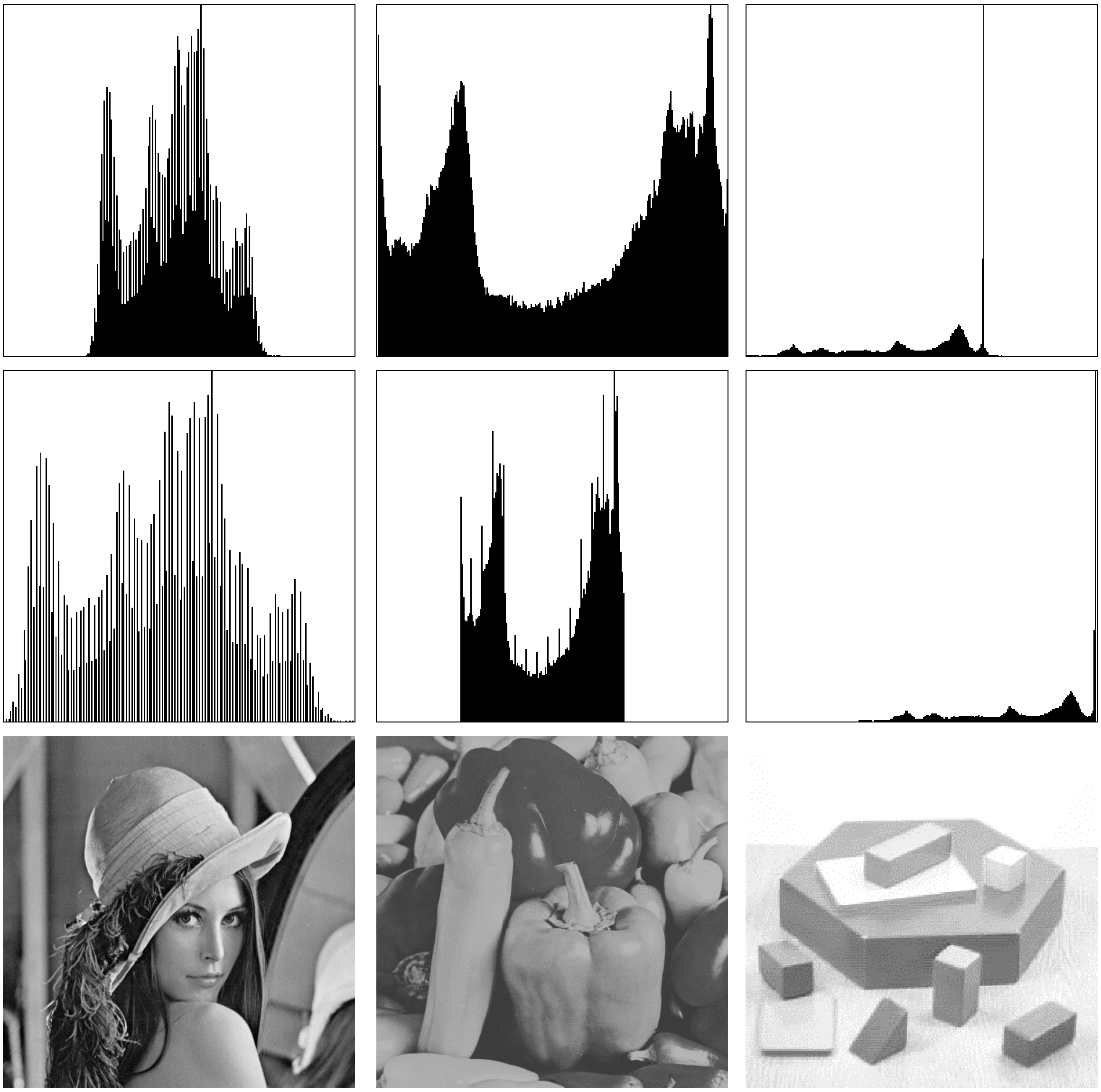

Figure 5.8. Histogram modification with CVIPtools. (*top row*) Histograms computed from the input images. (*middle row*) Histograms after applying: (*left*) stretching (*middle*) shrinking and (*right*) sliding. (*bottom row*) Resulting images after: (*left*) stretching, (*middle*) shrinking to the range [60,180] and (*right*) sliding.

These images were approximated with adaptive triangular meshes obtained by applying the technique presented in Section 3.3.2. These meshes contain: (House) 1,649, (Lenna) 7,492, (Peppers) 7,370 and (Blocks) 1,593 points respectively. They are shown in Figure 5.6(*top row*) and Figure 5.9(*middle*).

| Technique | Image | Stretch | Shrink | Slide |
|---|---|---|---|---|
| Proposed | House | 0.00024 | 0.00024 | 0.00014 |
| CVIPtools | House | 0.05 | 0.05 | 0.01 |
| Proposed | Lenna | 0.0013 | 0.0013 | 0.00057 |
| CVIPtools | Lenna | 0.19 | 0.19 | 0.04 |
| Proposed | Peppers | 0.0013 | 0.0013 | 0.00057 |
| CVIPtools | Peppers | 0.19 | 0.19 | 0.04 |
| Proposed | Blocks | 0.00024 | 0.00024 | 0.00014 |
| CVIPtools | Blocks | 0.05 | 0.05 | 0.01 |

**Table 1:** CPU times in seconds corresponding to the histogram modification operations with the proposed technique and with CVIPtools**.**

The same histogram modification operations have been applied to the original 8-bit images by using the public image processing library CVIPtools (Umbaugh, 1998). Table 1 shows the CPU times for computing the stretching, shrinking and sliding operations with both the proposed technique and CVIPtools.

In the previous examples, the execution time is two orders of magnitude faster when working upon triangular meshes than when the original images are processed. The obvious reason is that, in the first case, only the points of the triangular meshes must be sequentially processed while, in the second case, all the pixels of the 8-bit images must be considered.

## 5.3 Filtering Operations upon $2^1/_2$D Triangular Meshes

Two techniques for filtering digital images approximated with adaptive triangular meshes are described in this section. Contrarily to the previous modification operations, these operations belong to the category of neighbor operations, since they allow the smoothing of each point of the given mesh by considering that point and its neighbors.

As it was mentioned in Chapter 3, the adaptive triangular meshes representing digital images are defined in a 3D space in which coordinates *x* and *y* represent the columns and rows of the image space, and coordinate *z* represents the pixel depth. Every point of a triangular mesh is associated with a *polygonal patch*, the latter being the set formed by that point

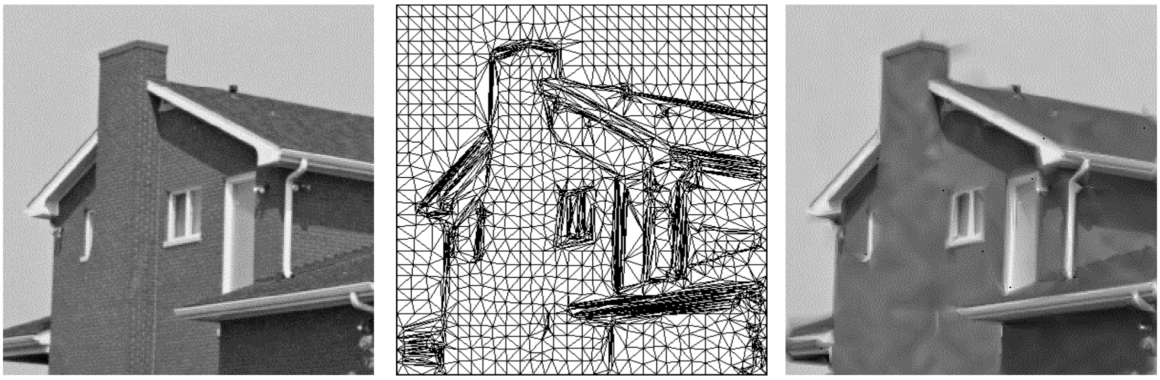

Figure 5.9. (*left*) Original image with 65,536 pixels (256x256). (*middle*) Adaptive triangular mesh that approximates the original image with 1,649 points. (*right*) Approximating image obtained from the previous mesh.

and its adjacent neighbors sorted in counter-clockwise order. Two points of the mesh are adjacent if there exists an edge between them.

The proposed filtering techniques compute a new triangular mesh from the given one. The new mesh is the same as the original one except for the *z* coordinates of its points. In particular, the *z* coordinate of each point *P* of the new mesh is computed as a function of the original *z* coordinates of the points contained in the polygonal patch associated with *P*. This process mimics the classical convolution technique utilized to filter digital images. Both a mean and a median filter have been implemented.

### 5.3.1 Mean and Median Filters

The mean filter computes the *z* coordinate of every point *P* of the new mesh as the average of the original *z* coordinates of the points that belong to the polygonal patch centered at *P*. On the other hand, the median filter sorts out the *z* coordinates of every polygonal patch. The new *z* coordinate of each point whose associated sorted list has an even number of values is the average of the two values in the middle of that list. If the number of values in the sorted list is odd, the new *z* coordinate is just the value in the middle of the list.

Figure 5.9 shows: (*left*) an original 8-bit image, (*middle*) an adaptive triangular mesh that approximates this image, and (*right*) the approximating image obtained from that mesh

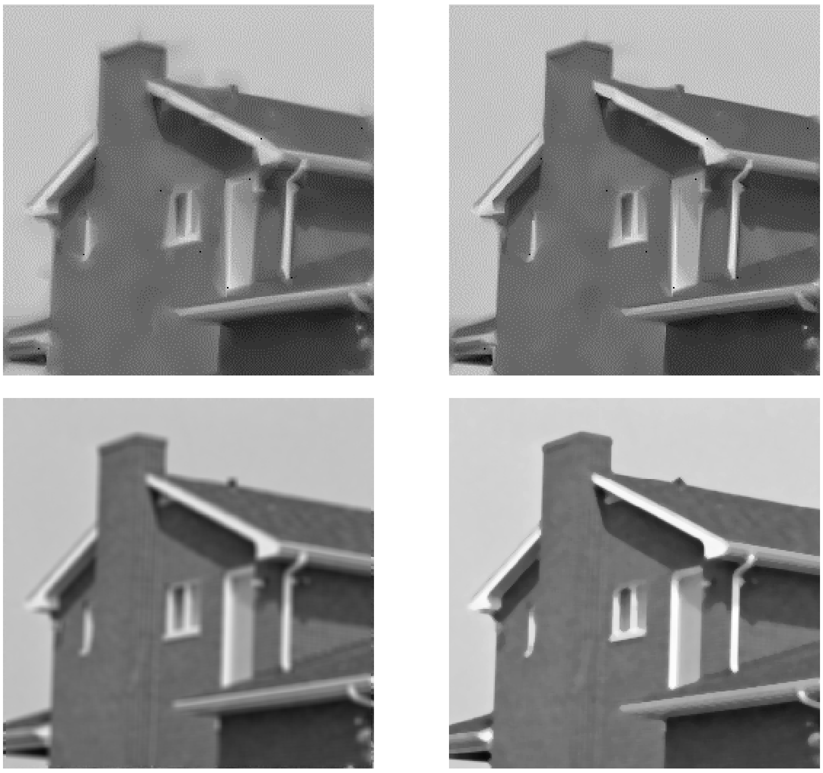

Figure 5.10. (*top row*) Filtered images with the proposed technique: (*left*) mean filter, (*right*) median filter. (*bottom row*) Filtered images with CVIPtools: (*left*) mean filter, (*right*) median filter.

through a sampling algorithm described in Section 3.4.2. Figure 5.10 shows the approximating images corresponding to the triangular meshes obtained by applying the proposed (*top left*) mean and (*top right*) median filters to the mesh shown in Figure 5.9. Figure 5.10(*bottom row*) shows equivalent filters applied to the original image with a conventional image processing algorithm, CVIPtools (Umbaugh, 1998), which processes the given image pixel by pixel.

The original triangular mesh was filtered in 0.8ms (mean) and 3.8ms (median) with the proposed technique. Similar filters applied to the original image with CVIPtools took 290ms (mean) and 170ms (median).

| Technique | Image | Mean Filter | Median Filter |
|---|---|---|---|
| Proposed | House | 0.0008 | 0.0038 |
| CVIPtools | House | 0.29 | 0.17 |
| Proposed | Lenna | 0.0123 | 0.0255 |
| CVIPtools | Lenna | 1.25 | 0.66 |
| Proposed | Peppers | 0.0123 | 0.025 |
| CVIPtools | Peppers | 1.25 | 0.66 |
| Proposed | Blocks | 0.0008 | 0.0037 |
| CVIPtools | Blocks | 0.29 | 0.17 |

**Table 2:** CPU times in seconds of filtering operations with the proposed technique and with CVIPtools**.**

### 5.3.2 Experimental Results

Several triangular meshes that represent 8-bit images have been processed with the proposed filtering operations. The CPU times have been measured on a SGI Indigo II workstation with a 175MHz R10000 processor.

Results for 4 of these images are presented in this section: House (256x256x8), Lenna (512x512x8), Peppers (512x512x8) and Blocks (256x256x8). These images were approximated with adaptive triangular meshes obtained by applying the technique presented in Section 3.3.2. These meshes contain: (House) 1,649, (Lenna) 7,492, (Peppers) 7,370 and (Blocks) 1,593 points respectively. They are displayed in Figure 5.6(*top row*) and Figure 5.9(*middle*).

The same filtering operations have been applied to the original images by using the public image processing library CVIPtools (Umbaugh, 1998). Table 2 shows the CPU times for computing the mean and median filtering operations with both the proposed technique and with CVIPtools (the latter with 5x5 masks).

As it occurred with the previous histogram modification operations, the execution upon triangular meshes is two orders of magnitude faster when working upon triangular meshes than when the original images are processed. The obvious reason is that, in the first case, only the vertices of the triangular meshes must be sequentially processed while, in the second case, all the pixels of the 8-bit images must be considered.

## 5.4 Summary

This chapter presents basic algorithms to perform image modification and filtering operations directly working upon $2^1/_2$D adaptive triangular meshes that represent digital images or terrain surfaces. The image modification operations include a gray-scale modification algorithm and a histogram modification algorithm. On the other hand, within the filtering operations, a mean and a median filter have also been implemented.

The gray-scale modification algorithm is composed of three stages. The first stage dissects a given triangular mesh with two planes parallel to the *xy* reference plane of the mesh. Each dissection plane intersects the *z* axis at a value that is defined by the user. The points of the triangular mesh which are contained between those dissection planes define the range of *z* values that must be mapped. The second stage consists of mapping the range of *z* values by using a linear mapping equation. Finally, the resulting triangular mesh is obtained in the third stage.

The histogram modification operation consists of three enhancement algorithms that modify the distribution of the *z* values of a given triangular mesh by applying a mapping function that stretches, shrinks or slides the histogram corresponding to that mesh.

Finally, the mean and median filters consist of computing a new triangular mesh from the given one. The new mesh is the same as the original one except for the *z* coordinates of its points. In this case, each point of the original mesh is associated with a polygonal patch, the latter being the set formed by that point and its adjacent neighbors sorted in counterclockwise order. The *z* coordinate of each point *P* of the new mesh is computed as a function of the original *z* coordinates of the points contained in the polygonal patch associated with *P*. Hence, the mean filter computes the *z* coordinate of every point *P* in the new mesh as the average of the original *z* coordinates of the points that belong to *P*'s polygonal patch, while the median filter sorts out the *z* coordinates of every polygonal patch. The new *z* coordinate of each point, whose associated sorted list has an even number of values, is the average of the two values in the middle of the list. If the number of values in the sorted list is odd, the new $z$ coordinate is just the value in the middle of the list.

Experimental results show that the implemented operations run significantly faster in the geometric domain than by sequentially processing all the pixels contained in the respective original images.