# CHAPTER 5

# LOCAL OPERATORS

After initial triangulation, there is a need to modify the mesh in such a way that the quality of the reconstructed image will improve or the mesh complexity will be reduced to achieve an optimal output. This chapter starts with an overview of the algorithm proposed by Dr. Lee to convert height field data into a triangular mesh. It will be followed by a new algorithm, which is adapted for gray-scale image data. Next, the chapter will present the method for region selection, which is necessary for triangle splitting and neighborhood assignment, which helps the edge-swapping operation faster. Eventually, the chapter will end with details about local operators that will be used for mesh simplification as well as image enhancement.

## 5.1 High-Level Algorithm

Since the approximation algorithm proposed by Dr. Lee is designed for height field data, it can be unsuitable for some images, which have slightly different characteristics from height field data. First of all, natural scene images tend to contain large changes in intensity while it is normally assumed that sharp spikes can hardly be found in terrain data. Secondly, geographical data usually does not contain clusters of spikes located near each other. However this is not the case for natural images especially for texture images where dark and white pixel data can randomly occur very close together. The major part of the algorithm in this thesis is still based on Dr. Lee's algorithm except for some minor adaptation. This will be discussed in Section 5.1.2.

### 5.1.1 SML High-Level Algorithm

In his dissertation, Dr. Lee performed initial triangulation based on wavelet coefficients as described in Chapter 4 as the first step. Next he performed *M-1* iterations of refinement, regularization and mesh reduction. The refinement includes candidate selection for splitting, neighborhood assignment for triangle pairing, edge swapping and shape-preserving splitting.

Finally, regularization and mesh reduction are used to decrease mesh complexity by removing redundant vertices.

The next step is to select candidates for refinement based on the magnitude of the wavelet coefficients. These candidate triangles are referred to as 0-neighbor triangles. This will be presented in Section 5.2.1. Next, triangles are assigned 1-neighborhood and 2-neighborhood status. This operation is known as neighborhood assignment and will be presented in Section 5.2.2. After this, the triangles that have no current status will be considered as 3-neighborhood (non-neighborhood). This is necessary for the convergence of the splitting and edge swapping operations. In splitting, triangles are paired only among 0-neighborhood or 1-neighborhood triangles while edge swapping can be swapped only among triangles that have 0 to 2-neighborhood status. This helps the algorithm to split only on the necessary non-smoothed triangles as well as limit the swapping area to 2-neighborhood triangles. Some algorithms use the fan-area limitation [Lu, Le and Yun 00]. Edge swapping and splitting operations are presented in Sections 5.4.1 and 5.4.3 respectively. Finally, mesh reduction, which consists of vertex removal and edge collapse operators, are used to remove redundant vertices and edges respectively. These operations will be shown in Sections 5.3.1 and 5.3.2. These mesh simplification operations try to reduce the mesh complexity as much as possible while, at the same time, try not to significantly degrade the quality of the image. Figure 5.1 shows the flow diagram of the approximation algorithm proposed by Dr. Lee. Its high-level algorithm is shown.

```
1.  Decompose two-dimensional data with discrete wavelet transform
2.  Perform initial triangulation from basic and variant templates
    a) Evaluate wavelet detail coefficients at coarsest resolution level $M$
    b) Assign basic and dual template labels for each region
    c) Tessellate square regions with variant templates
3.  For level = $M$ to 0
    a) Refinement and regularization
       For $k$ = 1 to 3
           i.   Calculate detailed energy for region candidate selection
           ii.  Perform neighborhood assignment
           iii. Refine by shape-preserving splitting
           iv.  Regularize with edge swapping
    b) Mesh simplification (vertex removal and edge collapse)
```

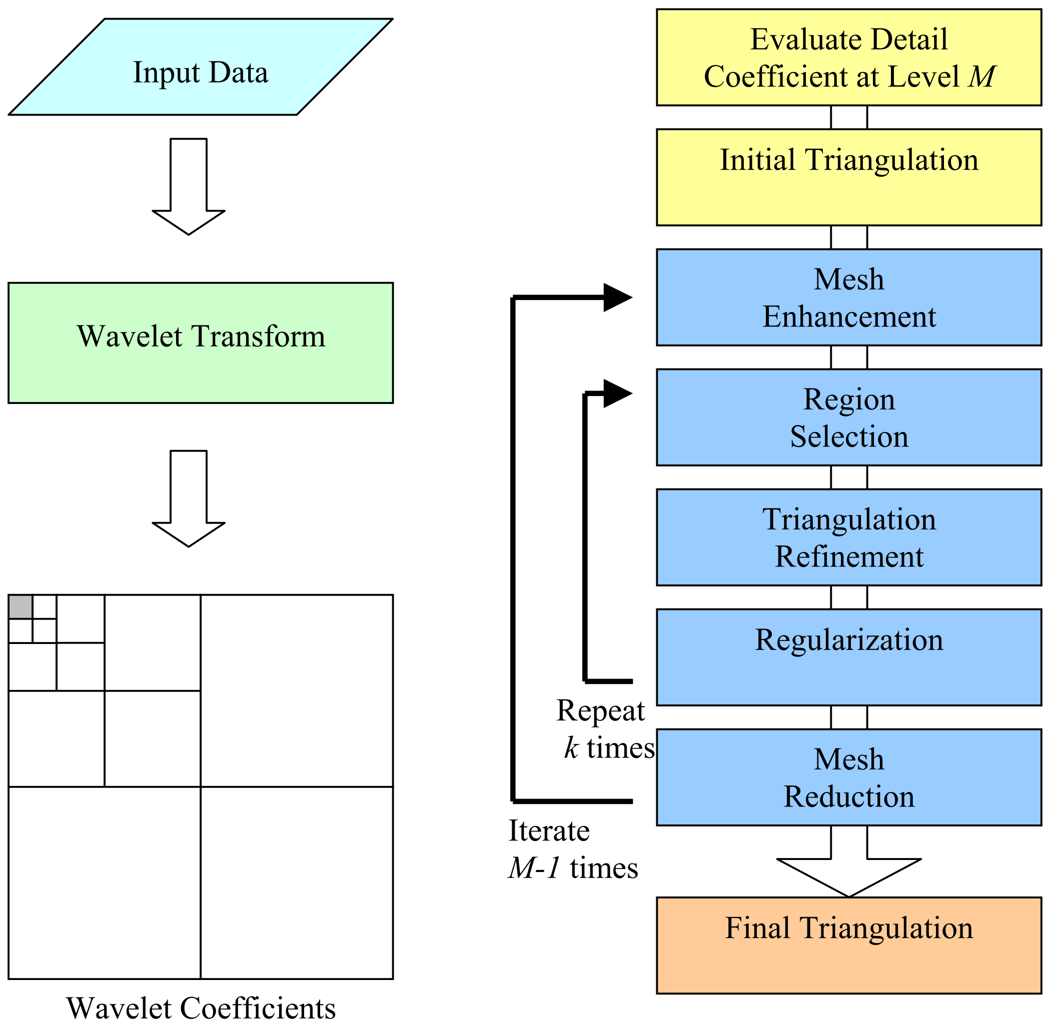

Figure 5.1 – Flow diagram of SML algorithm.

### 5.1.2 New Approach

In this alternative approach, most of the work is based on Dr. Lee's work except some minor changes and modifications for the image data. These modifications include the use of error-based candidate selection for the last three iterations in the algorithm and wavelet-coefficient checking for vertex removal operation. Since the wavelet transform could not detect all the edges in the image, it is necessary to use an error-based scheme to refine the undetected regions. Furthermore, after some refinement, most of the vertices are found to be located along the edges, where large wavelet coefficients lie. It would be an advantage to use this information to check for flatness of the group of triangles that shares the same vertex. Next, this algorithm introduces a wavelet attraction operation, which is used to move vertices to locations where large wavelet coefficients can be found. This operation is presented in Section 5.4.2. Finally, to remove vertex redundancy, a short edge removal operation is used to reduce the mesh complexity. This will be discussed in Section 5.3.2. The only difference in the algorithm is in step 3. Its high level algorithm is shown below while Figure 5.2 shows its equivalent flow diagram.

```
1.    Decompose two-dimensional data with Discrete Wavelet Transform
2.    Perform Initial Triangulation from basic and variant templates
      a) Evaluate Wavelet Detail Coefficients at coarsest resolution level $M$
      b) Assign basic and dual template label for each region
      c) Tessellate square regions with variant templates
3.    For level = $M$ to -2
      a) Candidate Region Selection
         If $M$>=0
                 Calculate wavelet detailed energy for region candidate selection
         Else
                 Calculate error energy for region candidate selection
      b) Perform neighborhood assignment
      c) Refine by shape-preserving splitting
      d) Regularize with edge swapping with $\alpha_1$
      e) Mesh simplification with vertex removal and edge collapse operations
      f) Regularize with edge swapping with $\alpha_2$
```

## 5.2 Region Selection

Region selection is important to the algorithm in that, first, it helps to select appropriate triangles to be used for splitting and, second, it limits the area for edge swapping so that this operation will converge in a short period of time. This operation is composed of candidate selection and neighborhood assignment.

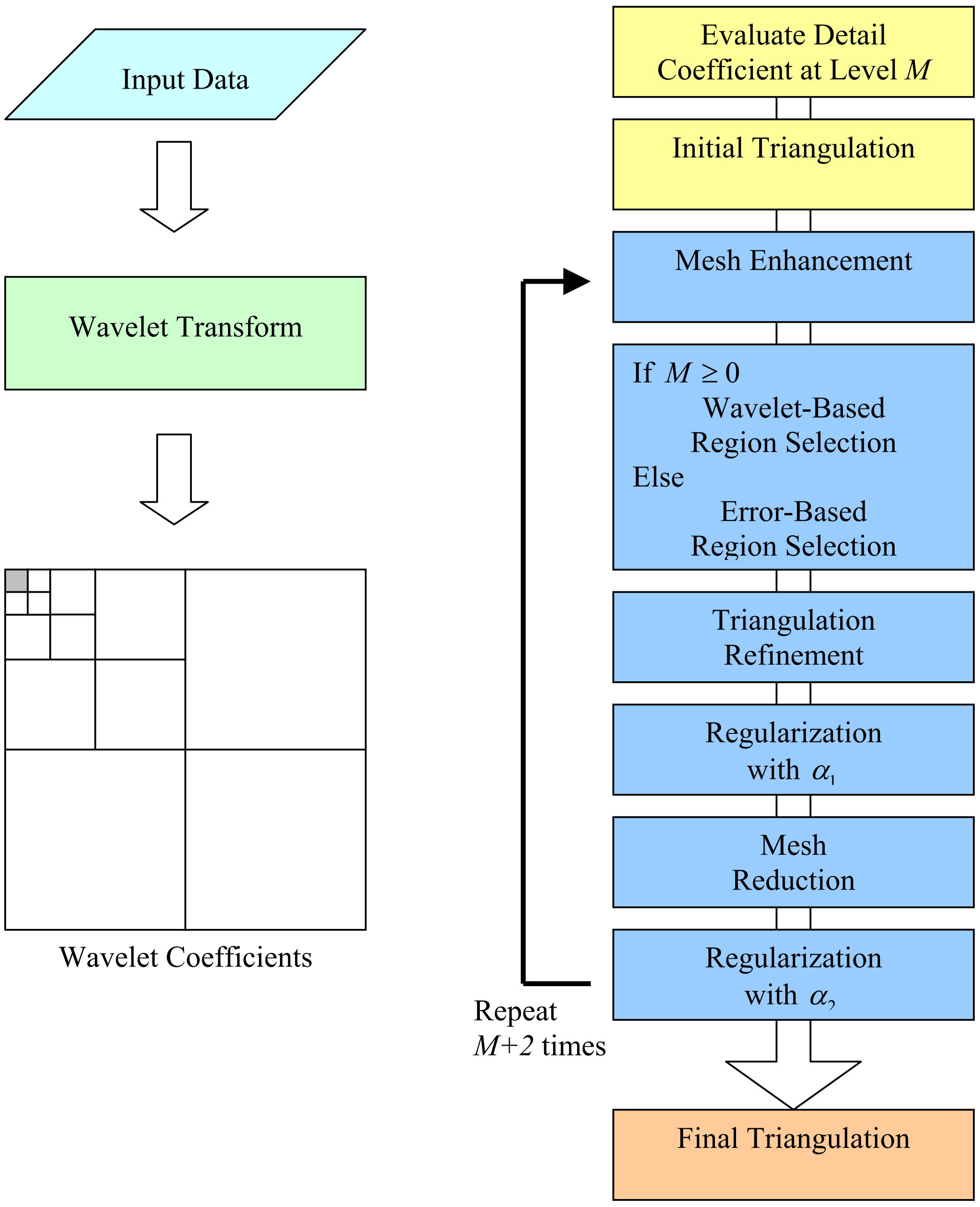
Input Data
Wavelet Transform
Wavelet Coefficients
Evaluate Detail Coefficient at Level $M$
Initial Triangulation
Mesh Enhancement
If $M \geq 0$ Wavelet-Based Region Selection Else Error-Based Region Selection
Triangulation Refinement
Regularization with $\alpha_1$
Mesh Reduction
Regularization with $\alpha_2$
Repeat $M$+2 times
Final Triangulation

Figure 5.2 – Flow diagram of the new algorithm.

### 5.2.1 Candidate Selection

To enhance the image quality, there should be some criteria to select the triangles for refinement. These triangles are called candidates. Typically, it is appropriate to subdivide a triangle containing large error into smaller triangles to produce a better approximation. These triangles can be analyzed by the total error energy.

$$\textit{Total Square Error} = \sum_x \sum_y \left[ f(x,y) - \hat{f}(x,y) \right]^2 \tag{5.1}$$

where $x$ and $y$ are locations within triangle of interest. Normally, when this error is higher than some threshold, this triangle is subdivided into some smaller triangles. The method for assigning the neighborhood status will be discussed in the next section.

In some applications, where large error must be avoided, it will be more appropriate to use maximum error as the criteria to subdivide the triangle. This error can be defined as

$$\textit{Maximum Absolute Error} = \max\left[ \left| f(x,y) - \hat{f}(x,y) \right| \right] \tag{5.2}$$

Alternatively, the other factor that can be use to identify unfit triangles is wavelet coefficients. Since wavelet coefficients can be used to detect the change of intensity, large wavelet coefficients can be used to identify triangles that need refinement. Since the wavelet transform always yields three types of wavelet coefficients (horizontal, vertical and diagonal), it is necessary to consider all of these. The wavelet scheme for selecting triangles is, therefore, based on the square sum of detail coefficients, which can be expressed as

$$e_{M,i,j} = \left| d^1_{M,i,j} \right|^2 + \left| d^2_{M,i,j} \right|^2 + \left| d^3_{M,i,j} \right|^2 \tag{5.3}$$

In the same way, if this decision criterion is greater than a threshold value, the corresponding triangle contains some detail information that cannot be neglected and, therefore, must be subdivided into smaller sub triangles.

There are two main schemes to this assignment: vertex-based and face-based assignment. For the vertex-based neighborhood, the magnitude of the sum of wavelet coefficient or error in all triangles surrounding the vertex is compared with a wavelet or error energy threshold. If it exceeds the threshold, the 0-neighborhood label is assigned to all the triangles whose corner lies on that vertex location. For the face-based neighborhood, the 0-neighborhood are assigned to those triangles whose sum of wavelet coefficients or error in the triangle is greater than the threshold. This means that the triangle contains non-negligible detail information could not be reasonably represented by a single triangle. In this thesis, face-based assignment is employed. There is no difference for 1-neighborhood and 2-neighborhood assignment for both schemes.

### 5.2.2 Neighborhood Assignment

After candidate selection is performed, it is necessary to assign its neighborhood status to neighboring triangles so that, when pairing among the candidates is not possible, these neighbors can replace the candidates. Furthermore these neighbor triangles can also be used as the limitation area for edge swapping. This method is called neighborhood assignment. Triangles can be classified into four types of neighborhood: 0-neighborhood (candidate triangle), 1-neighborhood, 2-neighborhood or 3-neighborhood (or non-neighborhood).

Assigning 0-neighborhood status to triangles has been discussed in Section 5.2.1. After candidate selection, 1-neighborhood status is assigned to triangles whose edges are common with those 0-neighborhood triangles. After 1-neighborhood assignment, 2-neighborhood status is assigned to all those triangles that have only one common vertex with 0-neighborhood triangles. The rest of the triangles are assigned as non-neighborhood. This means that they will not be used in refinement process. Figure 5.3 shows the difference between the two schemes. The neighborhood assignment for vertex-based scheme can defined by:

$$N_0(v) = \{T_i \mid v \in T_i\} \tag{5.4}$$
$$N_1(v) = \{T_i \mid T_i \cap T_j = E_j, T_j \in N_0(v) \ \ and \ \ T_i \notin N_0(v) \tag{5.5}$$
$$N_2(v) = \{T_i \cap T_j = v_j, T_j \in N_0(v)\} \tag{5.6}$$

while the neighborhood assignment for face-based scheme can defined by:

$$N_0(T) = \{T\} \tag{5.7}$$
$$N_1(T) = \{T_i \mid T_i \cap T = E_i\} \tag{5.8}$$
$$N_2(T) = \{T_i \mid T_i \cap T = V_k\} \tag{5.9}$$

The face-based approach has proved to be more popular than the vertex-based approach because it tends to be more efficient and accurate. This is because the vertex-based method usually produces more redundancy by selecting more 0-neighborhood triangles. Although it can produce a better approximation than the face-based scheme, it also produces many unnecessary vertices and triangles. This can increase processing time to the algorithm.

To consider the role of each neighborhood type, let the connections of 0-neighborhood triangles be active regions and connection of 1-neighborhood triangles be dependent regions. Generally, when the triangles are refined, it is necessary to find the point where it will use to subdivide. There are many decision criterions to find this point. For fast calculation, a triangle's centroid or barycenter can be used as split point. However, this will not guarantee a good approximation. Other choices can be the points where the largest error or wavelet coefficient are located. However, these points can lie on the edge of the triangle instead of inside the triangle itself. In this situation, the triangle is matched with another triangle in the active region before subdivision. Figure 5.4 shows the two triangle subdivisions.

In the refinement process, the criterion for selecting triangle pairs gives priority to the triangles in the active region over triangles in the dependent region. If the triangle is connected to no 0-neighborhood triangle or 0-neighborhood triangles that have already been refined, it will choose a triangle in the dependent region. The role of 2-neighborhood triangles is to define swapping area for the edge swap operation used to regularize the mesh. These 2-neighborhood triangles help the operation to converge faster. This area also includes active and dependent regions.

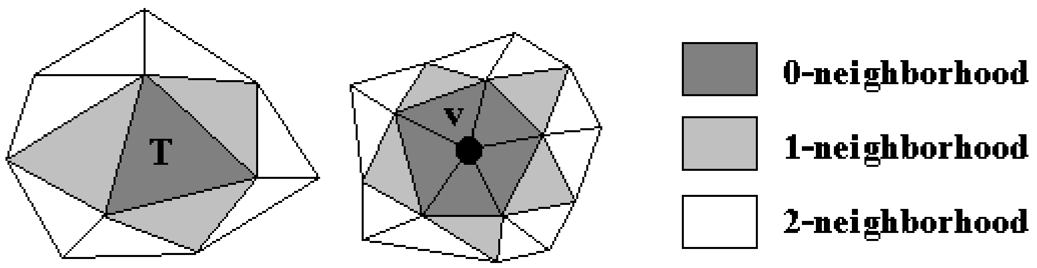

Figure 5.3 - Neighborhood assignment. (a) Face-based neighborhood. (b) Vertex-based neighborhood.

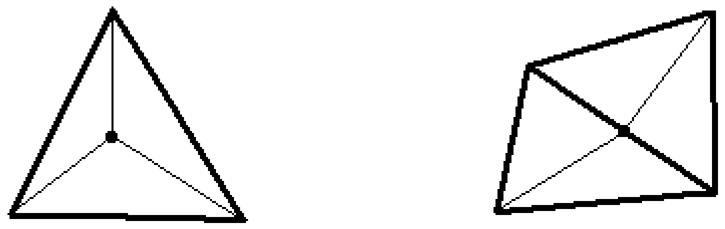

Figure 5.4 – Triangle subdivision. (a) Individual subdivision. (b) Pair subdivision.

## 5.3 Mesh Simplification

This section will give a detailed description about mesh simplification operators, which try to reduce the mesh complexity without considerably reducing the quality of the image. Since many unnecessary vertices or triangles can redundantly be added to the mesh after a refinement step, it is possible to remove these vertices without substantially degrading an image's quality. Two common methods are vertex removal and short edge collapse operations. However, to avoid violating triangular mesh requirement, there are some conditions that must be satisfied before the operations can be performed. Vertex removal operator will be discussed in Section 5.3.1 while the short edge collapse operator is presented in Section 5.3.2.

### 5.3.1 Vertex Removal Operation

Since the intensity data can be considered as representing height field data, each triangle plane that approximates the intensity value can be imagined as three-dimensional plane by projecting the intensity values to the height field value in *z*-axis. Normal vectors of these triangle planes are very useful to find the flatness of the connected triangles. The vertex removal operator can, therefore, be used to increase the compression ratio, without significantly degrading the quality of the image, by deleting the vertices whose surrounding triangles have approximately the same direction of normal vectors. One vertex and one face are reduced every time this local operator is performed.

First the operator starts by calculating the normal vector of each face and their average surface normal. Next, Gauss mapping is performed to find the deviation angle. This solution maps the surface normal vectors of all surrounding faces onto a unit sphere to check for the deviation in surface normal directions. This average normal vector and angle deviation can be calculated by

$$n_0 = \frac{1}{\left|N_0(V)\right|}\sum_{i=1}^{k} n_i \tag{5.10}$$

$$\theta_d = \max_{1\le i\le k}\left(\cos^{-1}\left(n_i \cdot n_0\right)\right) \tag{5.11}$$

where $k$ is the number of triangles surrounding the vertex $V$ . $n_0$ is the average normal vector of triangles under consideration while $\theta_d$ is the maximum deviation in angle of the vectors.

Figure 5.5 shows the concept of Gauss mapping. If the maximum angle deviation is less than a threshold value, this indicates that the *n*-connected planes are smooth and, thus, the vertex can be removed.

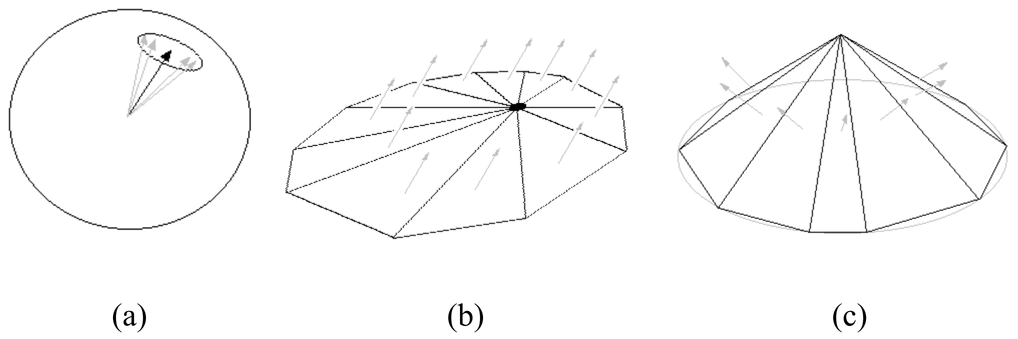

Figure 5.5 – Gauss mapping. (a) Gauss mapping maps normal surface vector to a unit sphere. (b) An example of smooth region that have approximately the same direction of normal vectors. This will satisfy vertex removal criteria (c) An example of region where normal vectors point to different direction. This will not satisfy vertex removal condition.

However, this criterion can be biased in some cases. For example, if the average normal vector is biased to a group of normal vectors, it could produce an undesired outcome, deleting a vertex from a non-smooth region. This is shown in Figure 5.6 (b). To solve this problem, an autocorrelation matrix is introduced. Since its eigenvalues represent the total deviation, these values can be used as the vertex removal criteria. This is shown in Figure 5.6(c) where the autocorrelation formula can be expressed as:

$$R(v) = \frac{1}{|N_0(v)|}\sum_{i=1}^{k} n_i n_i^T \tag{5.12}$$

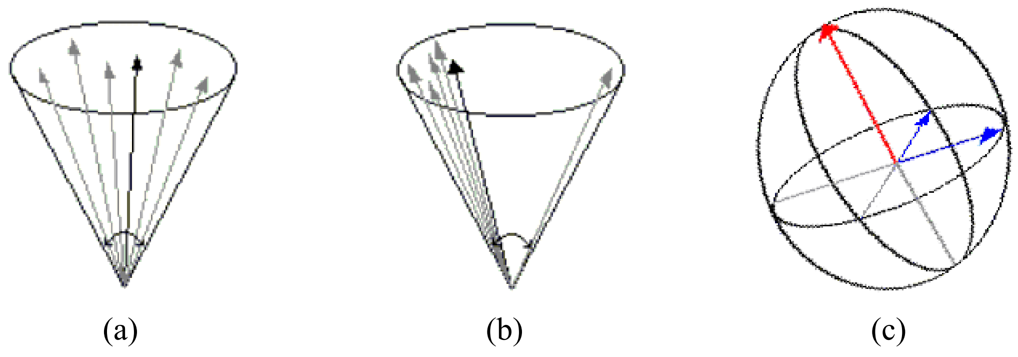

Figure 5.6 – Biased Gauss mapping and autocorrelation representation. (a) Example of good average surface normal vector $n_0$ (black arrow). (b) Example of biased average surface normal vector $N_0$. (c) Autocorrelation representation of normal vectors.

The results from this autocorrelation matrix are three eigenvalues that represent the approximation of the overall normal vectors. The largest value (red arrow) is equivalent to the average normal vector $n_0$ while the two smallest eigenvalues indicate the deviation in two perpendicular planes. Dr. Lee has found that these values perform better because it does not depend on the average normal vector that can be biased. The criteria for the vertex removal operation is

$$\lambda_1 \le \lambda_2 < \delta_\lambda \tag{5.13}$$

where $\delta_\lambda$ is the threshold that is used to control the maximum deviation. If this value is too low, very little or no redundant vertex will be deleted. However if this value is too large, some important vertex will be deleted, causing the final quality to degrade dramatically.

As the mesh becomes more complex, it requires more time to perform this local operation. One way to reduce processing time is to perform pre-checking. After some observation, it is observed that more than half of the vertices in the triangular mesh lie on the location where large wavelet coefficients occur. The pre-checking condition can be expressed as

$$\left|d^1_{m,i,j}\right|^2 + \left|d^2_{m,i,j}\right|^2 + \left|d^3_{m,i,j}\right|^2 < \delta_w \tag{5.14}$$

where $\delta_w$ is the threshold to control pre-checking strictness. Setting this value too low can cause redundant vertices to remain unresolved, while setting this value too high would cause the algorithm to gain no advantage of this pre-checking at all.

### 5.3.2 Edge Collapse Operation

The other common local operation that can be used to reduce the number of triangles in the triangular mesh is edge collapsing. Typically this method removes two triangles and one vertex from the triangulation by merging two adjacent vertices. Figure 5.7 shows some examples of this.

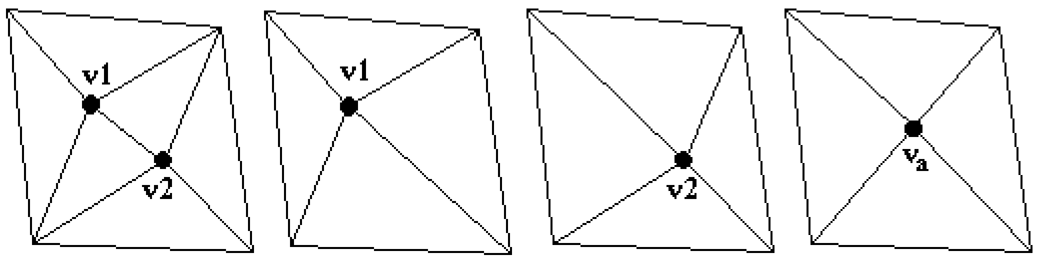

Figure 5.7 – Edge collapse example. (a) Original triangulation. (b) Edge collapse by removing vertex $V_2$. (c) Edge collapse by removing vertex $V_1$. (d) Edge collapse by replacing both vertices by $\frac{(V_1+V_2)}{2}$.

There are many criteria that can be used to select the vertices and triangles. One of the most popular ones is to use the distance between two vertices. Figure 5.8 shows an example of an edge collapse operator. If the edge is shorter than some edge collapse threshold, the edge is removed. Although it is consistent to remove the edge, this distance criterion does not guarantee a good approximation.

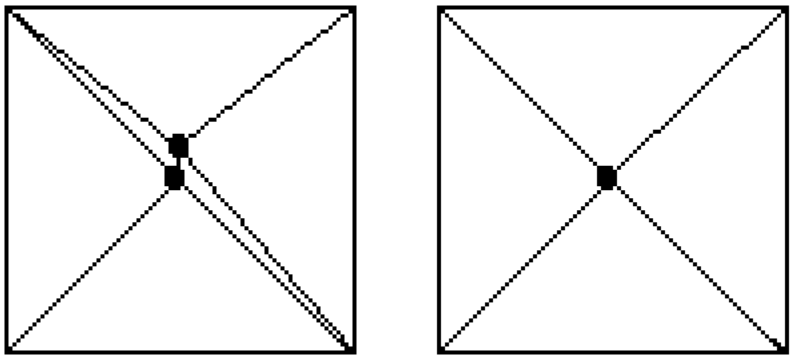

Figure 5.8 – Top view of an edge collapse operation.

For example, Figure 5.9 illustrates a condition where edge collapse will not produce a good approximate result. Although vertices $V_1$ and $V_2$ lie close together when looking from the top viewpoint, they can be located far away in the horizontal viewpoint (front view and side view). Figure 5.10 shows the result of edge collapse by removing vertex $V_1$ or vertex $V_2$. Both reconstructed cases do not resemble the original well. Therefore it is necessary to consider the vertical distance as well as the horizontal distance. Figure 5.11 shows an example of a desired case where the two vertices lie close to each other in both horizontal and vertical distance.

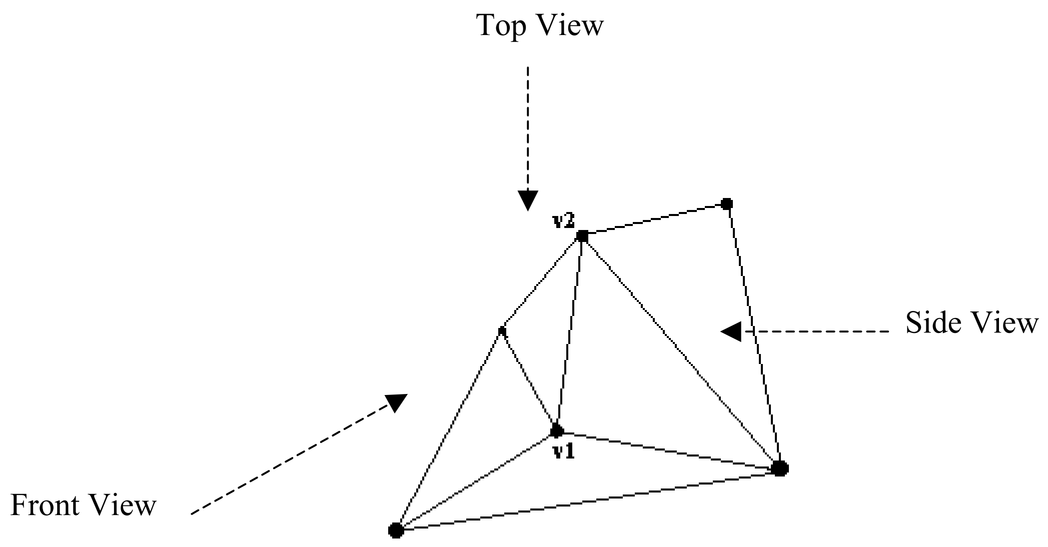

Figure 5.9 – Undesirable condition for edge collapse.

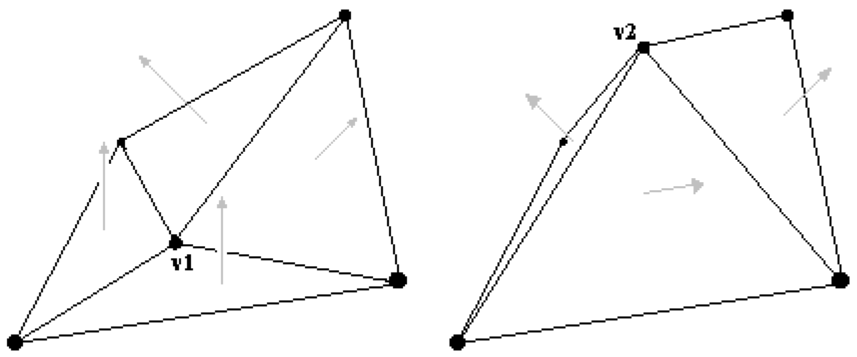

Figure 5.10 – Undesirable result of edge collapse.

The edge collapse conditions are

$$\left| z_{v_1} - z_{v_2} \right| < \delta_v \tag{5.15}$$

and

$$\sqrt{\left(x_{v_1} - x_{v_2}\right)^2 + \left(y_{v_1} - y_{v_2}\right)^2} < \delta_H \tag{5.16}$$

These two distances can be combined into one parameter as total distance to speed up time for computing as

$$\sqrt{\left(x_{v_1} - x_{v_2}\right)^2 + \left(y_{v_1} - y_{v_2}\right)^2 + \left(z_{v_1} - z_{v_2}\right)^2} < \delta_d \tag{5.17}$$

This means that any vertex $V_1$ that lies adjacent to vertex $V_2$ within a sphere distance of $\delta_d$ can be removed while maintaining an overall representation of the mesh. This operator is also efficient in time because it involves only calculation and comparing of the distance of two vertices.

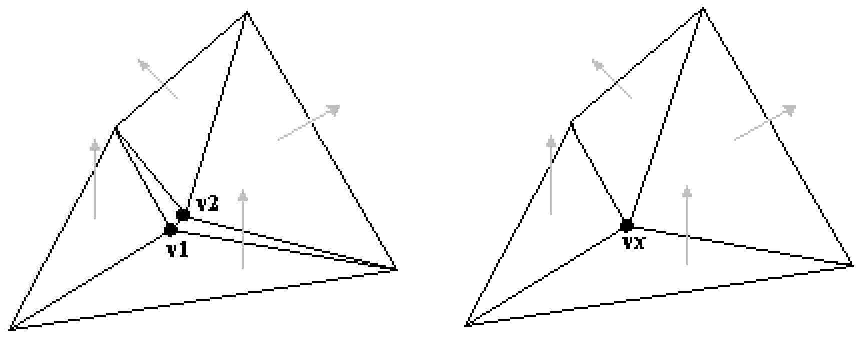

Figure 5.11 – Short edge collapse.

However there are some cautions that should be made before removing any edge from the triangular mesh. Figure 5.12 demonstrates how removing an edge can cause violation in triangulation. When vertices $V_1$ and $V_2$ are merged to either position $V_1$ or $V_2$, triangles $\Delta 132$ and triangle $\Delta 241$ are removed. However, four more triangles $\Delta 523$, $\Delta 153$, $\Delta 264$ and $\Delta 461$, are forced to have zero area and this causes violation in triangulation. In this case, it is also impossible to collapse edge $\overline{13}$, $\overline{32}$, $\overline{24}$ and $\overline{41}$.

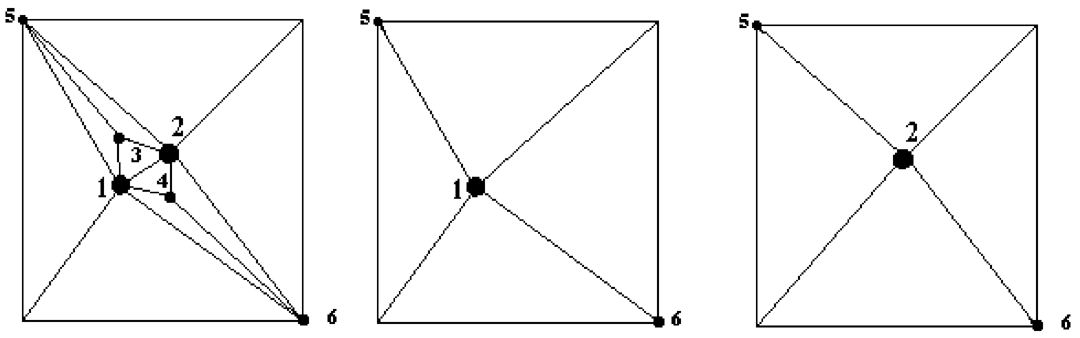

Figure 5.12 – Triangulation violation caused by edge collapse.

There are many solutions to this problem. Besides checking for the feasibility of removing the edge, an easy solution is to check whether the edge lies in the convex polygon of its surround vertices. This condition guarantees that no violation will occur. Figure 5.13 shows the possibilities of removing an edge from the polygon. One of the ways to check for convexity of the polygon is to find cross product of $\overrightarrow{V_0V_1}$ and $\overrightarrow{V_0V_2}$, where $V_0$ is the individual surrounding vertex, and $V_1$ and $V_2$ are the adjacent vertices of $V_0$. Since we are only interested in the *z*-axis, the criteria can be reduced to

$$(x_{V_1} - x_{V_0})(y_{V_2} - y_{V_2}) - (y_{V_1} - y_{V_0})(x_{V_2} - x_{V_0}) > 0 \qquad (5.18)$$

for each neighboring vertex $V_0$ that is adjacent to the deleted vertex. Figure 5.14 shows an example where a concave polygon can produce violation in the triangulation. Therefore, checking for polygon convexity before removing a vertex or an edge can ensure validation in triangulation.

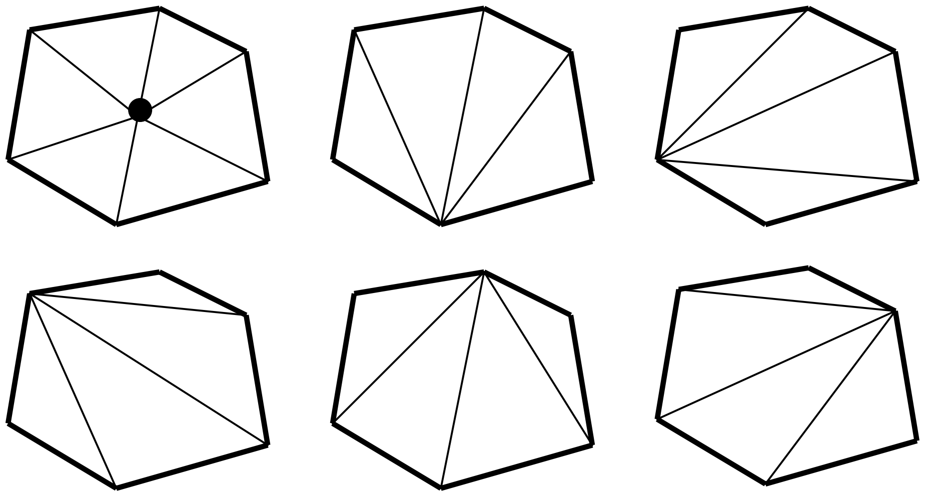

Figure 5.13 – Effect of edge collapse on convex polygon. It is possible to collapse any edge from the convex polygon without causing violation in the triangular mesh.

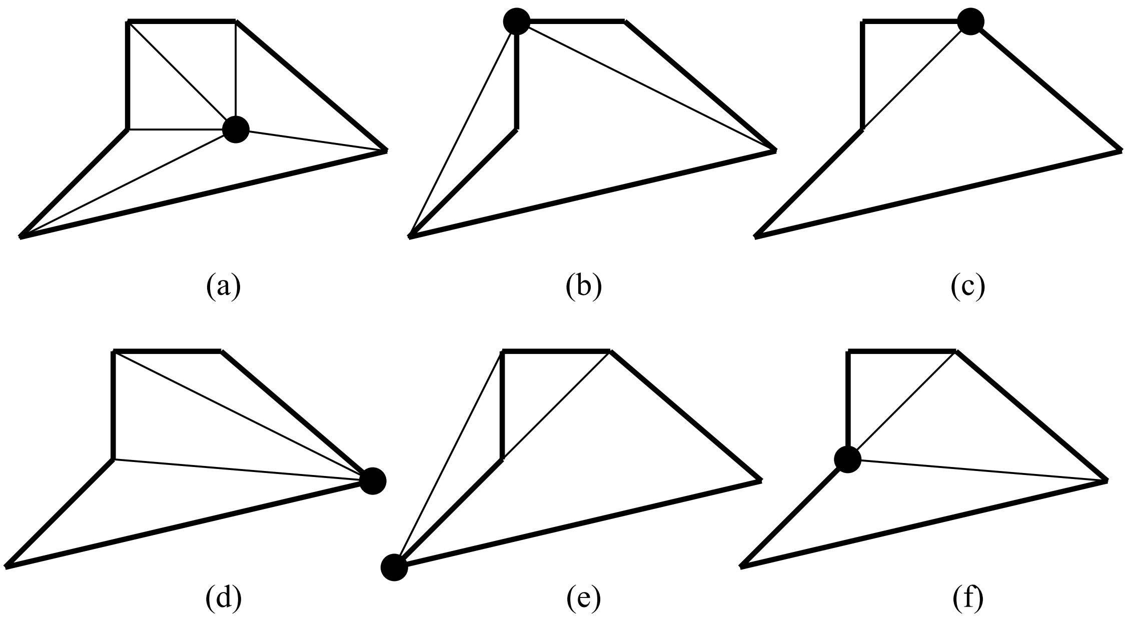

Figure 5.14 – Effect of edge collapse on a concave polygon. Removing a vertex from a concave polygon can cause a violation in triangulation if an inappropriate edge is chosen. (a) Original polygon with vertex of interest. For this polygon, only case (d) and (f) do not violate the triangulation rule.

## 5.4 Image Enhancement

After initial triangulation, the reconstructed image from the triangulation usually does not represent the original image well. This section will give the description of image enhancement operators that try to increase the quality of the reconstructed image by adding as few triangles as possible to the triangular meshes.

### 5.4.1 Edge Swap Operation

The edge swap operator is very useful in many triangular mesh applications. First, it can be used either to regularize the mesh or improve the image quality without inserting new vertices. One of the most popular regularization techniques is known as Delaunay triangulation. This will be discussed in Section 3.2. This scheme tries to maximize the minimum angle of all the triangles. Although the use of this 'fat' triangle is not desirable in some cases, improved Triangulated Irregular Network (TIN) [Kim, Park, Jung, Cho 99] proves that Delaunay mesh can be compressed at compression rate of 0.30 bits/vertex [Kim, Park, Jung, Cho 99]. While a smaller number of thin, sliver triangles can give a better approximation to the image, this requires up to 12 bits/vertex for non-Delaunay triangulation.

The decision for Delaunay triangulation is usually based on the angles of the triangles. The input is two connected triangles. The edge swap operation will try to regularize the mesh by maximizing the minimum angles found in these two triangles. Figure 5.15 shows an example of a Delaunay mesh decision. This can be formulated as

$$\min \measuredangle\left(\tilde{T}_i,\tilde{T}_j\right) > \min \measuredangle\left(T_i,T_j\right) \tag{5.19}$$

where $T_i$ and $T_j$ are the input triangles and $\tilde{T}_i$ and $\tilde{T}_j$ are the future swapped triangles.

The other common criterion for edge swap is the total error within the two triangles. To give a better approximation of the original image, the current and predicted error bounded within the triangle are calculated for comparison. This decision can be expressed as

$$E\left(\tilde{T}_i\right)+E\left(\tilde{T}_j\right) < E\left(T_i\right)+E\left(T_j\right) \tag{5.20}$$

However, this data-dependent schemes can be time consuming because it requires an error calculation for every image pixel within the triangles while Delaunay calculation needs only to compare the minimum angles. This scheme is shown in Figure 5.16.

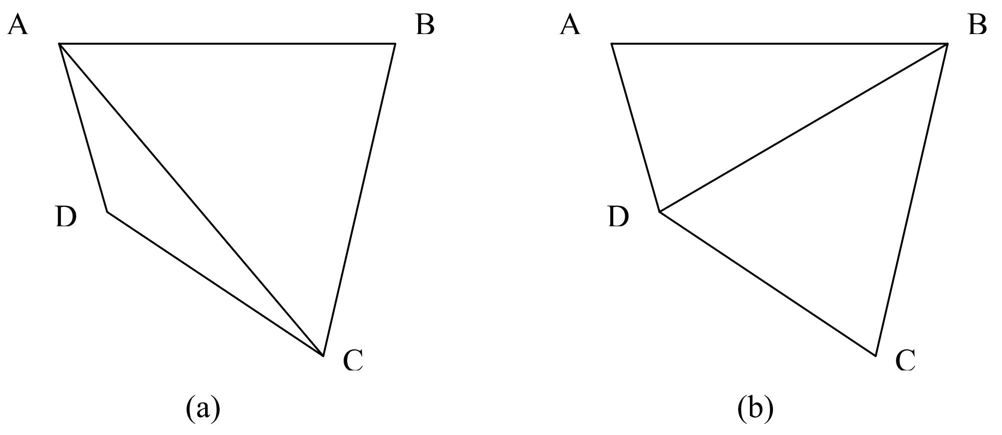

Figure 5.15 – Delaunay triangulation decision (a) Two connected triangles that have not been regularized. (b) Delaunay scheme regularizes the mesh by swapping the edge to maximize the minimum angle.

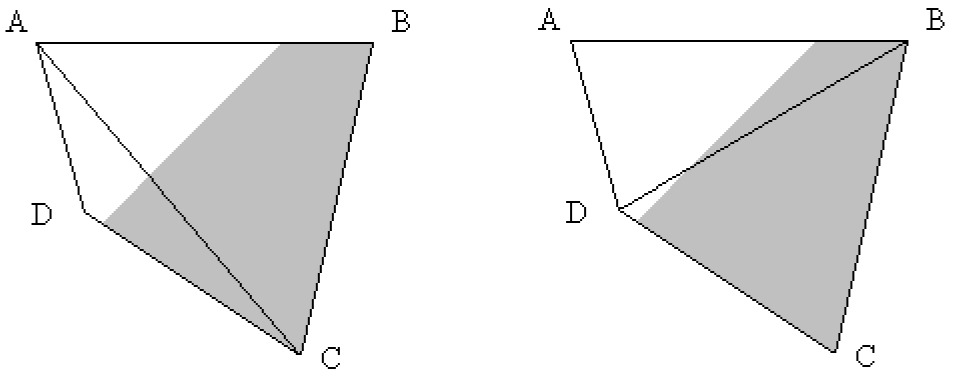

Figure 5.16 – Error-based edge swap (data-dependent) (a) Pair of triangles that do not represent the original data well. (b) New pair of triangles that produce a better approximation.

However, while data-dependent triangle is preferred for optimal result, sometimes it is undesirable to have long and sliver triangles. Therefore, in his dissertation, Dr. Lee has proposed a formula to compromise the importance of both the Delaunay and data-dependent schemes. This can be defined as

$$\xi\left(T_i,T_j\right)=\alpha\,\Re_A+\left(1-\alpha\right)\Re_E \qquad (5.21)$$

where

$$\Re_A=\frac{\min\measuredangle\left(t_i,t_j\right)}{\min\measuredangle\left(T_i,T_j\right)} \qquad \Re_E=\frac{E\left(T_i\right)+E\left(T_j\right)+1}{E\left(t_i\right)+E\left(t_j\right)+1} \qquad (5.22)$$

$\Re_A$ and $\Re_E$ are the ratios of minimum angles and ratio of total errors of the two set of triangles respectively. This mesh regularity factor is controlled by parameter $\alpha$. If $\alpha$ is 1, the triangulation will base on Delaunay decision and it will be data dependent scheme if $\alpha$ is 0. The range of $0 < \alpha < 1$ will give the combination of both the Delaunay and the data-dependent triangulation, depending on the weight of the $\alpha$. This result will be presented in Section 6.3.1.

### 5.4.2 Wavelet Coefficient Attraction Operation

There are many operators that can improve the quality of the reconstructed image. However, most of them involve insertion of new vertices or edge swapping. These operators usually do not involve in relocation of vertex position. Therefore the improvement will generally be limited by the fixed location of the vertices. Figure 5.17 shows how the edge swap operator has no effect in improving the result. Notice that the region in rectangle $\square bcfe$ and $\square dehg$ do not achieve any improvement after edge swap operator is performed.

One of the most common vertex-moving operations is to find a new vertex location with minimum error. However this brute force approach will be too time-consuming for many real-time applications. One of the new approaches to move the vertices efficiently is to use wavelet coefficients. Since wavelet coefficients indicate the locations of edges where vertices should be found, this coefficient can be used to attract the vertex closer to these edge locations. The vertical wavelet coefficient will attract the vertices horizontally while the horizontal wavelet coefficient will try to vertically pull the vertices towards itself. This concept is shown in Figure 5.18.

However there are many conditions that must be met before moving each vertex. First the vertex must be located in the area where wavelet coefficient is very low. It requires checking wavelet coefficients in the surrounding area with magnetic threshold, $\mu_P$, because it is unnecessary to move the vertex, which is already located near or on the large wavelet coefficient. The size of this area template, $\mu_T$, will depend on the size and resolution image as well as the weighing template, which will be discussed next. Figure 5.19 (a) shows a 3x3 template used for comparing with magnetic threshold, $\mu_P$.

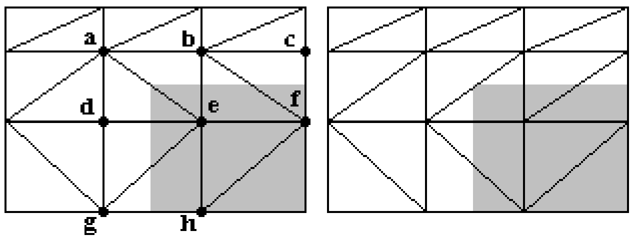

Figure 5.17 – Limitation of edge swap (a) Original mesh. (b) New mesh after edge swap operation does not produce much better result.

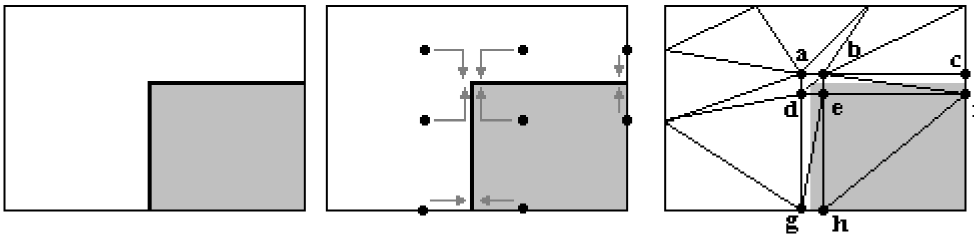

Figure 5.18 – Wavelet coefficient attraction (a) Edge (black line) detected by Wavelet Transform. (b) Vertices are attracted to the nearby large magnitude wavelet coefficients. (c) Better approximation by moving vertices.

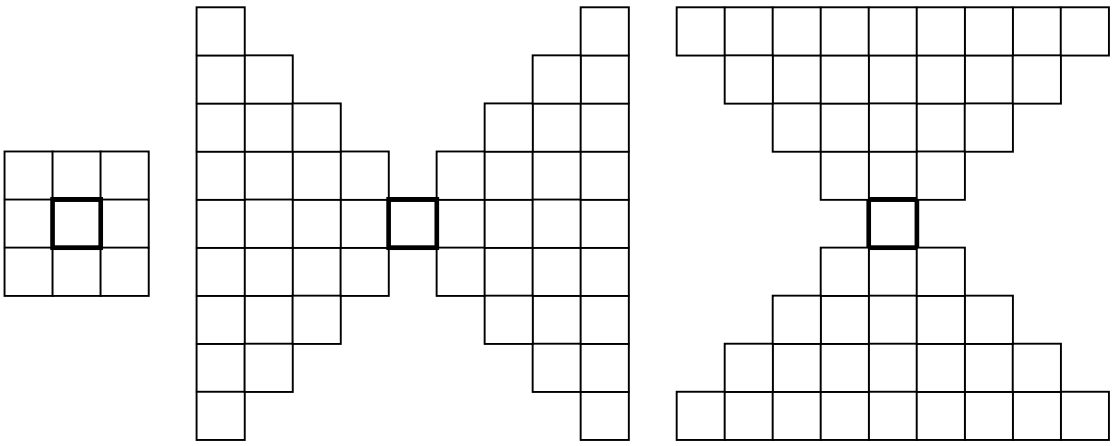

Figure 5.19 - Templates for wavelet coefficient attraction (a) Area template (b) horizontal attraction template (c) Vertical attraction template. The black-bordered box indicates the location of the vertex of interest.

After checking that the surrounding pixels have total detail energy lower than some threshold value, the vertex uses the wavelet horizontal and vertical attraction templates shown in Figure 5.19 (b) and (c) respectively to look for possible large wavelet coefficients nearby. While using the horizontal attraction template with length, $\mu_L$, the information is retrieved from the vertical wavelet coefficient and vice versa. During the horizontal check, it will try to compare the magnitude of the wavelet coefficients on the left and right of the template. The vertex will move to whatever direction that has the heavier (or larger) wavelet coefficient with the difference larger than magnetic additional weight, $\mu_W$ . This also applies to the vertical template where top and bottom balance is used instead of left and right comparison.

To prevent the creation of invalid triangular meshes, the new vertex location must be checked for consistency before moving the vertex. First, it must not be located on another vertex location or outside of image boundary. Second, it must satisfy clockwise orientation so that all triangles will always have positive normal vectors. Figure 5.20 shows an example when moving vertex can cause violation in triangular mesh.

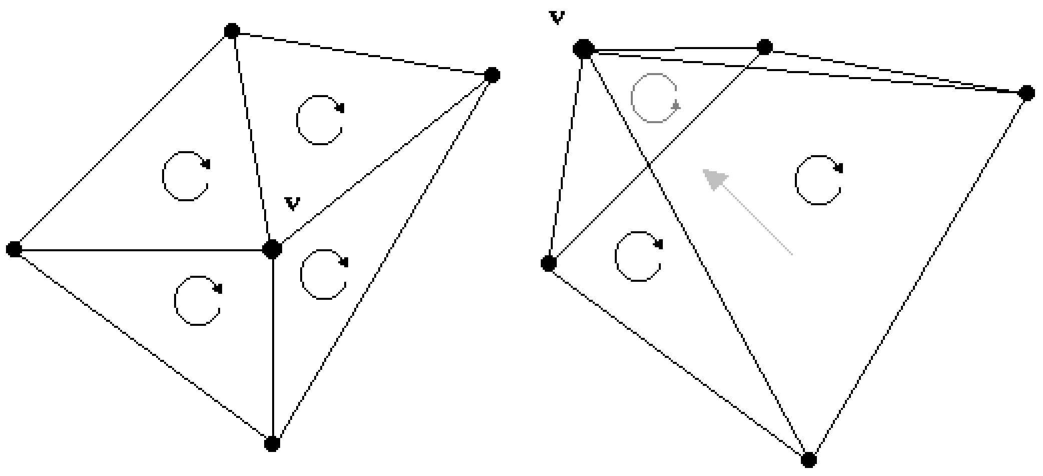

Figure 5.20 – Triangular mesh violation after vertex translation. (a) Original triangulation with strictly clockwise orientation. (b) Moving vertex *V* causing their vertex to be ordered in counter-clockwise direction.

### 5.4.3 Shape-Preserving Split Operation

There are many times when edge swapping and vertex translation is limited to improving the quality of the reconstructed image because of the limited number of current vertices. There is a need to perform triangle subdivision to improve the reconstructed output. For example when an image is represented by only ten triangles, it may be impossible to improve the output quality without inserting a new vertex to the triangulation to obtain a better approximation. Generally, there are two types of triangle subdivision: Subdivision within the triangle itself and subdivision among the triangles in active or dependent region. Splitting among triangles can be performed on two or more triangles. However, normally, it is applied to two triangles because of simplicity and its better approximation. Figure 5.21 shows an example of each type of subdivision.

One of the easiest ways to choose the location of the new location is to use the triangle centroid. Although this method is fast in computing and gives good regularity in the new triangles created, it does not guarantee the result and, therefore, can add unnecessary vertices and triangles to the triangulation. The other two possible solutions are to find the location of maximum error and absolute wavelet coefficient in the triangle. However, although the position is known, this does not give any suggestion as to how the triangle should be cut. Dr. Lee has stated in his dissertation that it is possible to use the wavelet detailed coefficient to find the best cutting direction instead of the vertex location. First the direction is calculated before find an appropriate paired triangle. Next the best cutting direction for the paired triangle is calculated. In the best case, the inserted point is the intersection location of these two cutting lines. If the lines do not intersect, the average of the locations of the cutting lines and the shared edge is used as the insertion point. Figure 5.22 shows the concept of this directional subdivision. Let $c_t$ be the barycenter point of the triangle and the gradient vector, which determines the discontinuities direction in the two-dimensional image. This can be expressed as

$$g = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}^T = \frac{1}{2^{-m}} \begin{bmatrix} -d_m^1 & d_m^2 \end{bmatrix}^T \tag{5.23}$$

Implicitly, the cutting direction, which is perpendicular to this direction, can be defined as

$$g^{\perp} = \begin{bmatrix} -\frac{\partial f}{\partial y} & \frac{\partial f}{\partial x} \end{bmatrix}^T \tag{5.24}$$

With this slope and passing point, the first line equation is obtained. The second line equation is easily computed from the edge information. With these two known line equations, it is possible to compute the intersection points. The splitting point is the average of the two intersecting points as described. If finding an appropriate pairing or splitting point is unsuccessful, splitting within the triangle is used to solve the problem.

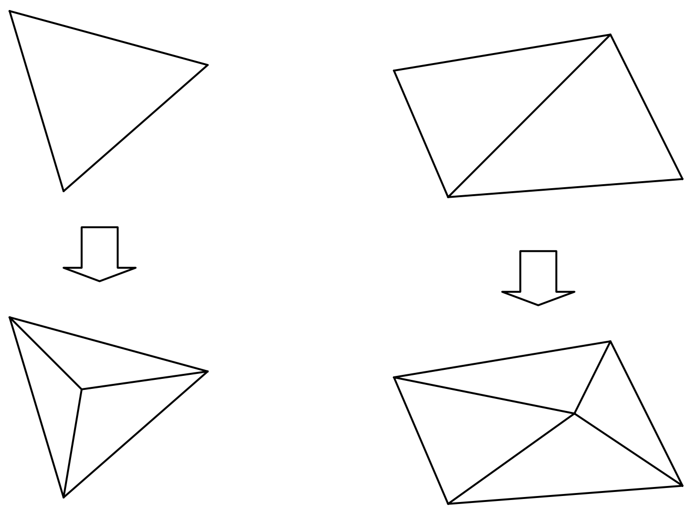

Figure 5.21 – Triangle split. (a) Split within triangle. (b) Split among triangles.

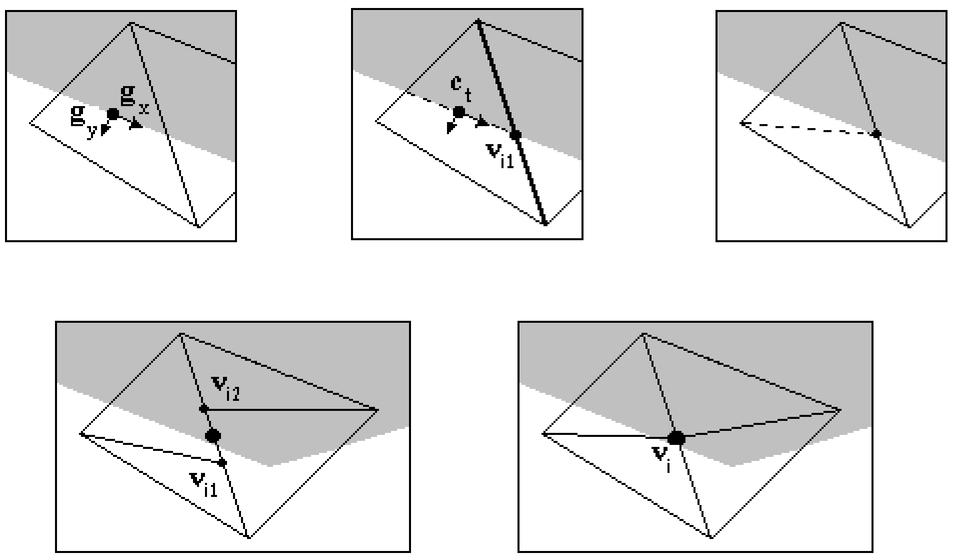

Figure 5.22 – Directional split based on wavelet coefficient. (a) Directional is calculated from vertical and horizontal detailed coefficient. (b) With the help of barycentric point, $c_t$, intersection point, $V_{i_1}$, and pair triangle can be found. (c) The vertex that is not on the shared edge is used as the final cutting line direction. (d) Basing on the pair triangle, the cutting line as well as the intersection point on the shared edge is calculated. (e) The average of two intersection points is used as the insertion point.