Chapter 2

Previous Related Work

This chapter presents a bibliographical review of previous work related with the topics dealt with in this dissertation, including: *(1)* mesh representations, *(2)* image sampling techniques, *(3)* data triangulation techniques, *(4)* generation of adaptive triangular meshes from images, *(5)* generation of images from triangular meshes, *(6)* triangular mesh processing and, due to its close relation to this dissertation, *(7)* a short review of previous techniques for the manipulation of compressed images.

2.1 Mesh Representations

A mesh is a set of points (also known as *nodes* or *vertices*) and segments that link those points, such that two segments can only intersect at a shared endpoint. Meshes are usually defined in both the 2D and 3D space.

There are two basic models of meshes: *structured meshes* and *unstructured meshes*. The main difference between them is determined by the data structure utilized to describe the mesh itself (Lattuada, 1998). Thus, the number of adjacent points for each point of a structured mesh is the same, except for the boundaries, Figure 2.1(*left*). Two typical examples of structured meshes are quadrilateral and hexahedral meshes. On the other hand, an

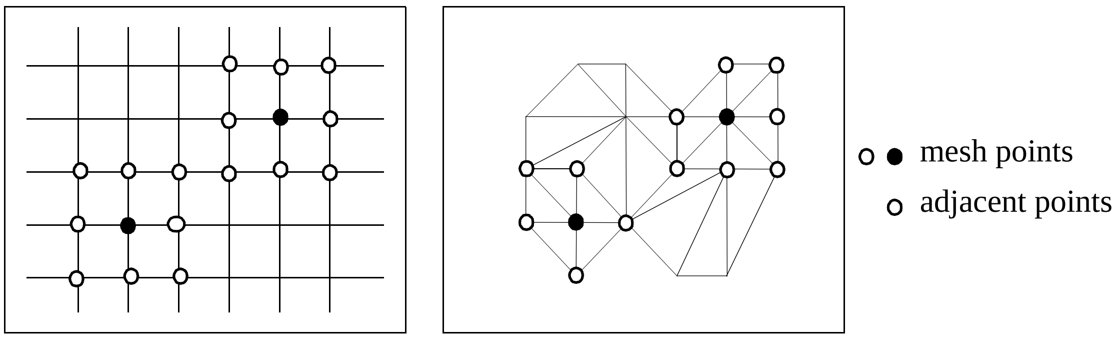

Figure 2.1. Structured and unstructured Meshes. (*left*) Quadrilateral, structured mesh. (*right*) Triangular, unstructured mesh.

unstructured mesh allows any number of adjacent points associated with a single point of the mesh. Figure 2.1(*right*) illustrates an example of unstructured mesh. Triangular and tetrahedral meshes are the most utilized unstructured meshes, although quadrilateral and hexahedral meshes can also be unstructured (Owen, 1998). Borouchaki and Frey (1996) propose a triangular to quadrilateral mesh conversion scheme. Likewise, there are methods that allow the combination of small quadrilateral meshes into a single triangular mesh (Davoine, Svensson, Chassery, 1995; Davoine, Robert, Chassery, 1997).

The main advantage of structured meshes over unstructured meshes is that they require less computer memory, as well as that they are simpler and more convenient to use for finite difference methods. Nonetheless, the big disadvantage of a structured mesh is its lack of flexibility in adapting to a domain with a complicated shape (Bern, Eppstein, 1995). For this reason, and given the need to model shapes with intricate domains, the common trend has been to use unstructured meshes, since they can arbitrarily adapt to complicated domains. Therefore, this survey is focused on unstructured mesh generators, concretely on 2D and 3D triangular mesh generators.

In the next section, a bibliographical review of previous techniques for image sampling is presented. Those sampled data are then used for generating triangular meshes, which can be contained in the 2D or 3D space.

2.2 Image Sampling Techniques

The first step for the generation of a 2D or 3D triangular mesh from an image is the choice (sampling) of a set of points that are to be triangulated. In this context, an image is considered to be a dense 2D array, such as a *gray level image*, a *range image* or a *digital elevation map*. Each element of the array is considered to be a *pixel*. A pixel has an implicit location in the array (e.g., a row and a column), and a property (e.g., gray level, range, elevation). Each image pixel defines a point in a 3D space in which, for example, the x and y coordinates correspond to the pixel's row and column respectively, and the z coordinate corresponds to the pixel's property. Those points are the data which are finally triangulated. Three possible techniques have been proposed for sampling an image: *(1) sampling all the original pixels*, *(2) sampling a predefined number of pixels* and *(3) sampling a non-predefined number of pixels*.

2.2.1 Sampling all the Original Pixels

The trivial technique for sampling an image consists of choosing all its elements (pixels). This procedure is generally used when a *fine triangular mesh* is sought. In this way, the resulting triangulation is a dense triangular mesh with a high resolution, which contains all the original data. Several techniques proposed in the literature generate an initial dense triangular mesh with the purpose of applying further decimation operations over it. They are overviewed in Section 2.6.1.

2.2.2 Sampling a Predefined Number of Pixels

When the original pixels contain redundant information, it is necessary to sample a subset of them in order to eliminate that redundancy. The sampled pixels must be chosen in such a way that they constitute a good approximation of the original image. Two methods can be utilized to sample a predefined number of pixels from an image: *(1) uniform sampling* and *(2) non-uniform sampling*.

2.2.2.1 *Uniform Sampling*

The simplest way of obtaining a uniform sampling from a given image consists of sampling that image at specific intervals both horizontally and vertically. Thus, this process chooses one pixel out of a predefined number of pixels along the rows and columns of the image. Davoine and Chassery (1994) and Lechat, Sanson and Labelle (1997) apply the uniform sampling process in order to obtain an initial triangular mesh which approximates the given image. Then, they work over this initial mesh to improve the approximation error of the final triangular mesh. Similar techniques are surveyed in Section 2.6.2.

2.2.2.2 *Non-Uniform Sampling*

Adaptive sampling is the most known non-uniform sampling technique. It varies the density of sampled pixels depending on the amount of information (details) present in the given image. In this way, the chosen pixels are concentrated in areas of high curvature and dispersed over low variation regions. García, Sappa and Basañez (1997a) propose a non-uniform sampling technique for the generation of adaptive quadrilateral meshes from range images. This technique selects a predefined number of pixels from a given image by first computing a curvature estimation of the whole range image. Then, they divide the range image into a user-specified number of equally-sized rectangular tiles. Taking into account the previous curvature estimation, a user-specified number of pixels is independently chosen for every tile. The final quadrilateral mesh is obtained from the pixels sampled at all the tiles.

Following this line, Huang and Zheng (1996, 1997) present two methods that allow to sample a predefined number of points ensuring a maximum approximation error between the resulting triangular mesh and the points of the original image. The algorithms divide the original image into small windows. Afterwards, the sampled points are determined for each window according to the spatial frequencies of each window by using the Short Time Fourier Transform or the Wavelet Transform respectively.

2.2.3 Sampling a Non-Predefined Number of Pixels

There are other techniques that subsample an image by taking into account the information contained in it, trying to approximate it as much as possible, but without guaranteeing the amount of pixels that will be finally chosen. García (1995a) proposes a technique that samples an original range image by applying a random selection process that tends to choose more pixels in areas of high curvature and fewer pixels in low variation regions. Curvature is used as a measure of the probability of choosing a pixel for the final triangulation. The points corresponding to the selected pixels are then triangulated in order to obtain a triangular mesh that approximates the given range image. This algorithm does not guarantee a specific number of points in the final mesh. However, the final number of points is indirectly bounded by a sampling rate parameter that determines the density of points of the final triangulation.

Another technique that selects a non-predefined number of pixels from a given gray level image is presented in (Altunbasak, 1997). Initially, the algorithm applies an image segmentation in order to obtain separate objects in the image. Then, each object boundary is approximated by means of a polygonalization algorithm. A set of points is selected after the previous process. Finally, a square region is grown around each point of the enclosing polygon of that object, until the spatial color gradient within the part of growing region that overlaps with the object reaches a predefined threshold. The pixels which define the square regions are sampled. The final set of pixels is obtained by applying the previous procedure to all boundary pixels.

Another technique that combines uniform sampling and does not guarantee a predefined number of sampled pixels was presented by Wolberg (1997). Wolberg applies an edge detector to a gray level image in order to capture all the pixels that define the contours of the image. Then, a uniform sampling is applied to the original image. The resulting set of pixels is a merging of the edge pixels and the uniformly sampled pixels. Those pixels are then utilized by an image approximation algorithm.

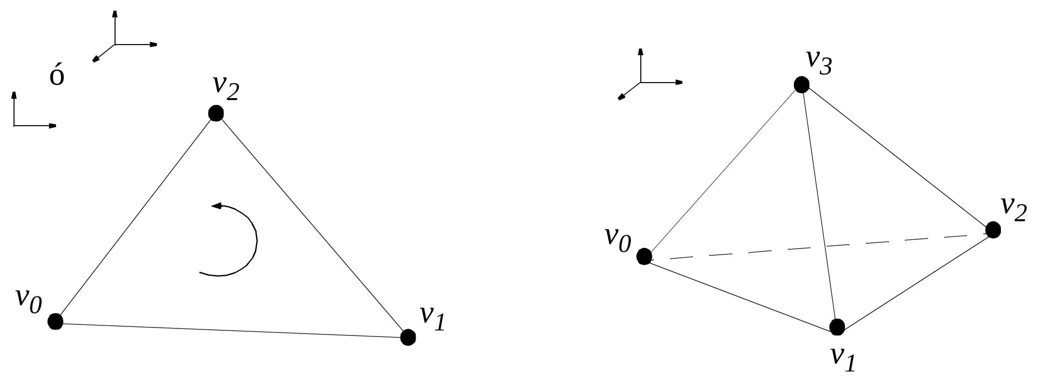

Figure 2.2. (*left*) triangle for 2D or 3D data. (*right*) tetrahedron for 3D data.

2.3 Data Triangulation Techniques

The input data for a triangulation process is a set of points which can be defined in the 2D or 3D space. Those points may also be linked by edges that have to be preserved after the triangulation. In the latter case, those edges act as *constraints* in the triangulation.

Once the input data for the triangulation have been defined, it is necessary to determine the type of triangulation. There are two possible types of triangulation. In the first type, the points of the resulting triangulation are exactly the input points, while in the second type, the resulting triangulation can contain, besides the input points, additional points called *Steiner points* (Bern, Eppstein, 1995). In both cases, the obtained triangulation is generated with or without optimization criteria.

In general, the objective of a triangulation is the partition of the initial geometrical domain, typically the regions defined by the given set of points, into a set of domains with simple geometry, such as triangles or tetrahedra. In this context, a *triangle* is a polygon with three segments (edges), which is defined by a triplet of points given, for example, in counterclockwise order, such as (v_0, v_1, v_2) in Figure 2.2(*left*). On the other hand, a *tetrahedron* is a polyhedron with four triangular faces defined by four points, such as (v_0, v_1, v_2, v_3) in Figure 2.2(*right*).

Taking into account the previous considerations, different solutions have been reported in the literature for the problem of data triangulation. For instance, Bern, Eppstein and Gilbert (1990) study several algorithms for the generation of triangular meshes starting with a given set of points. A more recent and complete survey of triangulation algorithms in both the 2D and 3D space is presented by Owen (1998). This paper also surveys software tools for generating triangular meshes.

Three different types of triangulation techniques can be defined: *(1) 2D triangulation*, *(2)* $2^1/_2$*D triangulation* and *(3) 3D triangulation*.

2.3.1 2D Triangulation

The triangulation of a set of 2D points is a partition of these points into a set of non-overlapping triangles. Depending on the shapes of the generated triangles, we can distinguish between 2D triangulation algorithms with and without optimization.

2.3.1.1 Two Dimensional Triangulation without Optimization

An algorithm that generates 2D triangular meshes without optimization does not consider any topological criteria when it generates the resulting mesh from the given data. The only considered constraint is that there are no triangles overlapped in the final triangular mesh (Kumar, 1996). Thus, the points contained in the final mesh are assumed to be the input points, without the addition of Steiner points. 2D triangulation algorithms without optimization criteria are not usually utilized since their results can lead to *degenerated triangles* (triangles with low area/perimeter ratio) and their computational cost is the same as the one of some triangulation algorithms with optimization.

2.3.1.2 Two Dimensional Triangulation with Optimization

Contrarily to the previous algorithms, the triangulation techniques with optimization consider one or more quality measures during the triangulation process. Those measures depend on the shape of the resulting triangles (O'Rourke, 1994). These algorithms may add

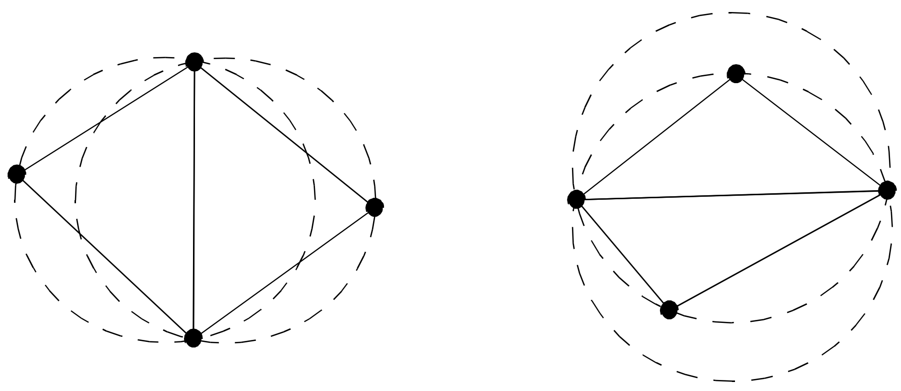

Figure 2.3. Example of the Delaunay criterion. The circumcircle of each triangle only contains the points of that triangle. (*left*) fulfils the Delaunay criterion while (*right*) does not.

Steiner points. With or without Steiner points, the typical quality measures take into account the angles, edge lengths, heights and areas of the generated triangles. The quality measure associated with the final triangulation is then taken to be the sum, maximum or minimum of the measures obtained for all the triangles (Bern, Eppstein, 1995).

Within the 2D triangulation algorithms with optimization, a popular technique is the *Delaunay triangulation*. This algorithm allows the generation of triangular meshes with or without the addition of Steiner points. The Delaunay triangulation allows the simultaneous optimization of several of the quality measures mentioned above. In this way, the Delaunay criterion states that any point must not be contained inside the circumcircle of any triangle within the mesh. In other words, any point must only lie on the circumcircles corresponding to the triangles that point belongs to. A *circumcircle* can be defined as the circle passing through the three points of a triangle. Figure 2.3 illustrates a simple example of this criterion. In practice, the Delaunay criterion is attained by maximizing the minimum interior angle and minimizing the maximum circumcircle of a resulting triangle. Many Delaunay triangulation algorithms have been proposed in the literature. Some of them are surveyed and evaluated by Su and Drysdale (1997), George and Borouchaki (1998) and Fortune (1987).

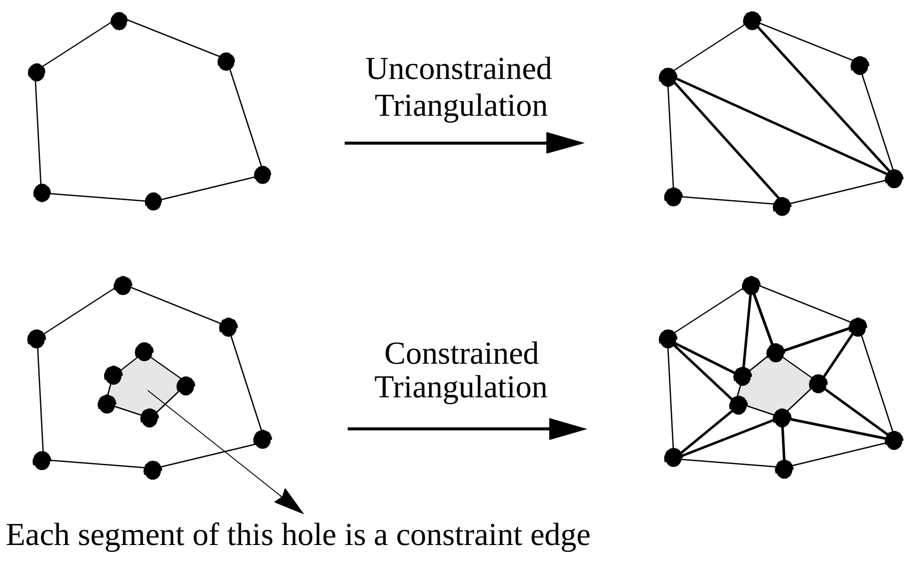

Figure 2.4. Examples of unconstrained and constrained triangulation. (*left column*) input data, (*right column*) resulting triangulation.

If the input data for the mesh generation process not only include a set of points, but also a set of segments joining those points, and those edges must be preserved, the Delaunay triangulation can be forced to maintain certain segments as constraints in the result. Such a triangulation is known as the *constrained Delaunay triangulation* (George, Borouchaki, 1998, pp. 73; Ruppert, 1995; Shewchuk, 1996a, 1996b, 1998). The constrained Delaunay triangulation usually contains Steiner points, because, in general, they are necessary for achieving shape bounds. An example that shows the 2D triangulation process, by applying both unconstrained triangulation and constrained triangulation starting with a given set of points and edges, is illustrated in Figure 2.4.

2.3.2 $2^1/_2$D Triangulation

The input points for a $2^1/_2$D triangulation process are defined in the 3D space. The process utilized to triangulate those points consists of three stages. In the first stage, the 3D space of the input points is reduced to a 2D space by projecting them onto a given reference plane along a predefined direction. The second stage triangulates the projected points in the 2D space by utilizing some of the techniques described in Section 2.3.1. Finally in the third

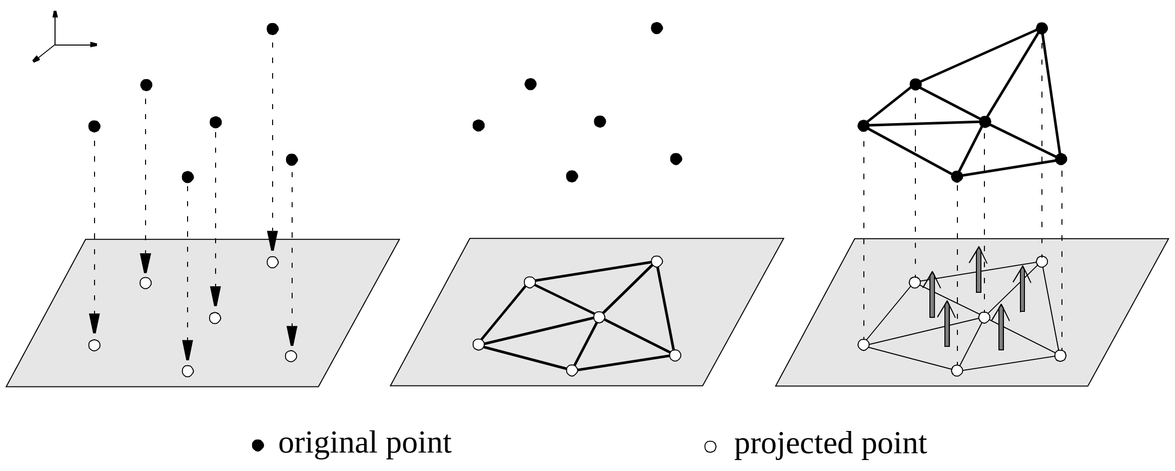

Figure 2.5. Illustration of a $2^1/_2$D triangulation process (Sappa, 1999). (*left*) Projection of the set of points to be triangulated. (*middle*) 2D triangulation of the projected points. (*right*) Mapping of the 2D triangulation to the original 3D space.

stage, the original 3D points are linked according to the resulting 2D triangulation. The $2^1/_2$D triangulation is only valid when the surfaces to which the original 3D points belong can be projected along the predefined direction without overlapping. If the surfaces overlap, a 3D triangulation is the only choice to triangulate the given 3D points (Section 2.3.3). Figure 2.5 illustrates the three stages applied during the $2^1/_2$D triangulation process.

Some work has been reported to generate triangular meshes by projecting the original data (De Floriani, 1989; García 1995b; Fayek, Wong, 1994; García, Sappa, Basañez, 1997b). Similarly to 2D triangulation algorithms, $2^1/_2$D triangulation algorithms can also be classified into triangulation algorithms with or without optimization.

2.3.2.1 $2^1/_2$D Triangulation without Optimization

Taking into account that the input points for a $2^1/_2$D triangulation without optimization are projected from the 3D space to the 2D space, this stage takes advantage of the proposed techniques for 2D triangulation without optimization (Section 2.3.1.1) to generate an initial 2D triangular mesh. Afterwards, it links the original 3D points according to the resulting 2D triangulation.

2.3.2.2 $2^1/_2$D Triangulation with Optimization

Similarly to the $2^1/_2$D triangulation without optimization, the $2^1/_2$D triangulation techniques with optimization take advantage of 2D triangulation algorithms (see Section 2.3.1) in order to generate an initial 2D triangulation. Then, the original 3D points are linked according to the obtained 2D triangulation. Finally, an iterative process flips (swaps) triangle edges whenever the approximation of an underlying discontinuity can be improved after the flip. This process is performed in the 3D space. Thus, the final triangular mesh preserves the surfaces and orientation discontinuities present in the input data as much as possible. These triangulations are sometimes referred to as *data dependent triangulations*. Some algorithms that follow this approach are described in (Dyn, Levin, Rippa, 1990; Brown, 1991; Soucy, Croteau, Laurendeau, 1992; Schumaker, 1993).

2.3.3 3D Triangulation

A triangulation in three dimensions is called a *tetrahedrization* (or sometimes *tetrahedralization*). This triangulation is necessary when it is not possible to project the original 3D data onto a predefined reference plane, since it causes the overlap of the surfaces to which the original points belong. A tetrahedrization is a partition of a set of 3D points into a collection of tetrahedra that only meet at shared faces.

As in two dimensions, the 3D triangulation can be carried out without applying any optimization criteria or by applying one or various optimization criteria simultaneously.

2.3.3.1 Three Dimensional Triangulation without Optimization

When the 2D triangulation is extended to the 3D space, many of the two dimensional properties do not hold any more. An example is that a unique number of tetrahedra can not be guaranteed in the three dimensional case. This means that, given a set of data, there is more than a single, valid 3D triangulation (Lattuada, 1998). Another important problem arises when the input data include non-convex polyhedra, since not all the polyhedra are tetrahedralizable (Bern, Eppstein, 1995). However, Steiner points can be required to solve the tetrahedrization of a non-convex polyhedron. The triangulation algorithms are not generally applied without optimization criteria.

2.3.3.2 Three Dimensional Triangulation with Optimization

In the 3D space, the triangulation algorithms with optimization consider as quality measures the number of tetrahedra contained in the resulting partition, as well as shape parameters. These quality measures are similar to those defined for 2D space triangulations (Section 2.3.1.2). They are usually obtained by adding Steiner points, which allow to improve the quality of the resulting triangular mesh (Bowyer, 1981; Watson, 1981; Joe, 1995). Furthermore, when the Delaunay triangulation is applied in the 3D space, Steiner points can be used to reduce the complexity of the Delaunay triangulation (Bern, Eppstein, Gilbert, 1990). Similarly to the 2D space, the Delaunay triangulation in the 3D space ensures that any point must not be contained inside the circumsphere of any tetrahedra of the mesh. A *circumsphere* can be defined as the sphere passing through the four points of a tetrahedron.

On the other hand, the constrained Delaunay triangulation applied to 3D data allows to maintain polygonal faces in the final triangulation. These constraints define the external and internal boundaries of a model. Some techniques for constrained Delaunay triangulation were proposed by Hazlewood (1993), Weatherill and Hassan (1994), and Cavalcanti and Mello (1999).

2.4 Generation of Adaptive Triangular Meshes from Images

The need to efficiently manipulate and extract the information contained in a digital image has given rise to the development of alternative ways to represent images. Geometric representations, such as adaptive triangular meshes, are gaining popularity as alternative representations of image data, since they lead to compact representations that allow the application of image processing operations directly in the 3D space.

Adaptive triangular meshes can be applied in order to approximate images, since the pixels of an image are considered to be 3D points in a space in which coordinates *x* and *y* represent the columns and rows of the image, and coordinate *z* represents some property of an object contained in the scene (e.g., gray level, color, range, terrain elevation). Additionally, since the obtained triangular meshes are compact representations of the original images, these meshes can later be used for applying further processing algorithms (e.g., decimation, refinement, segmentation, integration). Some of those techniques are overviewed in Section 2.6.

Several algorithms have been developed for generating an adaptive triangular mesh from a given image. Usually, these algorithms first generate a set of points which correspond to pixels sampled from that image by applying any of techniques described in Section 2.2. Afterwards, they use the resulting set of points for generating the corresponding triangular mesh through any of techniques presented in Section 2.3.

2.5 Generation of Images from Triangular Meshes

Besides being able to generate triangular meshes from images, it is also necessary to proceed on the other way round. Hence, it is also necessary to be able to generate images from triangular meshes. This section surveys previous work related to the generation of images from triangular meshes.

Several techniques that work with 2D triangular meshes take advantage of existent methods for image compression in order to reduce the redundancy present in the input data. *Fractal compression* is one of these techniques (Jacquin, 1993; Fisher, Menlove, 1995), which allows to compress and decompress gray level images. Those methods initially obtain a triangular partition (2D triangular mesh) of the given image. That mesh is then coded by grouping the triangles that fulfill a similarity criterion (e.g., homogeneous gray levels). Afterwards, they apply a set of affine transformations to the obtained groups. Later on, the image pixels contained in each of the previous blocks are coded by using some compression method, such as the *Discrete Cosine Transform* (DCT). During the decoding process, the block-based coded residue is decoded and back transformed by applying the inverse affine parameters, and added to the reconstructed image (decoded image). Following this line, different techniques have been proposed in (Davoine, Svensson, Chassery, 1995; Davoine et at., 1996; Altunbasak, 1997).

Alternatively, Lechat, Sanson and Labelle (1997) proposed a technique for rendering images from 3D triangular meshes. They render a 2D model, for example a gray level image, by interpolating the points in a triangular patch with a linear function of its vertex coordinates. This function defines a plane. The same process can be applied to triangular meshes representing color images, although in this case, the rendered process is independently performed on each band (*z* coordinate defining each color). The color bands are finally combined to produce the resulting color image. A similar technique is described in (Wolberg, 1997).

2.6 Triangular Mesh Processing

This section surveys different algorithms that work upon triangular meshes. The processed triangular meshes may represent image approximations (e.g., gray level images, range images, digital elevation maps) or they can model any object of the real world.

2.6.1 Decimation of Triangular Meshes

The aim of the decimation algorithms is to transform a dense triangular mesh into a coarser version, by reducing the number of vertices and triangles needed to represent that mesh, while trying to retain a good approximation of its original shape and appearance. In this way, the decimation process produces a *fine-to-coarse* transformation. The decimation algorithms are commonly applied to dense triangular meshes that are defined in the 2D or 3D space.

Considering that the *approximation error* is the maximum distance between the final triangular mesh and the given initial triangular mesh, the problem of decimating a triangular mesh can be defined in two different ways:

(1) Given a maximum allowed approximation error ξ, find the triangular mesh that ensures that maximum error with the minimum number of points.

(2) Given a predefined number *n* of points, find the triangular mesh that approximates the original data with the minimum approximation error.

A decimation algorithm starts with an original model, such as a triangular mesh containing all the points (pixels mapped to a 3D space) of a given image. Afterwards, an iterative algorithm simplifies that mesh by repeatedly removing triangles or points from it. This decimation process is iterated while the approximation error is below a maximum allowed error (*tolerance*) and until the triangular mesh cannot be further simplified. The result of a decimation process is a simplified representation, or *level of detail* of the given original model. Thus, a largely simplified mesh represents a low level of detail, while a marginally simplified mesh represents a high level of detail. The original mesh represents the highest level of detail.

Some decimation algorithms that work with 2D triangular meshes, which usually represent gray level images, have been proposed in (Davoine, Robert, Chassery, 1997; Lechat, Sanson, Labelle, 1997). Davoine, Robert and Chassery propose the decimation of an original mesh by computing the gray level associated with each triangle of the mesh, and by merging the triangles that have a similar gray level. A similar technique is presented by Lechat, Sanson and Labelle (1997).

Several fine-to-coarse algorithms working with 3D triangular meshes have also been reported in the literature. Some of them are surveyed by Erikson (1996) and Heckbert and Garland (1997). The main difference among the various proposals comes from the heuristics applied to each iteration in order to decide which triangles or points are removed. Schroeder, Zarge and Lorensen (1992) propose an algorithm that significantly reduces the number of triangles required to model a physical object. The algorithm is based on multiple filtering phases. It locally analyses the geometry and topology of the mesh and removes vertices that pass either a minimal distance or curvature degree criteria.

Cohen et al. (1996) propose the *simplification envelopes* method, which supports bounded error control by forcing the simplified mesh to lie between two offset surfaces. In (Cohen, Olano, Manocha, 1998), a new algorithm that allows not only the generation of low polygon count approximations of an original model but also the preservation of its appearance is presented. This algorithm converts the input model to a representation that decouples the sampling of three model attributes (color, curvature and position), storing the colors and normals in texture and normal maps. Then, a *texture deviation metric* is applied. This technique guarantees that these maps shift by no more than a user specified number of points. Following this line, different 3D decimation techniques are described in (Turk, 1992; Garland, Heckbert, 1995, 1997; Ciampalini et at., 1997; Amenta, Bern, Kamvysselis, 1998; Lindstrom, Turk, 1999; Trotts, Hamann, Joy, 1999; Guéziec, 1999).

Another approach applied to 3D triangular meshes consists of minimizing an energy function which simultaneously considers the number of vertices and their approximation error at each iteration. Different definitions of the energy function are presented by Terzopoulos and Vasilescu (1991), Vasilescu and Terzopoulos (1992) and Hoppe et al. (1993).

2.6.2 Refinement of Triangular Meshes

Contrarily to decimation algorithms, triangular mesh refinement algorithms start with a coarse triangulation, which has a reduced set of points, typically chosen from all the original points. Then, an iterative algorithm proceeds by adding more and more detail to local areas of the mesh at each step. In this way, more points are added and new triangular meshes are generated at each iteration. This procedure is performed until the maximum approximation error between the current triangular mesh and the given initial triangular mesh is below or equal to some required tolerance. A characteristic of this algorithm is that if the given error tolerance is high, this method obtains a triangular mesh which is a greatly simplified representation of the original model, whereas, if the error tolerance is low, it marginally simplifies the original model. Refinement algorithms are usually referred to as *coarse-to-fine algorithms*.

Altunbasak (1997) proposes a refinement technique for 2D triangular meshes. This technique initially generates a triangular mesh from a given image by using a 2D mesh generation method (Section 2.3.1). If the obtained triangular mesh approximates the given image with an approximation error below the specified tolerance, the algorithm concludes. The algorithm can also conclude if a predefined number of points is obtained. On the other hand, the algorithm applies a criterion for selecting triangles to be divided into three or four sub-triangles. It chooses the triangles which would yield the largest reduction of peak signal-to-noise ratio (PSNR). Another work in this line is presented by Davoine and Chassery (1994). Given an initial triangular mesh which is generated from a set of points sampled in a uniform way from a gray level image, the algorithm generates local modifications of the mesh by applying a split and merge method. The split step consists of adding a point at the barycenter of each non homogenous triangle by considering a gradient criterion. The split process continues until convergence. Thus, it stops when either the triangles are homogenous or the areas of the triangles are below a given threshold. The merge step consists of removing neighboring triangles with similar mean gray levels.

Some coarse-to-fine algorithms working with 3D triangular meshes are presented in (Rivara, 1996; Schneiders, 1996; Canann, Muthukrishnan, Phillips, 1996; Hoppe, 1997; Wilson, Hancock, 2000). A coarse-to-fine pioneering work was presented in (De Floriani, 1989). There, the farthest point from the current triangular mesh is selected at each iteration and inserted into the triangulation. The latter is updated accordingly.

2.6.3 Smoothing of Triangular Meshes

A problem with mesh generation methods is that they can produce poorly shaped elements. Such elements are sometimes undesirable for the final mesh. An approach that can be utilized to correct this problem consists of adjusting the mesh point locations in such a way that the overall topology of the generated mesh is preserved. In this way, the application of this process allows to reduce the element distortions and improve the overall quality of the mesh. Typically, the previous approach is known as *mesh smoothing*. Some algorithms that work with 3D triangular meshes are introduced below.

A commonly used technique is *Laplacian smoothing* (Field, 1988). This method moves a point of the mesh to the average location of any point connected to it by an edge. Some mesh smoothing methods working with Laplacian smoothing were proposed by Freitag, Jones and Plassmann (1995), Freitag (1997), and Canann, Tristano and Staten (1998).

Another mesh smoothing technique is *optimization-based smoothing*. This method measures the quality of the surrounding elements of a point and tries to optimize that quality measure by computing the local gradient of the element quality with respect to the point location. The point is moved in the direction of the increasing gradient until convergence. Freitag (1997), Amenta, Bern and Eppstein (1997), and Canann, Tristano and Staten (1998) present some algorithms that apply optimization-based smoothing.

Following the mesh smoothing line, different techniques are described in (Bossen, Heckbert, 1996; Shimada, 1997; Balendran, 1999). Additionally, a survey is presented in (Owen, 1998).

2.6.4 Segmentation of Triangular Meshes

In order to segment a triangular mesh that approximates an image into uniform regions, two different techniques have been proposed: *edge-based segmentation* and *region-based segmentation*. A combination of both techniques is also possible. In the latter case, the segmentation process is known as a *mixed segmentation*. When an edge-based segmentation is used, the generation of a region is based on the detection and connection of discontinuities that belong to the same region. On the other hand, in a region-based segmentation process, regions are formed by grouping adjacent triangles that have a similar orientation or, alternatively, according to some value associated with the triangles.

Segmentation is a typical stage of recognition processes in computer vision, in order to extract the objects present in a certain scene. Some segmentation techniques that work with 2D triangular meshes have been proposed by Gevers and Kajcovski (1994) and Gevers and Smeulders (1997). Alternatively, some segmentation techniques that work with 3D triangular meshes have been presented in (Cohen I., Cohen L., Ayache, 1992; Delinguette, Hebert, Ikeuchi, 1992; García, Basañez, 1996; Malladi et at., 1996; McInerney, Terzopoulos, 1997; Mangan, Withaker, 1999).

2.6.5 Warping of Triangular Meshes

Image warping or morphing refers to the process of deforming a 2D or 3D image into another one in a continuous manner. This process can also be applied to a triangular mesh representing an image by using affine transformations that allow to deform the polyhedrons (or polygons in 2D) contained in the mesh. A warping process is potentially useful for modeling biological and physical processes, such as muscle contraction, for which deformations are seamless in nature.

Fujimura and Makarov (1997) present a shape warping method that continuously deforms an image (spatially and temporally) without foldover, while observing a given set of trajectories of feature elements between initial and final shapes. This warping process is applied to a 2D triangular mesh that represents an image. Following this line, different approaches are described in (Goshtasby, 1987; Terzopoulos, Metaxas, 1991; Parent, 1992).

2.6.6 Integration of Triangular Meshes

Given a triangular mesh that approximates a certain image (e.g., gray level image, range image, digital elevation map), the objective consists of merging it with another mesh representing a new view or a reconstructed model, generating thus a non-redundant surface model that becomes the new current reconstructed model. Several applications have benefitted from the previous process, including world modeling, reverse engineering and object segmentation or recognition.

Two different methods have been proposed in the literature in order to integrate meshes: *(1) unstructured* and *(2) structured methods*. A brief overview of these methods is presented in (Sappa, 1999). Unstructured integration methods directly work by using the original data points, generating a polygonal surface from that arbitrary set of unorganized points. The integration, in this case, is obtained by collecting all the points from the different views. On the other hand, structured integration methods work by using the original data arranged in some way. In particular, most of the structured methods assume that the different views to be integrated have been properly triangulated. Therefore, they can make use of topological and geometric information associated with each point (e.g., neighborhood, curvature, surface orientation). Some relevant contributions that work with unstructured methods have been presented in (Boissonnat, 1984; Hoppe et al., 1992). Additionally, techniques that work with structured methods applied to 3D triangular meshes are presented in (Turk, Levoy, 1994; Sappa, García, 2000).

2.6.7 Compression of Triangular Meshes

Triangular mesh compression techniques can be divided into two basic categories: *geometric compression* and *topological compression*. The geometric compression techniques allow to reduce the redundant geometric information of a triangular mesh by generating a representation with fewer data points than the original mesh. This initial mesh is usually a dense triangular mesh containing, for example, all the original points of a given image. Several relevant contributions working in this line are presented in Section 2.6.1 (decimation of triangular meshes).

On the other hand, topological compression techniques seek the reduction of information necessary to store geometric representations such as triangular meshes. Taubin and Rossignac (1998) propose a topological compression algorithm that allows to encode the connectivity information of a triangular mesh by constructing two interlocked trees: a spanning tree of vertices and a spanning tree of triangles. The first tree stores the positions of all the vertices of the given mesh, by considering the offsets between the 3D positions of adjacent vertices. The tree of triangles allows to encode the connectivity information of all the triangles of the mesh. The position of each triangle with respect to one of its neighbors can then be codified with a pair of bits.

Similarly to the previous technique, some algorithms that use topological compression applied to triangular meshes have been proposed in (Deering, 1995; Touma, Gotsman, 1998; Gumhold, Straβer, 1998; Rossignac 1999). Since the representations generated through geometric compression can be further topologically compressed, both compression techniques are complementary.

2.7 Manipulation of Compressed Images

Although image compression standard formats, such as *gif* and *jpeg* (Kunt, Ikonomopoulos, Kocher, 1985; Wallace, 1991), allow a substantial reduction of the size of digital images, those techniques were not originally devised for applying further processing operations directly in the compressed domain. Thus, images codified in those formats must be uncompressed prior to being able to apply image processing operations upon them, no matter how big and redundant the images are. Nonetheless, some researchers have managed to apply various basic operations upon such compressed representations. This section presents a short review of techniques that work in the domain of compressed images.

Smith and Lawrence (1993) describe some techniques that allow to perform algebraic operations (addition and multiplication) directly on compressed *jpeg* data. Other techniques for image and video manipulation working in the compressed domain are presented in (Chang, 1995a; Chang, 1995b). Those algorithms perform image filtering, geometrical transformations, overlapping, pixel multiplication and convolution. Following this line, similar techniques for manipulating compressed images are described in (Chitprasert, Rao, 1990; Natarajan, Bhaskaran, 1995; Shen, Sheti, 1996).

In all cases, the range of processing operations that can be preformed in the compressed domain is quite limited, excluding many important operations commonly utilized for image processing and computer vision.