The Visual Computer

A texture-mapping approach for the compression of colored 3D triangulations

Marc Soucy, Guy Godin, Marc Rioux

Visual Information Technology Group, Institute for Information Technology, National Research Council of Canada, Ottawa, Canada K1A 0R6
e-mail: msoucy@innovmetric.com, godin@iit.nrc.ca

We present an algorithm that constructs compact and realistic descriptions of colored 3D objects using texture mapping on compressed triangulations. A high-resolution triangular mesh model is created by integrating measurements from a color 3D laser sensor. Each vertex is attributed with a RGB color value. The high-resolution triangulation is transformed into a compressed triangulation and a texture map. This map embeds the color information of the vertices removed during the geometric compression and projected on the lower resolution triangulation. We describe the algorithm for the rapid and efficient construction of a texture map for compressed triangulations of arbitrary topology. Experiments show that high compression rates can be achieved while maintaining good visual similarity between the original and compressed models.

Key words: Triangulation – Geometric compression – Texture mapping – Multiresolution – Color digitizing

Correspondence to: G. Godin

1 Introduction

Computer-generated virtual environments rely on efficient interactive computer graphics and require the real-time display of colored 3D surface models. Most of the current applications of this technology are based on the generation of purely synthetic universes. However, many others, such as simulators or virtual museums, need to incorporate representations of real 3D objects or scenes – in other words, to "virtualize reality". Manually creating geometric models of these objects through the use of a CAGD system is often tedious, and sometimes impossible. This is particularly true for objects with irregular geometries such as natural objects or works of art. In this context, laser range sensors (Besl 1988) are finding numerous new applications in geometric modeling for industry or animation. They provide a tool for digitizing 3D surfaces, usually in the form of arrays of 3D coordinates. In addition to the surface geometry information, some laser range sensors, such at the one developed at the National Research Council of Canada (Rioux 1984; Baribeau et al. 1992), are designed to record the intensity of light in one or several wavelengths that is reflected at each sensed point. The result of the sensing then consists of a set of 3D points with an attached trichromatic (RGB) color value, which provide the raw data to the creation of a geometric and color computer model of objects. Virtual models can then be created that not only reflect the object's geometry, but also its actual surface colors, instead of a synthetic pattern applied to the surface. The first step in reconstructing a surface representation from a set of color range images is to create a surface description based on triangular facets with color information attached to each vertex. A method for the automatic creation of a unified triangulation from these possibly redundant data is described by Soucy and Laurendeau (1995a, b). In order to generate a geometric representation that describes fine variations in the object shape and color, and to avoid making a priori assumptions on the object surface, it is generally necessary to use a small surface-sampling step. Therefore, the colored 3D triangulations built from color 3D digitizer data often contain a large number of triangles – in the order of hundreds of thousands. This represents an important limitation when the target application is the inclusion of the object model within a virtual environment. This paper

The Visual Computer (1996) 12:503–514
© Springer-Verlag 1996

addresses the problem of compressing high-resolution colored 3D triangulations to create compact and realistic geometric descriptions that can be displayed in real time on state-of-the-art graphics workstations. We propose a two-step approach for the generation of compact texture-mapped models from high-resolution colored models. First, the initial triangulation is decimated with a new compression technique that maintains a mapping between the original vertices and the compressed 3D shape. The mapping information is then used to generate a texture map automatically for the compressed 3D shape.

Several approaches have been proposed to model a surface triangulation at various levels of resolution. De Floriani (1989) and Schmitt et al. (1991) first compute a coarse triangulation from a user-selected set of points and edges. Vertices are then iteratively added to this model to refine the initial approximation. Both methods require that the set of vertices be mapped onto a 2D plane. The algorithm reported by Schroeder et al. (1992) takes as input a high-resolution 3D surface triangulation. Vertices that minimize the triangulation error are then removed iteratively to compress the initial triangulation. The error criterion is the distance between a point and the average plane modeling its neighborhood. A method for compressing a high-resolution triangulation is also presented by Hoppe et al. (1993). The resulting triangulation is obtained through a least-squares minimization technique. Therefore, the error criterion uses a quadratic norm. The hierarchical triangulation algorithm reported by Soucy and Laurendeau (1996) uses a sequential optimization process to remove, at each iteration, the vertex minimizing the retriangulation error. The vertices used to generate compressed triangulations are thus a subset of the original set of vertices. The operator controls the compression process by specifying a 3D tolerance level, defined as the maximum 3D distance between the original model vertices and the compressed surface representation. Once the triangulation error crosses a specified tolerance level, the compressed representation is saved in a file. Two major improvements have been brought to this original algorithm. First, the time complexity has been reduced almost to $O(n)$, which makes the compression of large models more tractable. Second, the new compression technique maintains a mapping between the original vertices and the compressed model. Once a model has been compressed, all removed vertices are attributed the barycentric coordinates (u, v, w) of their projection on the larger triangles of the remaining model. Thus each triangle of the compressed model has a color triplet at each vertex, as well as a variable number of removed vertices mapped onto its planar surface. Each carries three barycentric coordinates (u, v, w) and a color triplet (RGB). The color information associated with the removed vertices is transformed into a high-resolution texture map to be applied onto the compact triangulated mesh.

Texture mapping is an efficient technique that has been developed to give the illusion of a complex object with a small number of polygons (Heckbert 1986; Haeberli et al. 1993). A texture map is defined as a rectangular array of RGB colors. An element in this array has two discrete coordinates (s, t) and is the center of a square of side 1. The squares associated with each array element are named texels. In order to texture map a 3D polygon, two real texture coordinates (s, t) are associated with each polygon vertex. These texture coordinates define a corresponding polygon in the texture space. When a 3D polygon is rendered, a filling algorithm shades each screen pixel enclosed within its boundaries. Defining a corresponding texture polygon allows the interpolation of the surface material colors in the texture space instead of using a single color for the whole triangle. Texture mapping enables the rendering of complex surfaces such as trees, water, and mountains by mapping a realistic or photographic picture onto simple planar polygons, such as triangles and rectangles. The same idea is exploited in this paper, with the notable difference that the texture to be mapped is not an independently defined 2D image, but the result of the projection of color information from a high-resolution 3D triangulated geometric model onto a decimated one. Such a triangulated model with attached color may come, as is the main motivation for the work presented here, from a laser color-range sensor, but also from a geographical information system or a scientific visualization system. Traditional usages of texture mapping can be seen as mostly applications of a 2D pattern onto a 2D surface, or the intersection of a 3D volume texture with the

rendered 2D surface, in the case of simulation of sculptured material such as wood or marble. The situation described in this paper differs from these two classic cases in that it is concerned with the projection of spatially located color information onto a surface embedded in three dimensions with no a priori restriction on the topology of the triangulation and its efficient storage in a 2D texture map.

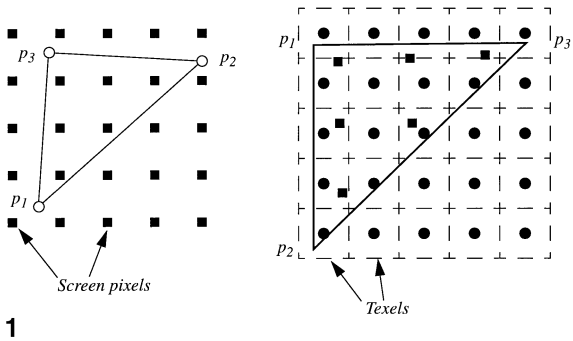
This paper presents an algorithm for generating a compact texture-mapped model having the visual impact of a high-resolution surface model with a smaller number of triangles. It is assumed that an initial high-resolution 3D triangulation is available and that RGB color components are attached to each triangulation vertex. This model is input to the multiresolution triangulation algorithm described by Soucy and Laurendeau (1996), yielding a compressed triangulation. At the end of this process, all removed vertices have three barycentric coordinates (u, v, w) that define their mapping on one triangle of the compressed triangulation. Generating a texture map for a triangulation raises three important problems: 1. How will the texture space be shared among the triangles? 2. What values will be given to each texel of the texture map? 3. How will the continuity of the texture space be ensured? All these aspects must be considered simultaneously in order to design an efficient algorithm to build a texture map for a compressed triangulation. Section 2 overviews the original solution that was developed to overcome these problems. The texture space is tessellated to accommodate the triangles of the compressed model. Each triangle has its own texture space. Each high-resolution vertex is projected in the texture space and its RGB colors are associated with the nearest texel center. An interpolation algorithm finally assigns to each of the remaining empty texels the RGB values of the nearest texel center on the 3D surface. Constraints are imposed to ensure that the texture space is continuous. Sections 3 and 4 address more specific aspects of this construction scheme. The method for tessellating the texture map is presented in Sect. 3, while the interpolation algorithm is described in Sect. 4. In Sect. 5, experimental results illustrate the performance of the proposed algorithm.

2 Construction of a texture map for a triangulation

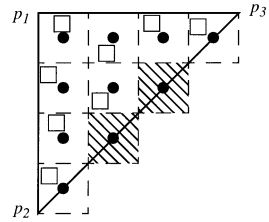
This section overviews the original solution that was designed to generate a texture-mapped model from a compressed colored 3D triangulation. The texture map is tessellated so that each triangle of the compressed model has its own texture space. Adjacent 3D triangles are not mapped contiguously onto the texture map so that the representation of models of any topology is possible. However, texture continuity between triangles is enforced by constraining the shape and size of the texture triangles and the RGB values of the edge texels. To assign RGB values to the texels of the map, the high-resolution vertices are projected in the texture space and their RGB colors are assigned to the nearest texel center. If two vertices are projected onto the same texel, their RGB components are simply averaged. A 3D nearest-neighbor interpolation algorithm is used to assign RGB values to empty texels into which no vertices were projected.

It may be useful to start with a summary of how a texture map is used to interpolate surface material colors for polygon rendering. A triangle having vertices p_1–p_2–p_3 is drawn on a computer screen (Fig. 1). These 3D vertices are first projected on the 2D screen space with the appropriate transformation. It should be noted that the projected vertices do not generally coincide with discrete screen pixels. A filling algorithm is used to determine which screen pixels are enclosed within the projected triangle. If texture coordinates have been defined for the triangle, it is possible to interpolate the surface material colors onto the texture map. In this paper, a nearest-texel texture rendering mode is chosen: the reasons for this choice are discussed later. The barycentric coordinates of a screen pixel with respect to the projected triangle are first computed. These coordinates are then used to determine a real position on the texture map. The surface material colors associated with the screen pixel are chosen as the RGB colors of the nearest texel center on the texture map. The dimension of a triangle in the texture space determines the texture resolution on the screen that can be reached for that geometric primitive.

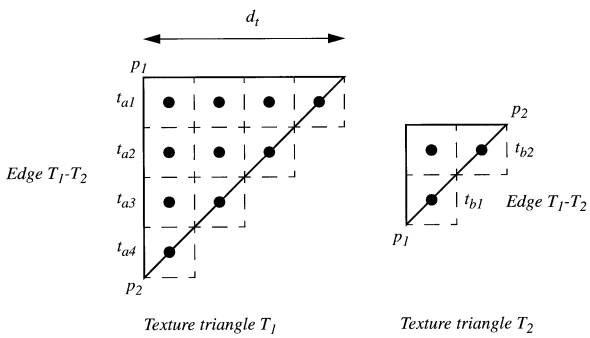
The first issue to address is the tessellation of the texture map. Each triangle of the compressed



1



3



2

Fig. 1. Interpolating the surface material colors in the nearest-texel texture-rendering mode

Fig. 2. Mapping of two adjacent 3D triangles in texture space. Continuity is achieved by the constraint that $t_{b1} = t_{a1} = t_{a2}$ and $t_{b2} = t_{a3} = t_{a4}$

Fig. 3. Projection of original vertices (represented by *white squares*) onto a texture triangle. The *hatched texels* are empty, and their RGB values have to be interpolated

triangulation must have its corresponding texture triangle on the texture map. We decided to map each triangle independently onto the texture map. This independent mapping of the triangles allows us to process any triangulation topology. As a result, adjacent 3D triangles are not necessarily adjacent, once they are projected onto the texture map. This design choice implies that the shape and size of the texture triangles and the RGB values of the texels have to be constrained in order to achieve texture continuity over the triangulation. In the context of this paper, a texture map is defined as continuous if the rendered image of any triangulation vertex or edge is uniquely defined. Indeed, vertices and edges are usually shared by more than one triangle (except along triangulation boundaries) and, consequently, are described by the same number of texture triangles. Texture continuity at a vertex of the triangulation can be achieved with constraint that all texels into which the vertex is mapped have the same RGB values. This constraint ensures that the image of a vertex is uniquely defined. To achieve texture continuity for a 3D edge, one must ensure that all texture edges corresponding to this edge are equivalent. A 3D edge shared by two adjacent triangles has two texture edges. Three conditions must be imposed to meet the continuity requirement. Firstly, the three vertices of a triangle must lie directly on texel corners. Secondly, a texture triangle must be a half-square triangle of dimension d_t, i.e., a triangle resulting from the diagonal subdivision of a square of side d_t. The advantage of using half-square triangles is that d_t texels now lie on each of its three edges (see Fig. 2). Finally, the ratio of the largest over the smallest of the dimensions d_{t1} and d_{t2} of two texture triangles that are adjacent in 3D space must be an integer number. Figure 2 shows how texture continuity between two adjacent triangles can be achieved when these conditions are verified. Triangle t_1 and t_2 are adjacent by edge t_1–t_2. Their vertices lie on texel corners, and they are both half-square triangles of dimension 4 and 2 respectively. Texture continuity may then be simply achieved with the constraint that texels t_{b1}, t_{a1} and t_{a2} be equal and that texels t_{b2}, t_{a3} and t_{a4} be equal. In addition to the continuity constraints, it is also necessary to allow different sizes of texture triangles, since the corresponding 3D triangles do not have the same sizes and may contain different

amounts of color data. A hierarchy of triangle dimensions based on powers of 2 has been chosen for that purpose. This choice meets the requirement of an integer ratio of triangle sizes, while being well adapted to current sizes of hardware texture memory on numerous graphic systems. Section 3 presents an algorithm to select texture triangle sizes and tessellate the texture map.

Once each triangle has its own texture space, color data attached to each original vertex of the high-resolution triangulation are projected in the texture space. Vertices that are part of the compressed triangulation are mapped directly into the texture space by means of their texture coordinates. As for removed vertices, their barycentric coordinates are used to map them in the texture space of their circumscribing triangles. The RGB colors of a removed vertex mapped onto a texture triangle are assigned to the nearest texel center. If several vertices are mapped onto the same texel, their RGB contributions are averaged. The RGB colors of texels that are constrained to be equal in order to have texture continuity are also averaged. Figure 3 illustrates the projection of the original vertices onto a triangle. Triangle p_1–p_2–p_3 is described by ten texels in the texture space. The original color 3D vertices have been mapped onto eight of these texels, thus setting their RGB values. As in Fig. 3, some texels may remain empty after all original vertices have been mapped in the texture space. An interpolation step is thus required to assign RGB values to these empty texels. To get as close as possible to the original color data and avoid the use of an arbitrary interpolation function, we decided to implement a 3D nearest-neighbor interpolation algorithm. The RGB values of an empty texel should be equal to the RGB values of the nearest texel center on the 3D surface modeled by the triangulation. An iterative 3D interpolation algorithm is described in Sect. 4.

3 Tessellation of a rectangular texture map

This section presents an algorithm for the tessellation of a rectangular texture map into a set of half-square texture triangles having dimensions that are powers of 2. It is assumed that a $W \times H$ texture map is available and that the width W is a power of 2. A texture filtering scheme independent of the 3D shape compression is also described. Texture filtering may be required to achieve antialiased imaging of the 3D models.

Let us define D_i as the average distance between the vertices of the high-resolution triangulated models. The average distance between the original vertices mapped onto a compressed triangulation can also be reasonably approximated by D_i. To obtain a high-resolution texture map and avoid as much as possible the averaging of RGB color data, the original vertices should be mapped onto distinct texels. This can be achieved by defining half-square texture triangles that oversample the projected color vertices. Let us consider E_{max}, the length of the longest edge of triangle T. Then, one can reasonably expect that, on average, the original vertices mapped onto triangle T will lie on distinct texels if the corresponding texture triangle has a dimension larger than E_{max}/D_i. Therefore, the ratio E_{max}/D_i is computed for each triangle of the compressed triangulation and rounded up to the next power of 2 in order to set the dimension of the corresponding texture triangles. Now that initial dimensions have been defined for the texture triangles, two important issues must be addressed. Firstly, one must design an efficient method for the tessellation of a $W \times H$ texture map into a set of texture triangles. Secondly, one has to devise a technique to reduce the dimension of the texture triangles if the size of the required texture map is larger than the available texture-map space.

To simplify the tessellation algorithm, triangles are paired to form rectangles. The pairing of half-square triangles is illustrated in Fig. 4. Triangles of equal dimension d_t are paired to form rectangles of dimension d_t by $(d_t + 1)$. It is worth noting that the width of these rectangles remains a power of 2. Since the width W of the texture map is also a power of 2, we can implement a very simple custom tessellation algorithm. Rectangles are placed in the texture map from the largest to the smallest, always at the highest position available in the texture map. Since rectangle widths and the texture map width W are powers of 2, the texture map is always fully used in the x-axis. An example of a tessellated texture map is shown in Fig. 5. One may notice that the only loss of texture space occurs at the bottom of the map.

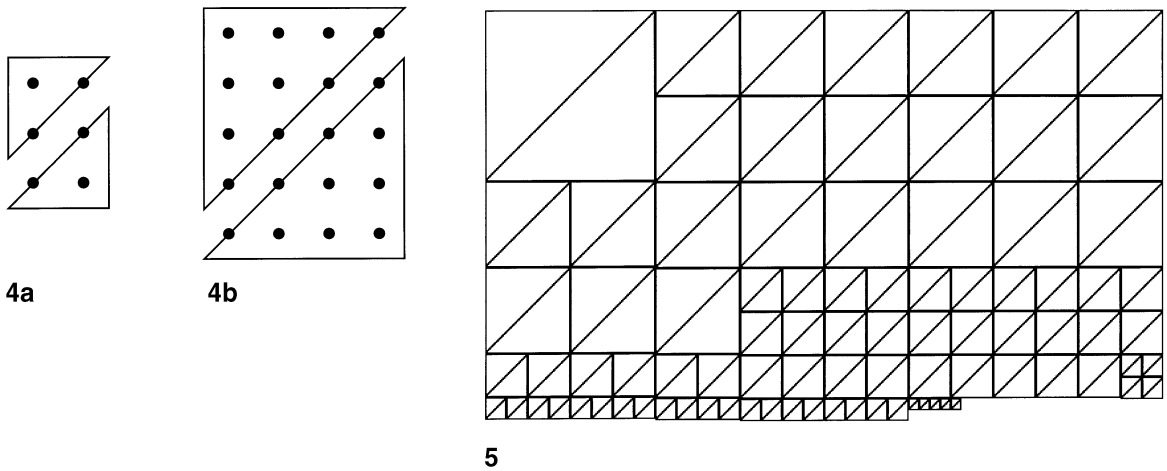


Fig. 4. Half-square triangles are paired to form rectangles: **a** two triangles of dimension 2 form a 2×3 rectangle; **b** two triangles of dimension 4 form a 4×5 rectangle

Fig. 5. Example of a tessellated texture map

The tessellation algorithm described is adequate when the required texture map is smaller than $W \times H$. However, it would be desirable to design a tessellation algorithm capable of dealing with arbitrary texture-map sizes. A texture map must contain a minimum of $3*n\_tri$ texels to accommodate a triangulation, where *n_tri* is the number of triangles. This corresponds to a situation in which each texture triangle is a half-square triangle of dimension 2. An algorithm for the tessellation of any texture map larger than this minimal texture map is presented here. If the allocated texture map is smaller than the space needed for the initial texture triangles, the texture triangle dimensions are iteratively reduced until the texture map can accomodate all triangles. As stated in Sect. 2, multiple color vertices projected onto one texel are averaged. Therefore, reducing the texture-triangle dimensions results in a low-pass filtering of the color information. The size of the texture map can thus be used to control the level of texture filtering, independently of the 3D shape compression achieved by the hierarchical triangulation algorithm. This texture-filtering process may be helpful to eliminate the aliasing effects observed when a high-resolution texture map is used to display an object located far from the observer. Indeed, standard approaches for texture-map filtering cannot be applied to our texture maps, since the triangles are not necessarily mapped contiguously onto the texture map.

Texture filtering is achieved through an iterative algorithm that reduces the size of the initial texture triangles. As stated in Sect. 2, the dimensions of the texture triangles are powers of 2. Each power of 2 may thus be considered as a class to which a given number of triangles belong. The classes are ordered from the largest to the smallest. At each iteration, the filtering algorithm reclassifies the triangle nearest to the next inferior class. The ratio between E_{max}/D_i, as we have defined it, and the size of the nearest inferior class is defined as the proximity coefficient of a texture triangle. Let us consider a pair of triangles T_1 and T_2 for which E_{max}/D_i equals 17.5 and 11, respectively. Initially, their corresponding texture triangles have dimensions 32 and 16 since E_{max}/D_i is rounded up to the next power of 2. To reduce the sizes of the texture triangles, their proximity coefficients are evaluated and are equal to $17.5/16 = 1.09375$ and $11/8 = 1.375$, respectively. Therefore, triangle T_1 is the nearest to an inferior class at iteration 1 and the dimension of its texture triangle is set to 16. The proximity coefficient of triangle T_1 is now equal to $17.5/8 = 2.1875$. Hence, triangle T_2 is the nearest to an inferior class at iteration 2 and the dimension of its texture triangle is set to 4. This reduction process can be

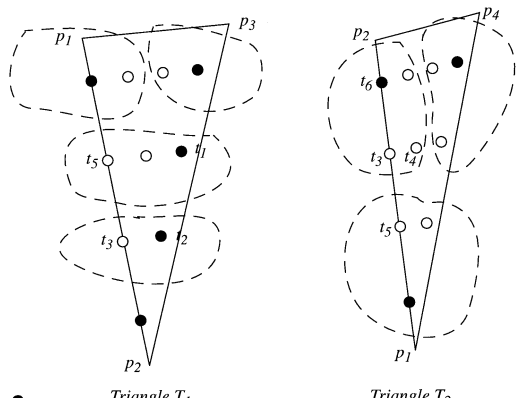
performed, in the worst case, until all texture triangles have the dimension 2. In practice, the size of the allocated texture map limits the reduction process. At each iteration, the size of the required texture space needed to accommodate the actual set of texture triangles is evaluated. Once the space becomes smaller or equal to the size of the allocated $W \times H$ texture map, the iterative process is ended and the texture map can effectively be tessellated. Just before tessellation, an adjustment step takes place to ensure that there is an even number of triangles classified in each dimension class but the smallest. This guarantees that no texture space is wasted between classes.

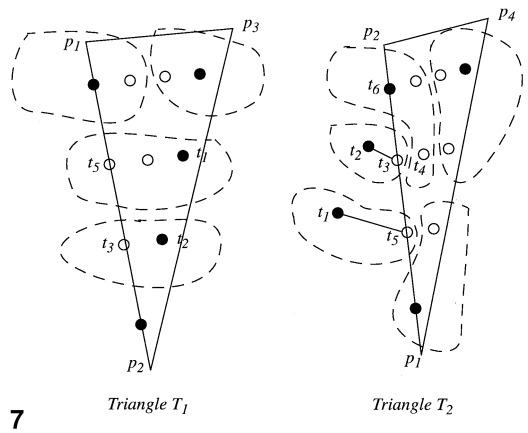
4 Three-dimensional nearest-neighbor interpolation of empty texels

The projection of the original vertices onto the texture map may leave numerous empty texels, as illustrated in Fig. 3. However, RGB values must be defined for all texels enclosed within texture triangles in order to define a usable texture-mapped model. An interpolation step is thus needed to assign RGB values to these empty texels. To get as close as possible to the original color data and to avoid the use of an arbitrary interpolation function, we decided to implement a 3D nearest-neighbor interpolation algorithm. The RGB colors of an empty texel should be equal to the RGB colors of the nearest texel center on the 3D surface modeled by the triangulation. An iterative algorithm for this 3D nearest-neighbor interpolation is presented in this section.

The texels of a texture map are distributed along a square grid topology. Finding the nearest neighbor within a texture triangle can thus be efficiently implemented with a distance transformation. A distance transformation is an operation that converts a binary picture, consisting of feature and nonfeature elements, to a picture in which each element has a value that approximates the distance to the nearest feature element (Borgefors 1984). In this paper, a feature element is the center of a texel that has RGB values and a nonfeature element is the center of an empty texel. A distance transformation is computationally efficient since it can be implemented as a sequence of raster scans. As a result, such a transformation has a linear time complexity functions $O(n)$, where n is the size of the picture. Several measures of distance are reported in the literature (Borgefors 1984). In terms of distance precision, the best technique reported by Borgefors (1984) is the 8-neighbor euclidean distance transformation first proposed by Danielsson (1980). An optimized implementation can be found in Leymarie (1992). The key advantage of this technique is that its maximal error is bounded by a constant. Therefore, the precision achieved by the technique will be the same for all triangles even though these triangles may have different sizes. A modified 8-neighbor euclidean distance transformation has been designed and implemented in this work. Distances are computed with the 3D coordinates of the texel centers instead of the 2D texel coordinate metric. In addition to the 3D nearest-neighbor distance, an integer number is also attributed to each empty texel. This integer number is the index of the nearest neighbor on the texture map, which is eventually used to copy the RGB values of the nearest neighbor onto the empty texel. Once the distance transformation has been applied to a texture map, each empty texel within a texture triangle has been assigned a nearest neighbor within that triangle. Figure 6 shows the results of applying the distance transformation to a texture triangle. The texel centers are mapped onto the 3D triangles to illustrate the effects of the algorithm. The nearest-neighbor regions surrounding each nonempty texel are drawn with dotted contours. These regions are analogous to Voronoi cells in a 2D Voronoi diagram except that they represent 3D neighborhoods on a surface. In Fig. 6, nearest-neighbor regions are shown for two adjacent triangles T_1 and T_2. One can notice that the centers of the two empty texels t_3 and t_5 lying on the edge p_1–p_2 are not part of the same regions in the two triangles. It is even possible that the nearest neighbor of an interior texel of triangle T is part of another triangle (see for example texel t_4 of Fig. 6). Hence, a single distance transformation is not sufficient to implement a global 3D nearest-neighbor interpolation technique.

It is necessary to design a way to propagate the distance transformation results between triangles that are adjacent in the model, but not necessarily on the texture map. Therefore, the nearest-neighbor interpolation has been designed as an iterative process. Following the first distance transformation, all empty texels lying on triangle edges are





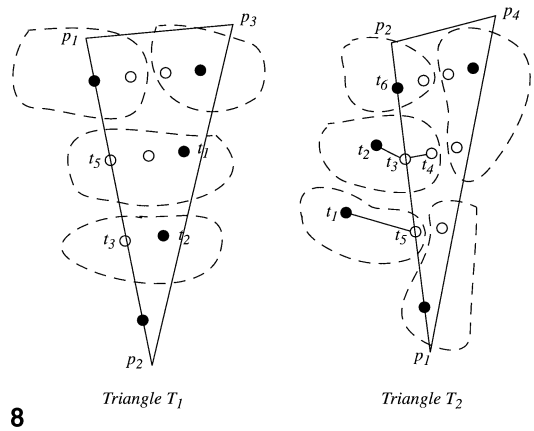


Fig. 6. Nearest-neighbor regions surrounding each nonempty texel following the first distance transformation performed on triangles T_1 and T_2

Fig. 7. Nearest-neighbor regions after distance propagation across edge

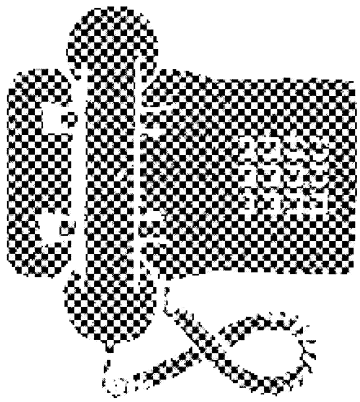
Fig. 8. Nearest-neighbor regions following the second distance transformation

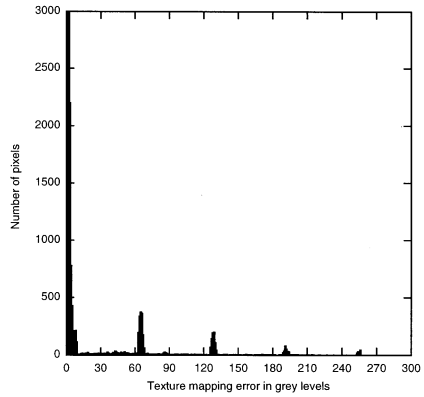
found. For each of these edge texels, the best nearest neighbor is determined, among the two possible nearest neighbors found independently in the two triangles adjacent to the edge. A second distance transformation is then applied to the texture map. During this distance transformation, edge texels are considered feature elements and are thus equivalent to nonempty texels. However, a distance offset must now be added to distances measured between an edge texel and an empty texel. This distance offset is the 3D distance between the edge texel center and its nearest neighbor. The total distance between an empty texel center of triangle T and its nearest neighbor now becomes the sum of two distances: one computed by the distance transformation within triangle T and one representing a distance computed in triangles adjacent to T. Following the second distance transformation, results are again propagated between the triangle edges. The process is iterated until no new nearest neighbors are found for the empty texels. Since this iterative algorithm uses distances that are the sum of distances computed within triangle boundaries, it performs a true 3D nearest-neighbor interpolation on the surface of the model. The results of propagating the distance transformation on the triangles of Fig. 6 are shown in Fig. 7. One can notice that the nearest neighbors of texels t_3 and t_5 are now respectively t_2 and t_1, which are part of triangle T_1. A second distance transformation is then applied to the two triangles and as a result, texel t_4 of triangle T_2 has a new nearest neighbor part of triangle T_1, as can be observed in Fig. 8. Indeed, the sum of distances t_2–t_3 and t_3–t_4 is smaller than the distance between t_6 and t_4. Texel t_2 is thus the nearest neighbor of texel t_4 on the surface modeled by the triangulation.

Once the nearest neighbors are determined for all empty texels, the RGB values of the nearest neighbors are copied in the empty texels. The continuity constraints previously described in Sect. 2 remain valid for edge texels.

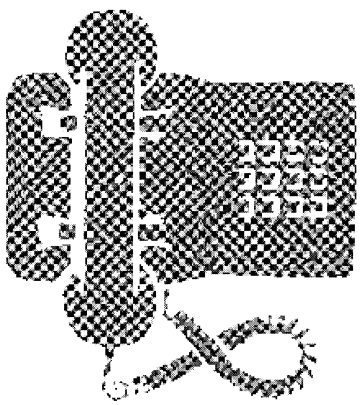
5 Experimental results

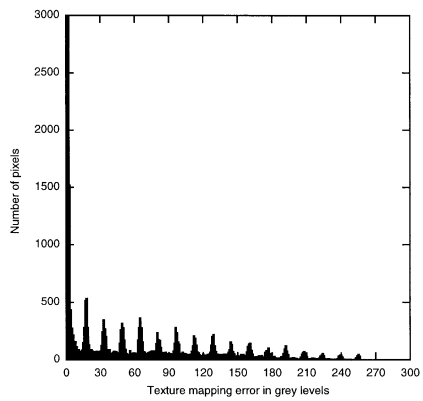
This section explains and illustrates the characteristics of the texture-mapping algorithm. An experimental protocol has been designed to assess objectively the quality of the texture-mapped





9a 9b





10a 10b

Fig. 9. Texture-mapped image of the telephone for the chessboard image (512×512 texture map): **a** texture-mapped image; **b** histogram of the texture-mapping error

Fig. 10. Texture-mapped image of the telephone for the chessboard image (256×256 texture map); **a** texture-mapped image; **b** histogram of the texture mapping error

images. Due to the lack of space, we only present the main conclusions of our experimental work.

A range image is a matrix of z values that describes an explicit surface $z = f(x, y)$. Five 256×256 range images from the National Research Council of Canada (NRCC) database (Rioux 1984; Rioux and Cournoyer 1988) were used in our experiments. These 3D images provided a sample of complex surfaces: a space shuttle replica, a hand, a toy soldier, a teapot, and a telephone. Actual object sizes varied from about 15 to 20 cm. All range images were triangulated, and these high-resolution triangulations were compressed by specifying 3D tolerances of 1.0, 2.0, and 4.0 mm. The compression levels achieved by the multiresolution algorithm were typically larger than 95%. Following the compression, all original range image points were mapped onto the compressed models. To measure quantitatively the fidelity of the texture mapped images, two 256×256 synthetic test images were mapped orthogonally on the range images in place of the actual surface colors. The first test image was a chessboard image made of 8×8 pixel, black and white squares. The second test image consisted of a ramp image in the red and green channels. The red component varied from 0 to 255 in the x-axis, while the green component varied from 0 to 255 in

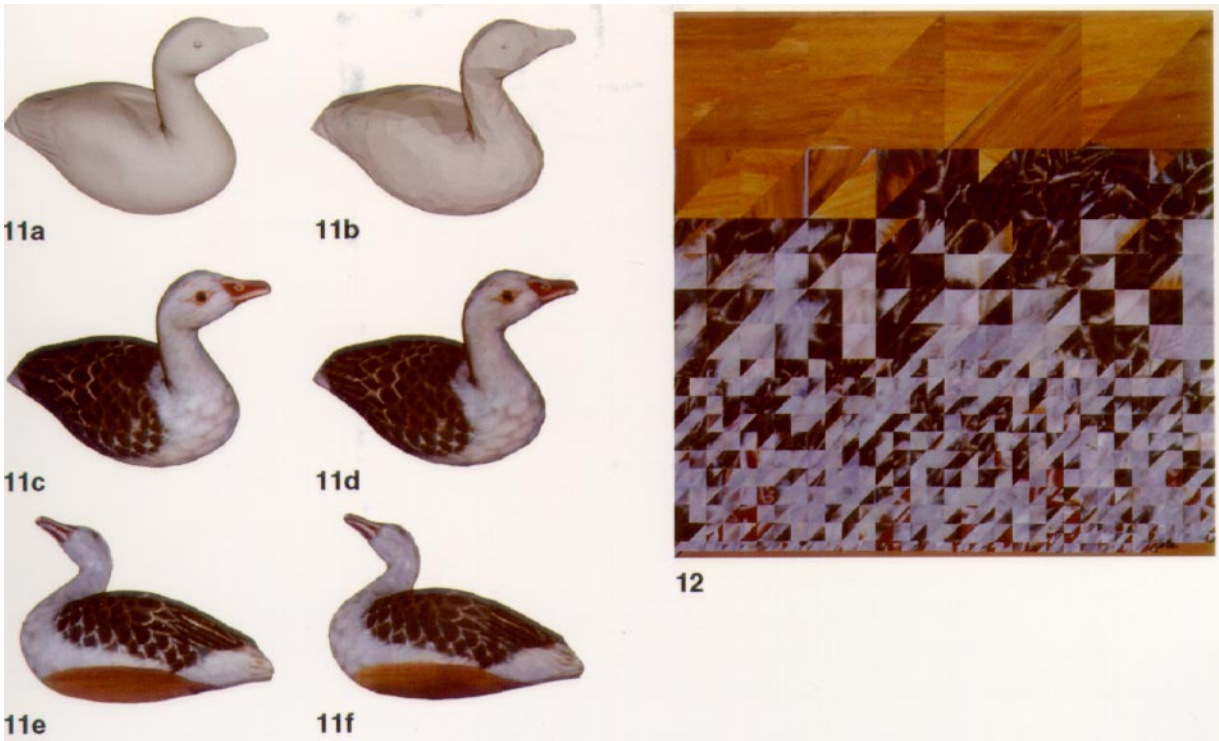


Fig. 11a–f. Results of applying the algorithm to the digitized model of wood duck. The original (**a**) and compressed (**b**) models are shaded with a uniform gray surface color. The high-resolution model is displayed with the color associated with each vertex (**c**). The compressed texture-mapped model (**d**) is rendered with smooth shading. Since they are 3D models, the original (**e**) and compressed models (**f**) can be viewed from a different angle

Fig. 12. The 512×512 texture map generated by the algorithm for the compressed model of the duck

the y-axis. The chessboard image was useful for evaluating the performance of the texture-mapping algorithm near color discontinuities. The ramp image allowed a measure of the quality of color representation when the color varied smoothly on the surface. All experiments were performed on a Silicon Graphics Crimson workstation with Reality Engine graphics. The triangles were rendered with the standard texture mapping functions of the IRIS GL library.

In two sets of experiments, we generated texture-mapped models from the 15 compressed models, using both the chessboard and ramp images, and allocating 512×512 texture maps. The texture-mapping error was measured by a pixel-to-pixel comparison between the original test pattern and the rendered image of the texture-mapped model. We used the same projection that would map the full resolution model back to the 256×256 original range image. We computed the mean square error introduced by our texture-mapping algorithm, as well as histograms of the texture-mapping error. When the ramp image was used, we found that the texture-mapping error was very small. This result demonstrated that our method is efficient for representing surfaces on which the color varies smoothly. In practice, we observed that such small errors could not be detected by the human eye. From the results obtained with the chessboard image, we observed that the texture-mapping error mainly occurs near color discontinuities on the surface.

In a third experiment, two texture-map sizes were used to represent the chessboard image mapped onto the telephone model having a 1.0 mm error. In Figs. 9 and 10, the allocated texture maps have the dimensions 512×512 and 256×256, respectively. The resulting texture-mapped image is depicted in Figs. 9a and 10a, while the histogram of the error is shown in Figs. 9b and 10b. One can

observe how the texture is filtered as the size of the texture map is reduced from Fig. 9 to Fig. 10.

A high-resolution, colored, 3D triangulation was used for the last experiment. Twelve color-range images of a wooden duck were first acquired with the NRCC color-range sensor (Baribeau et al. 1992), then integrated with the method described by Soucy and Laurendeau (1995b). The resulting high-resolution model is composed of 166 278 triangles and 83 141 vertices. Figure 11 shows the results of applying the compression and texture-map generation methods to this model. The compression algorithm is used to create a compressed model composed of 1000 triangles, corresponding to compression rate of 99.4%. The original high-resolution model is shown in Fig. 11a, and its compressed version in Fig. 11b: both models are rendered with a uniform gray surface color and flat shading on the triangles to enhance their geometric structure. The original colored model appears in Fig. 11c, whereas Fig. 11d shows the compressed model drawn with a 512×512 texture map computed by our algorithm. The compressed model is rendered with smooth shading over the triangles.

The visual impression given by the two rendered versions is very similar. Most of the noticeable differences are around the silhouette of the duck, since the high compression rate polygonizes the outline. The smooth shading method also introduces slight differences in the computed surface intensities, since finer surface-normal details such as on the duck's back are lost in the geometric compression. Figure 11e and f show the same model from another viewpoint, in high resolution and compressed versions, respectively. The duck model was closed and homeomorphic to a sphere, but the algorithm could have handled a different topology in the same manner since the triangles are processed independently.

Figure 12 shows the 512×512 texture map generated by the algorithm. The tessellation structure and hierarchical organization described previously are visible. It can be noted that the largest texture triangles (in the top rows of the map) all represent the natural wood color of the bottom of the duck (as seen in Fig. 11e and f), since this flat area produces the largest triangles in the compressed model. Conversely, the small triangles located on the red beak, a region of high curvature, produce small texture triangles that are found mostly at the bottom of the texture map. This behavior illustrates the ability of the proposed algorithm to maintain a texture resolution adapted to the structure of the original model.

6 Conclusion

A growing number of applications requires the real-time display of colored 3D objects represented by a triangulation with color information attributed to each vertex. Such models arise from the integration of multiple color-range images as described here, but also, for example, in geographic information systems. To obtain a high refresh rate while maintaining an acceptable level of realism in simulation systems, it is necessary to reduce the number of polygons describing these objects while preserving the visual impact by maintaining a high resolution for the color information. Towards such a goal, this paper described a texture-mapping approach that allows the efficient compression and representation of colored 3D triangulations. This algorithm can deal with any triangulation topology and allows the compression of the color itself, resulting in antialiased images. The performance of the algorithm is assessed through a series of experiments. One of the experimental results has shown that a 99.4% compression of a model can be achieved while a highly realistic color image of this model is maintained. Therefore, the proposed algorithm is expected to be applicable to virtual reality systems requiring fast and realistic imaging of complex colored 3D objects.

Acknowledgements. The authors thank L. Cournoyer and R. Baribeau for acquiring and preprocessing the color range images, as well as A. Rivard for useful discussions.

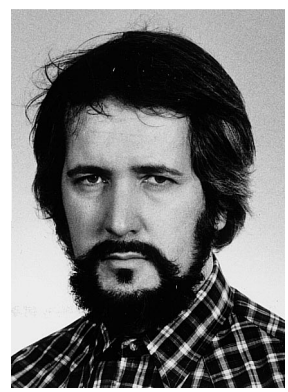
References

1. Baribeau R, Rioux M, Godin G (1992) Colour reflectance modelling using a polychromatic laser range sensor. IEEE Trans Patt Anal Machine Intell 14:263–269
2. Besl PJ (1988) Active, optical range imaging sensors. Machine Vision Appl 1:127–152
3. Borgefors G (1984) Distance transformations in arbitrary dimensions. Comput Vision Graph Image Processing 27:321–345
4. Danielsson PE (1980) Euclidean distance mapping. Comput Graph Image Processing 14:227–248

5. De Floriani L (1989) A pyramidal data structure for triangle-based surface description. IEEE Comput Graph Appl 9: 67–78
6. Haeberli P, Segal M (1993) Texture mapping as a fundamental drawing primitive. Proceedings of the 4th Eurographics Workshop on Rendering, Paris, pp 259–266
7. Heckbert PS (1986) Survey of texture mapping. IEEE Comput Graph Appl, vol. 6:56–67
8. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1993) Mesh optimization. Comput Graph SIGGGRAPH '93 Proceedings 19–26
9. Leymarie F, Levine MD (1992) Fast raster scan distance propagation on the discrete rectangular lattice. CVGIP: Image Understanding 55:84–94
10. Rioux M (1984) Laser range finder based on synchronized scanners. Applied Optics 23:3837–3844
11. Rioux M, Cournoyer L (1988) The NRCC three-dimensional image data files. Technical Report, 29077 National Research Council of Canada, Ottawa
12. Schmitt F, Chen X, Du WH, Sair F (1991) Adaptive G1 approximation of range data using triangular patches. In: Laurent PJ, Le Méhauté A, Schumaker LL (eds) Curves and surfaces. Academic Press, San Diego, pp 433–436
13. Schroeder W, Zarge J, Lorensen W (1992) Decimation of triangle meshes. Comput Graph (SIGGRAPH '92 Proceedings) 26:65–70
14. Soucy M, Laurendeau D (1995a) A dynamic integration algorithm to model surfaces from multiple range views. Machine Vision Appl 8:53–62
15. Soucy M, Laurendeau D (1995b) A general surface approach to the integration of a set of range views. IEEE Trans Patt Anal Machine Intell 17:344–358
16. Soucy M, Laurendeau D (1966) Multi-resolution surface modeling based on hierarchical triangulation. Comput Vision Image Understanding 63:1–14

MARC SOUCY received his BSc and PhD degrees in Electrical Engineering from Laval University, Quebec, Canada, in 1988 and 1992, respectively. From 1992 to 1994, he worked at the Visual Information Technology Group of the National Research Council of Canada, Ottawa, Canada. In April of 1994, he cofounded InnovMetric Software, Quebec, Canada, where he is currently developing software products to build color polygonal models with color 3D digitizer data. InnovMetric's URL is http://www.innovmetric.com.

GUY GODIN received his BIng degree in Electrical Engineering from the Ecole Polytechnique de Montreal in 1984 and his MEng degree, also in Electrical Engineering, from McGill University in 1989. From 1984 to 1985, he worked as a Research Associate in the Computer-Aided Design Laboratory at the Ecole Polytechnique. Since 1989, he has been working in the Visual Information Technology Group of the National Research Council of Canada. His interests include 3D computer vision and image processing, computer-assisted geometric design, and computer graphics.



MARC RIOUX received his BSc degree in Engineering Physics in 1971 and his MS degree in Physics in 1976, both from Laval University, Quebec. He worked for 5 years on TEA CO_2 laser development and applications, and for 2 years in infrared holography, before joining the National Research Council of Canada in 1978 to work on optical sensor research. His present interests are in 3D optical sensing for machine vision, media applications, and dimensional inspection.