Abstract—Three-dimensional (3D) urban models, due to their huge data sizes, pose a great challenge for streaming and rendering on devices with low bit rates and smaller computational powers. In typical 3D maps, which mainly have texture and geometry information, the texture comprise a large part of the data, say over 80%. Simplification of texture has an important role in reduction on the data sizes. In this paper, we present a numerical technique for simplification of the texture of these models, while preserving their major features. In the simplification process, we first select textures that can be segmented by straight lines using Hough transform from texture database. The selected texture images are simplified by the Hough transform. The other texture are filtered by the anisotropic diffusion and then the details are removed to reduce their data sizes. The representative colors of the image are chosen from the image, and the image is quantized based on those colors. The simplified image, with all major features of the original image preserved, has much smaller size as compared to the original. The removal of the unnecessary details from the original image makes it easy to the human eye to perceive the major features, and identify the objects of interest in bulk of huge 3D models.

Keywords: Color Quantization, Image Segmentation, Urban Model Simplification

1. INTRODUCTION

Recently there has been a greater interest in virtual urban environments, which have a large number of important applications including urban planning and management, design review, environmental studies, virtual tourism, navigation guides and several civilian and military applications etc. These virtual environments can be generated from a variety of data sources, including aerial images, ground based laser scanners, CAD (Computer Aided Design) models, and the GIS (Geographic Information Systems) data [1].

The data size for 3D urban models is very large and throws up a challenging problem for visualization. With the faster machines becoming available, it is possible to visualize a virtual environment of a smaller size, say a power plant or a museum, on dedicated terminals. However, the problem is more challenging for applications like walking through or flying over a large virtual city. The research carried out to realize efficient interactive visualization systems could be divided into two categories; a) the organization of data at different levels of details [2], and b) enhancing the rendering performance [3]. The basic idea behind each of these techniques is reduction in the complexity of the overall model before rendering.

Along with geometry, texture is an important part of 3D models, and reduction in its size is as important as simplification of the geometry for a meaningful reduction in overall size of the model. Moreover, on devices with smaller screens, like cell phones, it is desirable to display the models with minimum details of the texture. This makes it easily understandable and identifiable among a large number of objects. For example, a seven-eleven convenient store should be represented by the information that is just enough to make it distinguishable among other buildings. Providing more details, would not only make the data size larger, but will also make it unsuitable for certain applications. For example, a user can find it hard to follow a 3D interactive map on the small screen of the cell phone, if the minor details are displayed.

In a normal 24-bit RGB image, there are more than 16 million colors, however most of images can be represented by much smaller number of colors, without any degradation of the quality detectable to the human eye. For urban 3D models, where each scene has several texture images, this 24 bit representation results in data, too heavy to be downloaded and rendered on low bit rate and low computational power devices like cell phones. Color reduction, which is already being widely used for transmission, segmentation, and compression of color images, comes as an obvious choice for simplification of texture in 3D urban models. In this research, we adopt the color reduction concept for our application, taking
advantage of some special characteristics of the common texture of the urban models.

The simplest technique for color reduction is color quantization [4-13]. In this technique, similar colors are grouped together and then all the pixels of the image belonging to a certain group are assigned a single representative color. Selection of the true representative colors is very important, because a color that does not truly represent the group may result in considerable difference from the original image. Several methods for selection of the representative colors like those based on median and minimum variance etc. can be found in literature [5-7]. Some other techniques, based on cluster analysis, have also been presented for color quantization [8-13]. Another technique widely used for simplification of gray scale images is multithresholding [13-15]. This technique uses image histograms to select different gray value classes and uses the centers of these classes as representative gray values. Extension of this technique to color images is possible by using histograms for each color component. However, this technique does not give satisfactory results when object and background cannot be selected from the color information only [16]. A more effective adaptive color reduction technique using neural network has been presented in [17].

A large part of urban 3D models consists of buildings, and the texture images associated with these models contain the information of the walls, doors, and the windows etc. In other words, the regions with different colors are separated by straight-line segments in these images. Taking advantage of this fact, such images can be separated easily from rest of the complicated texture images in the urban 3D models, and can be represented with very small number of colors. For buildings, we are not very much concerned about the details of the texture; rather we want the texture to be with minimum details in order to enhance its major features and make it easily recognizable by a human eye. This gives us much freedom to reduce the total number of colors, and achieve considerably smaller size as compared to the original image.

2. THE PROCEDURE

In a typical 3D urban model, texture is represented by images represented in JPEG or some other suitable format. Based on certain characteristics of the texture files associated to them, the objects in urban models can be divided into three classes, as given below:

a) Road signs are the objects hanging over the roads offering directions and distances to the nearby places. The image files associated with these objects have only two colors, i.e., white text and arrows on blue or green background, depending on whether the road sign is hanging over a normal or a toll road, respectively. The road signs can be identified from the histogram of the image file, by checking if the peaks lie on a predefined positions or not.

b) Most of the objects in the urban models are buildings, and the texture files associated to these objects contain information of the walls, streets, doors, and windows.

c) Rest of the images, which do not belong to any of the above classes.

The images belonging to the first category can be reduced to binary images and this results in considerable reduction in their size, while the objects belonging to the last category can be simplified using the available color quantization techniques. For the images belonging to the second category, the color quantization can be done very effectively by taking advantage of the fact that the objects in these images are defined by straight-line segments, and this can lead to considerable reduction in their size. The method for identification and simplification of these objects is described in detail in the forthcoming discussion.

A. Creating the Gradient Image

Let I(i, j), where i and j indicate the row and column numbers respectively, be an RGB color image. A simple way to get the gradient of this color image is getting three gray scale images comprising of R, G, and B values respectively, and then combining their corresponding gradient images. Gradient image of a gray scale image can be calculated easily using one of the many available
methods. In this research, we use a 3 x 3 Sobel filter to calculate the gradient images of the gray scale images. Sobel operator is computationally efficient and relatively less sensitive to additive noise because of the averaging of the pixels in the neighborhood.

Consider a typical image shown in Fig. 1a. Its gradient obtained by using the method described above is shown in Fig. 1b. Here, it should be noted that using some more sophisticated filters like Canny filter, or using RGB components together instead of separating the color image into three sub images [18], could produce relatively better gradient images. However, for our application, which involves urban images, a simple Sobel filter gives quite satisfactory results for implementation of our algorithm.

B. The Hough Transform

The Hough Transform is a standard tool in image processing, for detecting certain patterns in images. As an example consider a straight line expressed in parameterized form,

\[ x \cos \theta_0 + y \sin \theta_0 = \rho_0, \]

where \( \rho_0 \) is the perpendicular distance from the line to the origin, and \( \theta_0 \) is the angle between \( x \)-axis and the normal to the line. Now each point \((x_i, y_i)\) on the line, corresponds to a sinusoidal curve in \((\rho, \theta)\) parameter space, all of which intersect at \((\rho_0, \theta_0)\). This gives us a simple way of detecting straight lines hidden in large amount of data. First, we transform all the points in the gradient image to \((\rho, \theta)\) parameter space, and locate the points where large numbers of sinusoidal curves intersect. These points correspond to the possible lines in the image space.

We use Hough transform to transform all points in the gradient image to the \((\rho, \theta)\) parameter space, and use a suitable threshold value to select strong peaks, corresponding to lines in the image space. Due to a sinusoidal curve for each point in the gradient image, several fake peaks arise in the vicinity of the true peaks, and make it difficult to detect relatively smaller true peaks. In our algorithm, we update the \((\rho, \theta)\) parameter space after detection of each line, by removing the curves corresponding to the points lying on the detected line; in order to we do away with the effect of fake peaks.

At this stage, we compare the gradient image with the detected lines. If most of the points (above a predefined threshold) lie over these lines, the image describes the texture of a building and can be simplified by using the procedure described in the forthcoming discussion. If this is not the case, then we stop the procedure here and use one of the available color quantization techniques for simplification.

The detected lines are divided into smaller segments by finding their intersections and a decision is made for each segment to keep or remove it, by comparing it to the actual gradient image. The line segments, kept after the process, give the final gradient image, and divide the image into distinct regions. The gradient image consisting of the line segments obtained from the gradient image of Fig. 1b is shown in Fig. 1c. It should be noted that most of the information in the original image of Fig. 1a lies in the pixels under these lines.

C. Image Segmentation

Removal of certain line segments that do not match with the actual gradient image, as described above, leaves some other line segments unconnected. We sort the removed line segments based on their length, and then start putting them back in the gradient image, the shortest one processed first, till both ends of each lines segment are connected either to some other line segment or to the boundary of the image.

Similarly, there are certain very small line segments, like those created when a line intersects two very close parallel lines. Due to their short length, these segments sometimes get to the final gradient image, even if there are few matching pixels for them in the original gradient image. These line segments unnecessarily divide the image in a large number of regions. We remove all such segments, based on the criteria that their removal should not leave any unconnected line segment. Again, the shortest segment is processed first.

The process divides the image into separate regions (segments) based on their color. The segments are closed regions, which is a necessary condition to perform the color reduction process as discussed in the next subsections.

D. Selection of Representative Colors

The straightforward method for selection of representative colors is to get the image histogram, and then select the peaks, i.e., the colors with large number of pixels. This method, however, has two major problems. First, two very similar colors can be selected as the representative colors, if number of their corresponding pixels is large. Secondly, there might be some important features those we would like to have in the final image, but their representative colors might not be chosen due to smaller numbers of corresponding pixels. With the segmentation already done in the previous step, we can solve the second problem easily, by selecting one color from each region. However, the first problem remains, as two major segments might have representative colors very close to each other. This makes compression of the final image inefficient, and therefore results in a relatively larger size. We use the following algorithm for selection of the representative colors, which successfully handles both of the problems stated above:

1. Select one representative color from each region.
2. Find closest match to each selected color among other selected colors, based on the minimum Euclidian distance. The Euclidian distance of a color \((R, G, B)\) to another color \((r, g, b)\) is given by
\[
d = \sqrt{(R - r)^2 + (G - g)^2 + (B - b)^2}
\]
3. If the minimum distance between two closest colors is below a certain threshold, replace the color with smaller number of pixels by the other with larger number of pixels.
4. Recalculate the closest matches to each color among the remaining colors, for which the color removed in step 3 was the closest match.
5. Repeat 3 to 4 until the minimum distance between any two colors is below the predefined threshold.

In most of the images, the boundary between two regions (pixels under the line segments in gradient image) has a color identical to the color of one of the two neighboring regions. However, in some cases the boundary has a color different from the colors of both neighboring regions. To handle such situations, we include the representative colors for each boundary segment also in the initial list of colors in the step 1 of the above algorithm. However, when the algorithm is implemented, only the colors that are considerably different from the colors of the regions, will get included in the final list of representative colors.

E. Creating new image

After selection of the representative colors, the color of each region and the boundary in the segmented image is replaced by their representative colors. The number of colors of the image shown in Fig. 1a is reduced to four using the procedure described above, and the resultant image is shown in Fig. 1d. It can be noted that the reduced image is very close to the original image and all of the important features of the actual image are preserved. The size of the original image was 6.74 kilobytes (KB) in JPEG format, which has been reduced to 1.24 KB.

F. Representing image in Colors-of-Regions-and-Edges (CRE) format

The new images created in above subsection E can be represented in a very compact format, instead of common JPEG or other formats. We call this new format as Colors-of-Regions-and-Edges (CRE) format. This new format is a text file containing the following information:

1. The size of the image
2. The representative colors
3. The start and end points of the edges and the index numbers of the representative colors
4. The index numbers of the edges making a region and the index number of the associated color

The three examples shown in the previous subsection represented in CRE format are reduced to 896 bytes, 740 bytes, and 1.86 KB respectively. These sizes can be further reduced to almost half by converting the CRE text files to binary format.

G. Texture Simplification by Anisotropic Diffusion

Although the method mentioned above realizes an efficient simplification, it has a limitation, the images are segmented by straight lines. In our 3D maps, for example, 45 to 55 % of textures can be simplified by the above method. In our framework, the rest of the textures are smoothed by the Anisotropic Diffusion, and then their colors are reduced by the quantization.

The Anisotropic Diffusion is a time dependant edge-preserving smoothing filter [19]. The intensity of images are diffused, depending on partial differential equations (PDEs) of the same images. By iteratively applying the diffusion controlled by the PDEs, the images are converged to piecewise constant approximations. Let intensity of an input image \(I(x, y)\). An n-th iteration of the Anisotropic Diffusion is:
\[
I_n = \frac{\partial}{\partial x} \{\alpha(x, y, n)I_x\} + \frac{\partial}{\partial y} \{\alpha(x, y, n)I_y\}
\]
where \(I_x = \frac{\partial}{\partial x} I(x, y, n), \ I_y = \frac{\partial}{\partial y} I(x, y, n), \ \alpha(x, y, n)=\frac{1}{1+\frac{I_x^2+I_y^2}{K^2}}, \) and \(K\) is a constant conductance factor.

After the isotropic diffusion is applied, the color is reduced by the color quantization to reduce the size of the textures.

4. EXPERIMENTAL RESULTS

I. Hough Transform:

Two more examples have been shown in Figs. 2-3. The image shown in Fig. 2a, originally having a size of 6.11 KB in JPEG format has been reduced to a 1.36 KB image shown in Fig. 2b. To show the details lost in the reduced images, zoomed views of both of the images are shown in the figure. It can be seen that while the major features of the actual image are preserved, the details have been lost in the second segment from top, and this is quite acceptable and even desirable at times, in our application of rendering urban 3D models on cell phones. As discussed earlier, our goal is to remove the unnecessary details in the image, in order to make it smaller and more
easily perceptible to a human eye. Figure 3 shows yet another example, where an image of 11.7 KB has been reduced to 2.7 KB.

II. Anisotropic Diffusion

Fig. 4 shows the results of the simplification by the Anisotropic Diffusion. The size of original images compressed by JPEG, 6.71KBytes are reduced to 3.46Kbytes by removing unimportant edges.

4. CONCLUSIONS

The data content of 3D urban models, including geometry and the texture mainly, is very high and needs to be reduced for efficient streaming and rendering on low bit rate and computationally less powerful devices like cell phones. In this research, a simple method for simplification of the texture in these models has been proposed. The simplification method, based on color quantization, preserves the major features and removes the unnecessary details, to make the image smaller in size and more easily perceptible to the human eye.

ACKNOWLEDGEMENTS

We wish to thank ZENRIN CO., LTD and Geo Technical Laboratory Co., Ltd. for providing 3D digital maps. This work was partly supported by Grant-in-Aid for Young Sciences (#14750305) of Japan Society for the Promotion of Science, fund from MEXT via Kitakyushu innovative cluster project, and Kitakyushu IT Open Laboratory of National Institute of Information and Communications Technology (NiCT).

4. REFERENCES

Fig. 2: (a) The JPEG test image of size 6.11 KB (b) The JPEG reduced image of size 1.36 KB.

Fig. 3: (a) The JPEG test image of size 11.7 KB (b) The JPEG reduced image of size 2.7 KB.

(a) Original JPEG Image of size 6.71 Kbytes
(b) Simplified JPEG Image of size 3.46Kbytes
Fig.4 Results of Anisotropic Diffusion