Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2009, Article ID 432853, 8 pages
doi:10.1155/2009/432853

Research Article

Texture Classification for 3D Urban Map

Hiroyuki Inatsuka, Makoto Uchino, Satoshi Ueno, and Masahiro Okuda

Department of Information and Media Sciences, Faculty of Environmental Engineering, The University of Kitakyushu, 1-1 Hibikino, Wakamatu Kitakyushu, Fukuoka 808-0135, Japan

Correspondence should be addressed to Masahiro Okuda, okuda-m@env.kitakyu-u.ac.jp

Received 27 February 2008; Revised 15 December 2008; Accepted 23 February 2009

Recommended by Stefano Tubaro

This paper proposes a method to control texture resolution for rendering large-scale 3D urban maps. Since on the 3D maps texture data generally tend to be far larger than geometry information such as vertices and triangles, it is more effective to reduce the texture by exploiting the LOD (Level of Detail) in order to decrease whole data size. For this purpose, we propose a new method to control the resolution of the texture. In our method we classify the textures to four classes based on their salient features. The appropriate texture resolutions are decided based on the classification resulsts, their rendered sizes on a display, and their level of importance. We verify the validity of our texture classification algorithm by applying it to the large-scale 3D urban map rendering.

1. Introduction

Three-dimensional urban maps (3D map) have a variety of applications such as navigation systems and disaster management and simulations of urban planning. As the technology that acquires 3D range data and models of cities from the real world has advanced [1–4], more sophisticated visualization of photo-realistic 3D map is becoming available.

In general the 3D map has geometry information of 3D meshes and texture images. Since the amount of the data is huge, its data size often becomes a problem when it is treated in devices including PCs, car-navigation systems, and portable devices. Therefore, the reduction in the volume of data becomes important to make the 3D map applications user friendly.

Many methods on the LOD control of general 3D models have been proposed so far [5–7]. Moreover there exist some visualization techniques [7–12] of terrain data that express the surfaces of a ground with geographical features like mountain district. While these conventional methods assume that a single model is locally smooth and consists of topological manifold that contains a large number of meshes, the city model consists chiefly of buildings. Since the data in the 3D map mostly consist of many small, already simplified meshes (like buildings made of some cuboids), these simplification methods cannot be applied to it. Although some methods on modeling and rendering of the 3D maps have been already proposed [13–15], there exist few methods concerning the reduction of the texture data for the 3D map. In the 3D map, the proportion of the total amount of texture data tends to be much higher than that of geometrical information such as vertices and triangles. For example, in the 3D map we use for a simulation (Figure 10) the whole size of the textures is 461 MBytes in JPEG format, while the geometry information has only 15 MBytes in gzipped VRML. It means that the LOD of the texture has a capability of effectively reducing the whole data size rather than the LOD of the geometry. For this purpose, we propose the technique for controlling the resolution of the texture. Wang et al. [16] propose a method to detect the repetition of content in the textures to reduce storage and required memory. In this paper we focus on urban map rendering from a ground level (as shown in Figure 11) rather than bird's eye view. These systems can be used in car-navigation and human-navigation systems.

In the following sections, we describe the overview of our rendering system with consideration of the level of texture importance. In Section 3, we propose our classification method based on K-Nearest Neighbor approach. Then we

show some experimental results to verify the validity of our classification method in Section 4. In Section 5 we summarize the strength of our method.

2. Proposed Method

2.1. Outline. Our 3D map is composed of simplified polygonal meshes (as shown in Figure 3) and textures mapped on the meshes. The textures are made of photographs of real scenes.

In order to reduce rendering cost, it is an efficient stategy that only visible data from a user's viewpoint are loaded and rendered. To implement this, we introduce a representative viewpoint (RV), which is a discrete point in the 3D map used for a reference point of rendering. In our rendering system, we spread RVs all over the map in advance. Using the Z-buffer algorithm, visible objects from each RV are determined. When rendering, we first find the RV closest to a current viewpoint of a user. The objects seen from the RV on the map are found, and then only the visible objects from it are rendered.

To further reuduce the rendering cost, we introduce a method to determine an appropriate resolution for texture. In the framework of our rendering system, first of all, a set of images at multiple resolutions is prepared for all the texture images beforehand. Our strategy is that *an appropriate resolution* is found for each texture, and the texture at the resolution is loaded. The appropriate resolution is determined for each texture image by two criteria "Level" and "Class". The former considers a look from a specific viewpoint, and the latter evaluates importance of the texture, which we will explain in Section 3. The entire algorithm is depicted in Figure 1.

Note that in our setting, different types of objects are separately saved to different image files, for example, no image includes a billboard and a window simultaneously. Since our classification is done for images directly, the algorithm would fail for images that simultaneously contain objects to be classified to different classes.

2.2. Selection of Representative Viewpoints. In this step we select the representative viewpoints (RVs) on the 3D map. In our framework, we suppose that a user walks on a street of the 3D map. Only rendering from a ground level is considered. Although to make the points equally spaced in the map, one has to define them based on the geometrical form of terrain data, we adopt the following simple procedure to reduce its computational complexity, as we assume the terrain associated with the map is flat.

To select the RVs, the entire map is divided into rectangle areas, and then we set the points on the grids as candidates for the RVs. Figure 2 shows an example of the RVs (note that in actual cases the distance between the lattice points is shorter, and the points are more densely located).

Next, the points in the area thought to be paths when the user actually walks through in the 3D map are selected as the RVs from the candidates on the lattice points in Figure 2.

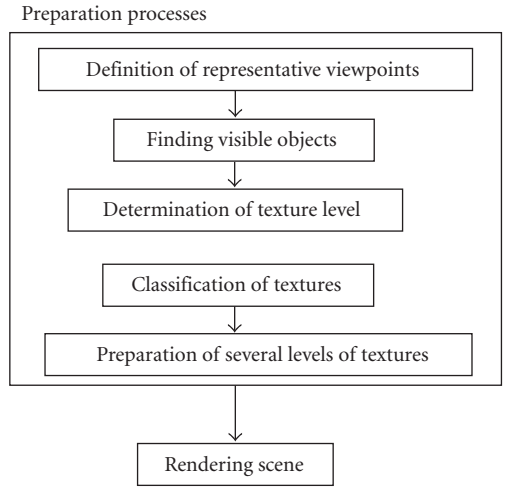

Figure 1: Outline of our method.

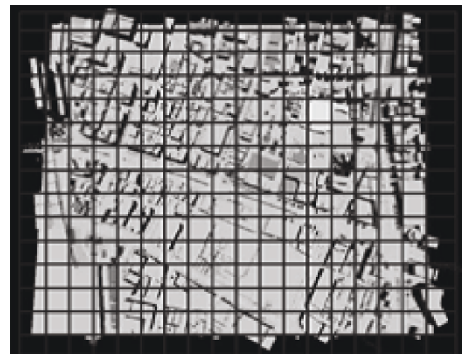

Figure 2: Candidates for representative viewpoints.

This is done simply by excluding the candidate points in high places such as the tops of the buildings.

2.3. Determination of Visible Objects. A set of the objects that are visible from each of the RVs are determined by using the Z-buffer algorithm [17]. The accuracy of the determination depends on the resolution of the Z-buffer. Increasing the resolution, both of the accuracy and the computational cost increase, and vice versa. We have determined the resolution of the buffer by trial and error. Note that one needs to do this processing to a 3D map only once as preprocessing. For example, Figure 3 depicts the objects judged as "visible" from the RV denoted by the sphere in the middle. The determination is done for all the RVs, and we store the indices of the objects in a list.

2.4. Texture Resolution. In general there is a trade-off between the quality of the texture and its computational cost to load and render it. It is necessary to find an appropriate resolution of the texture that maintains an adequate image quality and low computational cost in order to enhance the efficiency in rendering the 3D map. The determination of the resolution is done for all the textures on the visible objects from each RV. Our previous work [18] defines the "Level" of

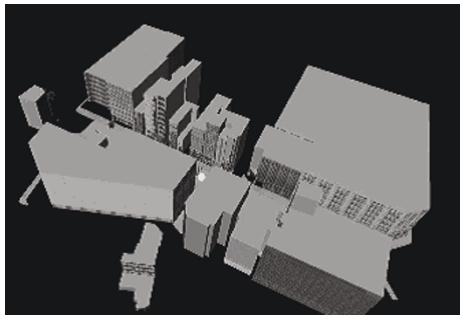

Figure 3: Objects visible from a viewpoint.

the importance for each texture, and then based on the level we select an appropriate resolution for the texture among several resolutions prepared in advance. We determine the level by two criteria: (1) the size of the texture on a display and (2) the "look" of the texture. The size of the rendered texture is decided by the following three elements.

(i) The distance d between the RV and the surface where the textures are mapped.

(ii) The area S of the texture in the 3D map.

(iii) The number of repeats r.

The distance d can be found by the barycentric coordinates of the surface where the texture is mapped and the xyz coordinates of the RV. When the surface is slant to the RV, the area of the texture is narrower than the one seen from the front. To take this into account, we first consider a virtual plane P that is perpendicular to the line connecting center of the plane and the RV. The area S in the new surface made by projecting each vertex of the surface, where the texture is mapped, to the plane P is calculated. Sometimes one texture is repeatedly mapped on a large surface. The number of the repeats r can be found by the texture coordinate (s, t). Finally the size of the texture is estimated by using d, S, and r.

The "look" of the texture is evaluated by (1) the ratio of sharp edges in the texture and (2) the number of colors used in the texture. As for texture with sharp edges, one can easily identify what it is due to its recognizable feature in a real scene and is often used as markers in some applications such as a navigation system. Thus in our framework, we consider that such a texture has high level of importance. Conversely, for a texture without sharp edges the level of importance is considered low and low resolution is enough to express such texture. To distinguish these types of images, we first introduce the ratio of sharp edges. To actually evaluate the edges, first we calculate the intensity Y from RGB values:

$$Y = 0.299R + 0.587G + 0.114B. \tag{1}$$

Then, the edges are detected by applying the Laplacian filter to Y. Two thresholds t_l and t_s ($t_s < t_l$) are prepared, and the number of pixels whose absolute values exceed each of the thresholds is counted. Denoting, the numbers of pixels that exceed t_l and t_s as $c(t_l)$ and $c(t_s)$ respectively, the ratio of the sharp edges R_e is given by

$$R_e = \frac{c(t_l)}{c(t_s)}. \tag{2}$$

The threshold t_l is used for saliency detection and t_s control a weight on images with smooth background. Of course the two thresholds affect the performance of the algorithm, for example, the smaller t_s put more weights on images with flat background. However the thresholds do not sensitively affect a final result. We choose these thresholds by trial and error.

Next we consider the colors of the texture. In this method, the texture composed of the wide variety of colors is considered as "less important" texture, since in many applications including the navigation system, too complex information seldom plays an important role, and it is even unrecognizable from a distance. To evaluate the color complexity, we use the variance σ^2 of the RGB histograms. Note that as each bin of the histogram means its frequency, a low variance indicates that the texture contains a wide variety of colors and a small number of colors mean a high variance.

Finally we simply combine the measures

$$V = \frac{S}{d \cdot r} \cdot \min\{\beta R_e, \sigma^2\}, \tag{3}$$

where β is a normalization factor. We use this V for judging the level of importance of the texture.

3. Texture Classification

In the previous section, we define the value V that is obtained from the display size and the features of the texture. In our previous report [18], one of the multiple resolutions is assigned according to V of each texture, when actually rendering the 3D map. In the system of [18] , we prepare four levels of the textures, which are simply created by reducing the resolution of an original texture by 1/2, 1/4, 1/8. However with this method, salient texture such as road traffic signs and unremarkable texture like the walls of buildings are treated similarly. To address the problem, we introduce a new LOD control based on the texture classification. We make it possible to control the resolution more reasonably based on image features by classifying the texture into some classes, and then changing the reduction ratio based on the classes. By using this method it becomes possible to make the resolution of the image in one class smaller than the one in an other class even if it has larger V in (3).

3.1. Definition of Class. We first define some classes. Figure 4 shows some examples of the textures that are often used for walls of buildings. These textures mainly have soft edges, and their colors are apt to be saturated. As these types of textures have simple patterns, they rarely have remarkable meaning like letters and marks. Therefore a low resolution is often enough to maintain its visual quality in the 3D map.

On the other hand, the textures with sharp edges and vivid colors shown in Figure 5, unlike in Figure 4, have salient

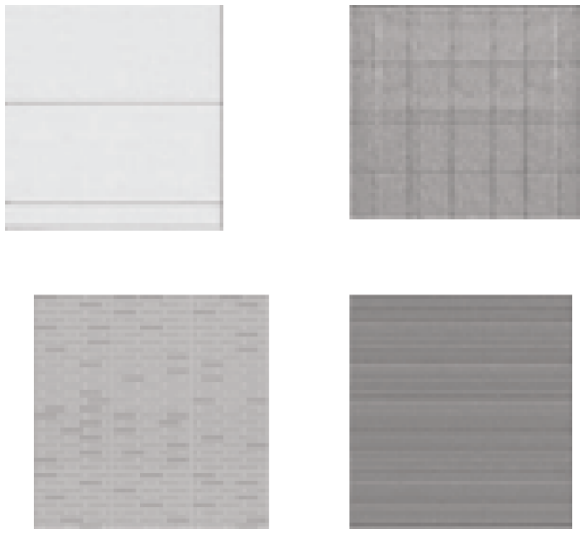

Figure 4: Walls with soft edges (Class 1).

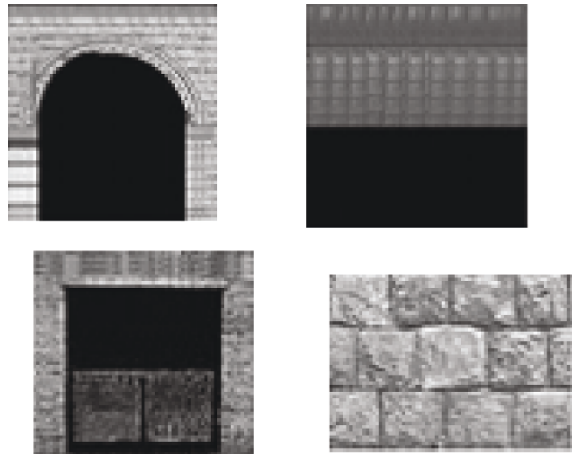

Figure 6: Walls with sharp edges (Class 3).

Figure 5: Billboards and directional signs (Class 2).

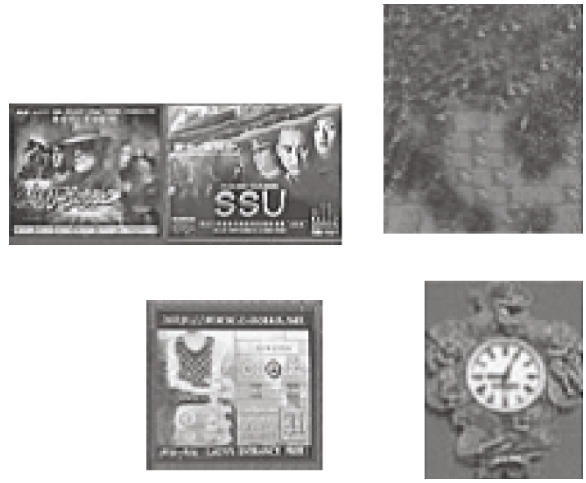

Figure 7: Detailed texture.

features, and often contain important information. They are often used as landmarks in the application of navigation system. Moreover, a human easily perceives blurring and ringing artifacts when the texture is rendered at a low resolution, which results in a decrease of visual quality. Thus a higher resolution should be allocated for this kind of the textures.

The texture that contains some sharp and soft edges simultaneously like in Figure 6 is located between those in Figures 4 and 5. In other words they are less simple than those in Figure 4 but also less significant than those in Figure 5. The low resolution display will somewhat decrease the quality of such textures. However it will not become a serious problem since those seldom contain significant features.

Some example of the fourth class is shown in Figure 7. The textures with minute and complex structures are not often used as markers and considered less important. And the visual quality of these images are not very much affected by reducing the resolution, since a masking effect of such complex images is higher than smooth images from a psychophysical point of view [19]. Moreover this kind of textures is less compressible due to its granularity, which is inefficient in terms of overall trade-off between data sizes and data visual quality.

In the end, we define the following four classes.

Class 1: walls with soft edges.

Class 2: texture with some clearly outlined objects with smooth background.

Class 3: walls with sharp edges.

Class 4: others.

In our framework, we select the resolution based on these classes besides the levels based on (3). We consider the case that four levels of textures are prepared, where the reduction ratios are 1, 1/2, 1/4, and 1/8, and all the textures are labeled as each of the Levels 1 to 4. Then we classify all of the original texture to Class 1 to 4. For the textures in Class 2, their original images are allocated to the Level 1 and 2 and the images shrunk to a half are allocated to other levels, while for Class 1 the images shrunk to 1/8 are used for all the levels. By this strategy, efficient LOD control of the texture is made possible than the case of only taking (3) into account.

3.2. Classification Method. The K-Nearest Neighbor (K-NN) algorithm is used to automatically classify a large amount of textures. In general the selection of feature vectors greatly affects the accuracy of the classification. Here we introduce two feature vectors based on colors and edges.

The first feature is color moments of color. The color moments have been widely used in conventional image retrieval systems [20, 21]. The first order moment (mean) roughly measures the appearance of images. Although it has been proved to be efficient in the retrieval system, in our application measuring the color difference does not improve this classification, that is, blue and red signboards should be equally treated. On the other hand, the second moment (variance) of the color measures the minuteness and has the capability of discriminating simplicity and complexity of the textures. The third (skewness) and forth (kurtosis) moments may also be applied, but these high-order moments are often sensitive to small changes and may degrade the performance. For the reasons stated above, we adopt the second-order moment for the feature of colors.

The choice of color space is also an important issue. The drawback of directly using the RGB pixel values is that the RGB color space lacks perceptual uniformity. Uniform color spaces such as CIE 1976 $L^*a^*b^*$ and $L^*u^*v^*$ have been successfully used in some retrieval methods. However to transform the RGB space to these uniform spaces, the information of the three primaries and reference white are needed, since they are device independent. Instead the HSV is used in our system. The HSV is suitable for our application since it is easily translated from RGB, and color saturation plays an important role to evaluate the saliency of images. In the end, we calculate the variances of H, S, and V channels with respect to all the pixels, and then the three variances are adopted as the first feature. As the second element for the feature vector, we use the energy of edges derived by the anisotropic diffusion filter [22]. The anisotropic diffusion filter is an iterative nonlinear lowpass operator that flattens smooth regions while keeping sharp edges. At each iteration, one multiplies the weights called "edge-stopping function" $w(\cdot)$ to the difference between a current pixel I_i and its neighbors, and then it is added to the current pixel value. The output at each iteration is expressed by

$$I_{i,j}^{k+1} = I_{i,j}^{k} + \frac{\lambda}{4} \sum_{\{m,n\} \in N_{i,j}} w\left(I_{m,n}^{k} - I_{i,j}^{k}\right) \cdot \left(I_{m,n}^{k} - I_{i,j}^{k}\right), \quad (4)$$

where k is an iteration number and $N_{i,j}$ is a neighborhood of (i, j). λ is a parameter that controls the smoothing effects. In [22], they introduce two functions for $w(\cdot)$. We adopt one of them:

$$w_{i,j} = e^{-\left(m_{i,j}/K\right)^2}, \quad (5)$$

where K is a parameter that controls the strength of the edges to remain, and $m_{i,j}$ is the intensity of the edge at site (i, j). When $m_{i,j} > K$, the weight $w_{i,j}$ becomes small, which reduces the affects of smoothing. Oppositely when $m_{i,j} < K$, $w_{i,j}$ approaches to 1 and it reduces the difference from its neighbors. Figure 8 illustrates the effect of the anisotropic diffusion. After the operation, we calculate the difference between an original image I and its smoothed version I^a, and then we adopt the energy of the difference as the second feature F_e, that is,

$$F_e = \sqrt{\frac{\sum_{i,j} \left| I_{i,j} - I_{i,j}^{a} \right|^2}{M}}, \quad (6)$$

where M is the number of pixels. The role of the feature is to take only sharp edges into consideration.

Figure 8: Anisotropic diffusion (left) before filtering and (right) after filtering.

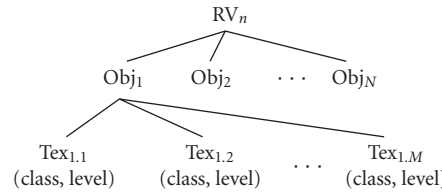

Figure 9: Hierarchical structure of RV list.

3.3. Rendering. By applying the pre-processing of Sections 2 and 3, all the objects that are visible from each RV are selected, and the classes and levels of all the texture for the RVs are determined. Then we construct the list of the RVs. In the list, each RV has the indices of all the visible objects and textures that belong to the objects. The class and level assigned to the texture are listed as in Figure 9.

When walking through the 3D map, a current position and its closest RV are found, and then loaded are all the textures at the appropriate resolutions that belong to the nine RVs, that is, closest RV and eight neighboring RVs. The reason why the neighboring RVs are loaded is that loading cost can be reduced and scenes can smoothly change when the user moves to the area of another RV.

4. Experimental Results

4.1. Precision of Classification. In our experiment, we use a 3D map with 5140 textures shown in Figure 10. For learning of the K-NN, 400 textures are used for the four classes, and we set $K = 30$. To evaluate the validity of our classification method, we manually classify the textures to the four classes randomly select 50 textures in each class, and then use them as ground truth. Then, we count the numbers

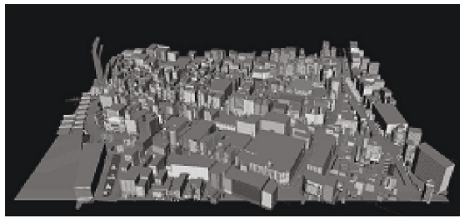

Figure 10: 3D map used for our experiment.

of correctly classified textures and calculate the precision of the classification by

$$P = \frac{\sum_i R_{C_i}}{\sum_i N_{C_i}}, \tag{7}$$

where C_i is the classes and R_{C_i} and N_{C_i} are the number of the textures that are correctly classified and are manually classified as the ground truth (in our experiment $N_{C_i} = 50$), respectively. The precision for each class is defined as

$$P_{C_i} = \frac{R_{C_i}}{N_{C_i}}. \tag{8}$$

First of all we verify the validity of our feature vector: the moment of the color and the edge energy based on the anisotropic diffusion. The feature vectors in our method are compared with features that are often used for general image retrieval [20, 21, 23–25]. Table 1 shows the comparison in color, as follows.

(i) Var. Hist.: the variance of the color histogram.

(ii) Hist(512): the color histogram quantized to 512 bins.

(iii) CCV: color Coherent Vector proposed in [23].

(iv) Moment: proposed feature vector.

In Table 2 we show the comparison in the features of edges, as follows.

(i) Sharp edges: the cost in (2) that we use to determine the texture level.

(ii) Wavelet coeffs.: quantized wavelet coefficients (high pass outputs of the dyadic wavelet).

(iii) Directionality: the quantized direction of the edges that is obtained by Sobel filter.

(iv) Anisotropic diffusion: proposed feature vector.

Among the feature vectors in colors, our method gives the highest score. For the edges, our method and the wavelet coefficients perform better than others. In Table 3, we show the precision of the combination of the two feature vectors. After experiments for every combination of the vectors in Tables 1 and 2, we have confirmed that our method outperforms others. Note that although it is possible to increase the dimension of vectors to add some features, we have seen little improvement with more vectors.

Table 1: Precision of features on colors.

	Class 1	Class 2	Class 3	Class 4	All
Var. Hist.	0.96	0.18	0.60	0.82	0.64
Hist(512)	0.46	0.54	0.12	0.98	0.53
CCV	1.00	0.36	0.42	0.84	0.66
Moment	0.90	0.66	0.66	0.74	0.74

Table 2: Precision of features on edges.

	Class 1	Class 2	Class 3	Class 4	All
Sharp edges	0.92	0.62	0.38	0.52	0.61
Wavelet coeffs	0.92	0.54	0.60	0.76	0.71
Directionality	0.88	0.44	0.34	0.74	0.60
Anisotropic diffusion	0.88	0.66	0.38	0.86	0.70

Table 3: Precision of two features.

	Class 1	Class 2	Class 3	Class 4	All
Moment + Wavelet	0.96	0.76	0.68	0.84	0.81
Hist(512) + Wavelet	0.96	0.40	0.56	0.74	0.67
Moment + Anisotropic diffusion	0.96	0.92	0.82	1.00	0.93

Table 4: Reduction ratio of texture resolution.

	Level 1	Level 2	Level 3	Level 4
Class1	1	1	1/2	1/2
Class2	1/2	1/4	1/8	1/8
Class3	1/8	1/8	1/8	1/8
Class4	1/2	1/4	1/8	1/8

4.2. Data Size and Quality of 3D Map. We investigate the data size and the quality of the 3D map with and without our resolution control technique. In this experiment, we set the reduction ratio of the resolutions as in Table 4, where 1/2 means that the resolution is reduced by half both in rows and columns. Note that all the texture images are stored in JPEG format.

Figure 11 illustrates a snapshot of the 3D map from some viewpoint, in which our resolution control is not applied, that is, the textures at original resolutions are used. Figure 12 shows the result obtained by our resolution control with the reduction ratio of Table 4.

We adopt the Visual Difference Predictor (VDP) [26] to quantitatively evaluate the visual quality. The VDP is an image assessment tool that models some properties of the HVS, such as nonlinearity, frequency selectivity, direction selectivity, and masking. The VDP outputs a probability map that predicts a probability of error visibility for each pixel. Thus higher values represent that errors are more perceivable. The numerical comparison is shown in Table 5 when it is displayed at the resolutions of 1024×768 and 640×480. The values in VDP75 and VDP95 of this table indicate the ratio of pixels in percentage that have higher probability than 75% and 95% in the probability map. Thus only a few percentages of pixels may be perceptually different. The data

Figure 11: Rendering scene without data reduction.

Figure 12: Rendering scene with our method.

Table 5: VDP and data size.

	Original	Our method (2024×768)	Our method (640×480)
Data size (MB)	41.1	10.6	10.6
VDP75 (%)	0	5.17	0
VDP95 (%)	0	2.8	0

size in Table 5 is the total sum of the texture data sizes to load and render for displaying Figures 11 and 12. In practice 25.8 % of data reduction is achieved. The total number of pixels to load can also be reduced by 25%. Note that, however, storage required by the algorithm is larger than the method without the data reduction by 87.5% since we need to prepare the textures of the sizes 1/2, 1/4, and 1/8 as well as the original textures. In the end, it can be seen from Figure 12 and Table 5 that we achieve a rendering quality without any significant visual loss while the amount of the data to load is reduced to one quarter. In other rendering examples we have tested, we have confirmed that our algorithm significantly reduces the data with little visual differences.

5. Conclusion

In this paper, we proposed the method that controls texture resolutions based on their features. By allocating low resolutions to visually unimportant textures, we reduce the data size to load for rendering without much degradation of quality.

Acknowledgments

The authors are grateful for the support of a 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.

References

[1] N. Haala, C. Brenner, and K. Anders, "3D urban GIS from laser altimeter and 2D map data," in *Proceedings of the ISPRS Commission III Symposium on Object Recognition and Scene Classification from Multispectral and Multisensor Pixels*, pp. 339–346, Columbus, Ohio, USA, July 1998.

[2] N. Haala, M. Peter, J. Kremer, and G. Hunter, "Mobile LiDAR mapping for 3D point cloud collection in urban areas: a performance test," in *Proceedings of the 21st International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS '08)*, vol. 37, part B5, Commission 5, p. 1119ff, Beijing, China, July 2008.

[3] N. Cornelis, K. Cornelis, and L. Van Gool, "Fast compact city modeling for navigation pre-visualization," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 2, pp. 1339–1344, New York, NY, USA, June 2006.

[4] M. Pollefeys, D. Nistér, J.-M. Frahm, et al., "Detailed real-time urban 3D reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[5] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 99–108, New Orleans, La, USA, August 1996.

[6] H. Hoppe, "View-dependent refinement of progressive meshes," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pp. 189–198, Los Angeles, Calif, USA, August 1997.

[7] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*, Morgan Kaufmann, San Francisco, Calif, USA, 2003.

[8] R. Pajarola, "Large scale terrain visualization using the restricted quadtree triangulation," in *Proceedings of the IEEE Visualization Conference (Vis '98)*, pp. 19–26, Research Triangle Park, NC, USA, October 1998.

[9] F. Losasso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids," in *Proceedings of the 31st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '04)*, pp. 769–776, Los Angeles, Calif, USA, August 2004.

[10] H. Hoppe, "Smooth view-dependent level-of-detail control and its application to terrain rendering," in *Proceedings of the IEEE Visualization Conference (Vis '98)*, pp. 35–42, Research Triangle Park, NC, USA, October 1998.

[11] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "BDAM—batched dynamic adaptive meshes for high performance terrain visualization," *Computer Graphics Forum*, vol. 22, no. 3, pp. 505–514, 2003.

[12] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "Interactive out-of-core visualization of very large landscapes on commodity graphics platform," in *Proceedings of the 2nd International Conference on Virtual Storytelling (ICVS '03)*, pp. 21–29, Toulouse, France, November 2003.

[13] J. Döllner and H. Buchholz, "Continuous level-of-detail modeling of buildings in 3D city models," in *Proceedings of the 13th ACM International Workshop on Geographic Information Systems (GIS '05)*, pp. 173–181, Bremen, Germany, November 2005.

[14] J. Hu, S. You, and U. Neumann, "Approaches to large-scale urban modeling," *IEEE Computer Graphics and Applications*, vol. 23, no. 6, pp. 62–69, 2003.

[15] Y. Takase, K. Yano, T. Nakaya, et al., "Visualization of historical city Kyoto by applying VR and web3D-GIS technologies," in *Proceedings of the 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST '06)*, Nicosia, Cyprus, October-November 2006.

[16] H. Wang, Y. Wexler, E. Ofek, and H. Hoppe, "Factoring repeated content within and among images," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–10, 2008.

[17] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, Mass, USA, 1995.

[18] H. Inatsuka, M. Uchino, and M. Okuda, "Level of detail control for texture on 3D maps," in *Proceedings of 11th International Conference on Parallel and Distributed Systems Workshops (ICPADS '05)*, vol. 2, pp. 206–209, Fukuoka, Japan, July 2005.

[19] B. A. Wandell, *Foundations of Vision*, Sinauer Associates, Sunderland, Mass, USA, 1995.

[20] F. Long, H. J. Zhang, and D. D. Feng, "Fundamentals of content-based image retrieval," in *Multimedia Information Retrieval and Management*, D. Feng, W. Siu, and H. Zhang, Eds., pp. 1–26, Springer, Berlin, Germany, 2003.

[21] M. Flickner, H. Sawhney, W. Niblack, et al., "Query by image and video content: the QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[22] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[23] G. Pass, R. Zabih, and J. Miller, "Comparing images using color coherence vectors," in *Proceedings of the 4th ACM International Multimedia Conference*, pp. 65–73, Boston, Mass, USA, November 1996.

[24] R. Zhang and Z. M. Zhang, "A clustering based approach to efficient image retrieval," in *Proceedings of the 14th International Conference on Tools with Artificial Intelligence (ICTAI '02)*, pp. 339–346, Washington, DC, USA, November 2002.

[25] A. A. Goodrum, "Image information retrieval: an overview of current research," *Informing Science*, vol. 3, no. 2, pp. 63–67, 2000.

[26] S. Daly, "The visible difference predictor: an algorithm for the assessment of image fidelity," in *Digital Image and Human Vision*, A. B. Watson, Ed., pp. 179–206, MIT Press, Cambridge, Mass, USA, 1993.