AUTOMATIC TEXTURE EXTRACTION FROM AERIAL PHOTOGRAPHS USING THE ZI-BUFFER

Dong Yeob Han*, Yong Il Kim**, Ki Yun Yu**, Hyo Seong Lee***, Byoung Uk Park****

* Chonnam National University, hozilla@chonnam.ac.kr
** Seoul National University
*** Sunchon National University
**** Hankyong National University

ABSTRACT 3D virtual modeling such as creation of a cyber city or landscape, or making a 3D GIS requires realistic textures. Automatic texture extraction using close range images is not yet efficient or easy in terms of data acquisition and processing. In this paper, common problems associated with automatic texture extraction from aerial photographs are explored. The ZI-buffer, which has depth and facet ID fields, is proposed to remove hidden pixels. The ZI-buffer algorithm reduces memory burden and identifies visible facets. The correct spatial resolution for facet gridding is tested. Error pixels in the visibility map were removed by filtering.

KEY WORDS: ZI-buffer, texture extraction

1. INTRODUCTION

3D virtual modeling has a variety of applications such as simulating a cyber city, a landscape, virtual war, public works, and in creating a 3D Geographic Information System (GIS). The demands for such modeling have been increasing. In the past decade texture-mapping techniques have advanced to real-time perspective simulations of real-world areas. Its concern is not to extract texture images but to map them directly onto the screen. A database needs to be created for the required texture to increase the reality of modeling or for the recognition of subfeatures. Generally, close range images for eye-level viewing and aerial images for aerial-level viewing are appropriate. A vehicle-borne sensor system has been developed for generating a textured computer aided design (CAD) model of the outdoor urban environment and detailed textures obtained from ground level images can be incorporated. However, it is difficult to automatically obtain a consistent texture using close range images that are taken from various urban street environments, including close buildings, vehicles, trees, people, and other variables. Consequently, an aerial photograph was used to generate relatively uniform results for large areas. In extracting textures, it is important to differentiate between visible and hidden surfaces. Many techniques have been proposed to remove hidden pixels in computer graphics. Using Triangular Irregular Networks (TINs) triangulated from 3D range sensor data, hidden triangles can be removed by back-face detection. It is however, difficult to triangulate coarse points in CAD building models created previously. The Z-buffer algorithm may be applied to a CAD model without additional processes to generate its topology. With urban scenes that contain many thousands of surfaces, the Z-buffer method takes a nearly constant processing time. On the other hand, the Z-buffer algorithm requiring massive memory may pose problems, especially when using large datasets such as aerial photograph and satellite imagery.

In this paper, a ZI-buffer algorithm is proposed which has facet ID information and decreases memory use. The image texture coordinates can be converted into object coordinates through rotation. Furthermore, the maximum spatial resolution in facet gridding corresponding to the grid size of polygons determined by the screen size used in computer graphics was explored. Filtering was used to remove errors from the facet ID visibility map.

2. VISIBILITY MAP GENERATION

2.1 Facet Gridding

Textures extracted from aerial photographs can be used for the measurement and recognition of subfeatures or objects such as road markings, cars, roof structures, and windows. This work must be performed accurately and without reduction of image resolution. Rotational transformation is made whilst maintaining the area and the shape of facets, and the relationship between grid and 3D coordinates can be obtained. When a facet is to be rotated about an axis that is not parallel to one of the coordinate axes, a composite matrix can be set up (see Eqn. 1). Rotated 2D facets were rasterized to generate a visibility map. Facet image coordinates can be converted to object coordinates by the facet's inverse rotation matrix:

$$R(\theta) = R_z(\theta)R_y(\beta)R_x(\alpha)T \qquad (1)$$

Where T indicates the translation. $R_x(\alpha)$, $R_y(\beta)$, $R_z(\theta)$ indicate the rotation matrix about the x, y, and z axis.

Aerial Photographs taken over landscapes exhibit a continuous range of scales associated with the variations in terrain elevation [2]. In a visibility map, the visible pixels must exclude the hidden ones. Therefore, the facet grid interval must be smaller than that for the maximum height of the study area. The photograph scale and the photo distance d determine the ground distance. The facet's maximum grid interval Dmax can be derived as follows:

$$S_{\max} = \frac{f}{H - h_{\max}} \quad (2)$$

$$D_{\max} = \frac{d}{S_{\max}} \quad (3)$$

Where f is camera focal length, H is aircraft flying height above some datum, hmax is the area maximum height, Smax is its maximum scale. When considering spatial aliasing and the measurement error in variables, the facet grid interval can be a little smaller than Dmax.

Algorithm 1 ZI-buffer algorithm

```
FacetID, Zbuffer = A Large Number
For F(1):F(n)          /* each Facet
  F(i)_P(r,c) = f(Rotate & Rasterize)
End
For P(r,c) of F(i)
  XYZ = invf(r,c)
  [m, n] = Collinearity Equation(XYZ)
  Calculate the depth z
  If (z < Zbuffer(m,n))
    Zbuffer(m,n) = z
    FacetID(m,n) = i
  End
End
FacetID = Noise Removal(FacetID);
For P(r,c) of F(i)
  ... // same as above
  If (i == FacetID(m,n))
  P(r,c) = Image(m,n);
End
```

2.2 ZI-buffer

The conventional Z-buffer algorithm stores a depth value for each pixel in a view plane. It then determines the visibility of remaining polygons based on these depth values. If the calculated depth value is less than the stored one, the depth image and visible intensity are refreshed in that position. Because of the massive size of aerial photograph data sets, which generally exceed 100 Mbytes, and the facet texture dataset, some of which is nearly 1 Mbyte for one tall building, it is both difficult and time consuming to load all the facet grids and the visibility map in memory at the same time. A ZI-buffer algorithm is therefore proposed to reduce the memory burden and identify visible facets. The ZI-buffer algorithm generates depth values and current visible facet IDs (2 byte integer) in image space. Facet grid files saved as mask images in gridding facets are processed individually. After a depth value is unloaded from memory, the texture of each facet can be easily generated from the facet ID image (or visibility map). The facet ID can be used to detect its error from the visibility map and to verify whether a certain facet ID is visible or not. The procedure of employing the ZI-buffer is summarized in Algorithm 1. Table 1 shows that the memory occupation of the ZI-buffer is about half that of the Z-buffer. The main concern is to load all the facets in the Z-buffer. Considering that the size of study area is one fifth of an aerial photograph and only 28 buildings among about 70 buildings in study area are used, the task could be larger. Because the texture extraction is focused, the computational cost is not great.

Table 1. The comparisons of memory allocation in the study area†

Components	Z-buffer(Mbytes)	ZI-buffer(Mbytes)	
		Step 1	Step 2
Depth map	104.95	104.95	0
Intensity	26.24	0	26.24
Facet ID image	52.47	52.47	52.47
All Facets	110.51	Max. 2.22	Max. 2.22
Total memory	294.17	159.64	80.93

† The size in image space: 5089×5406 pixels, facet spatial resolution: 10 cm, maximum ground resolution distance: 12 cm. The depth map data type is single float

10	10	10	915.50	915.50	915.49
9	1	10	915.58	958.40	915.58
9	9	9	915.63	915.62	915.61

(a) facet ID : depth

1012	9	9	997.34	915.20	915.19
1012	1	9	997.34	915.25	915.24
1012	1	9	997.35	915.30	915.29
1012	1	9	997.35	915.39	915.38

(b) facet ID : depth

Figure 1. Visibility map errors: (a) not removed by visible facets, (b) two facets have same depths. Left figure is facet ID and right figure is depth value.

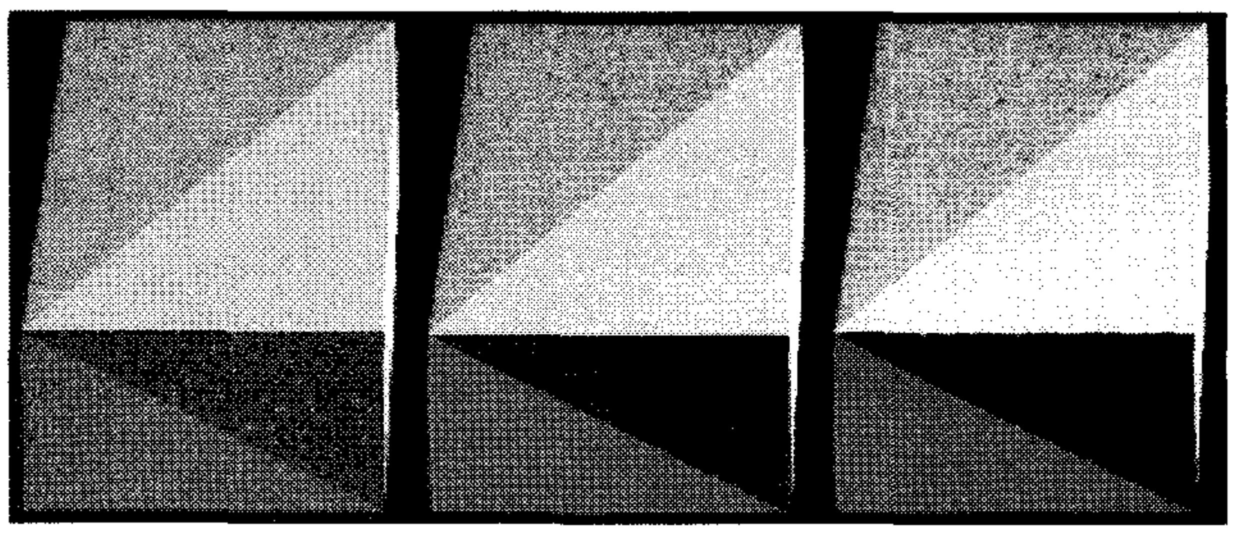
2.3 Noise Pixel Filtering

The visibility map shows occasional errors in facet boundaries (see Fig. 1). The reasons for these are: (1) the XYZ coordinate used in two facet grids has the same depth value, and (2) the facet grid does not completely eliminate hidden pixels because of an incorrect CAD model. Generally a mean, median and Gaussian filter remove noises but are not appropriate in this case because the correct facet ID in the visibility map must not be changed and the computed facet ID cannot be used.

The spot noise of the visibility map was processed first. When the facet ID and depth value in the 3×3 filter were checked, it was replaced with a facet ID similar to its depth value instead of a majority value. Secondly, if the ratio of facet visible area and projected area in the visibility map was less than 0.05, the facet pixels were replaced with an adjacent facet ID according to the depth value. The facet texture was generated using this noise-removed visibility map.

3. EXPERIMENTS AND CONCLUSIONS

The experiments were performed using a grayscale aerial photograph with a 1:5,000 nominal scale, 12 μm pixel size and 948 m flight height. The Jongno-gu, Seoul study area covers the suburb of Deoksugung and contains some tall buildings. A total of 1340 triangles were created from a CAD model of 28 buildings and some terrain points (see Fig. 4). A building of 83.8 m height was used for facet gridding as depicted in Fig. 2. One pixel ground distance at the maximum height of this building is 6.32 cm. When the spatial resolution of the facet grid used was 6.32 cm the facet boundary was not smooth. A facet grid resolution of less than 6 cm was more appropriate. The visibility map generated by the ZI-buffer algorithm had error pixels, which can be removed through filtering. Finally, all the facet textures were obtained from the visibility map. These textures can be used in the texture library, features measurement, realistic visualization and other applications.


(a) (b) (c)
Figure 2. The visibility map according to varying facet spatial resolution; (a) 0.07m, (b) 0.0632m, (c) 0.05m

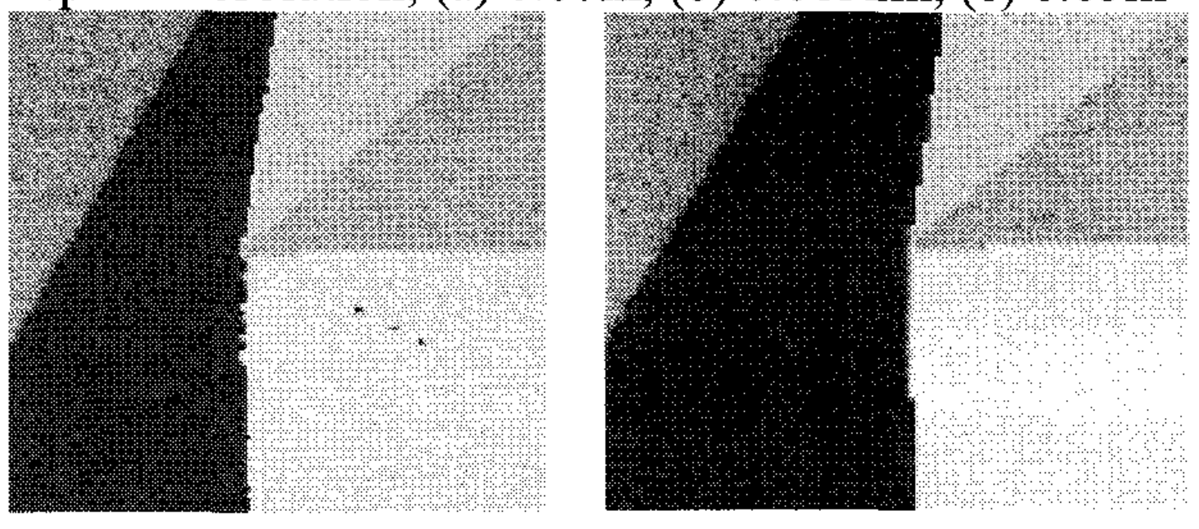


Figure 3. Visibility map; (left) before noise removal, (right) after noise removal

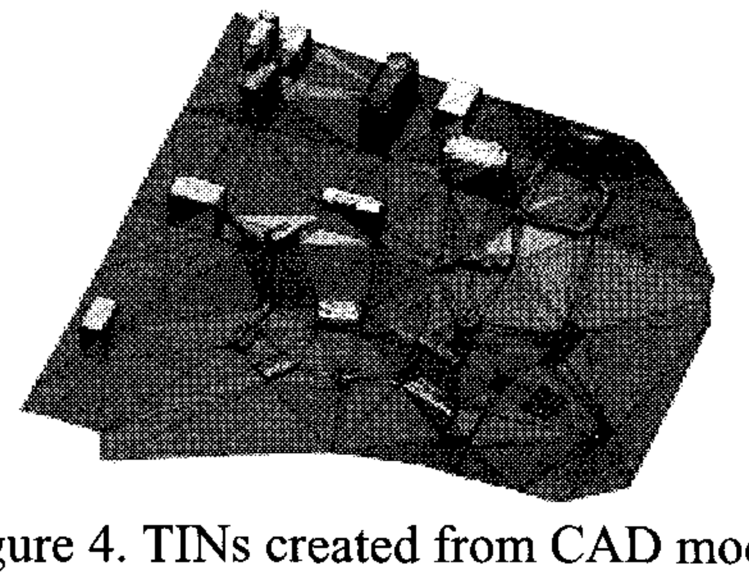


Figure 4. TINs created from CAD model

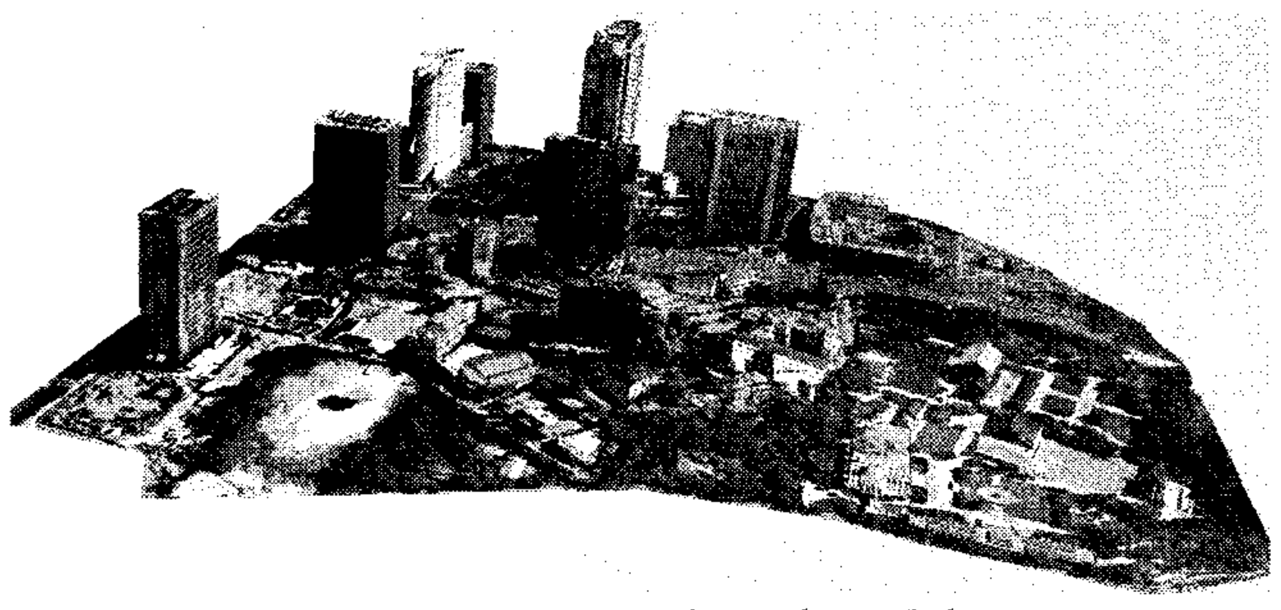


Figure 5. Textured vrml model

References

Donald Hearn, M. Pauline Baker, Computer Graphics 2nd Ed., Prentice Hall 1997.

Thomas M. Lillesand, Ralph W. Kiefer, Remote Sensing and Image Interpretation 4th Ed., John Wiley & Sons, 2000.

Daniel Cohen-Or, Eran Rich, Uri Lerner and Victor Shenkar, "A Real-Time Photo-Realistic Visual Flythrough", IEEE Visualization and Computer Graphics, Vol. 2 , No. 3, September 1996, pp.255~265.

Frederick M. Weinhaus, Venkat Devarajan, "Texture Mapping 3D Models of Real-World Scenes", ACM Computing Surveys, Vol. 29, No. 4, 1997, pp.325~365.

John Danahy, "Visualization Data Needs in Urban Environmental Planning and Design", Photogrammetric Week, 1999.

Sabry F. El-Hakim, Claus Brenner, and Gerhard Roth, "A multi-sensor approach to creating accurate virtual environments", ISPRS Journal of Photogrammetry & Remote Sensing, Vol. 53, No. 6, December, 1998, pp. 379-391.

Sung Chun Lee, Soon Ki Jung, and Ram. Nevatia, "Automatic Pose Estimation of Complex 3D Building Models", 6th IEEE Workshop on Applications of Computer Vision, Orlando, Florida, December, 2002.

Maresch M., D. Scheiblhofer, "A vehicle based multi-sensor platform for facade recording", OeAGM 96, 1996.

Huijing Zhao, Ryosuke Shibasaki, "Reconstructing a textured CAD model of an urban environment using vehicle-borne laser range scanners and line cameras", Machine Vision and Applications, Vol. 14, No. 1, 2003, pp. 35~41.

Sithole, G., Digital photogrammetry for automatic photo-texture extraction, MSc thesis, ITC, The Netherlands, 1997.

Zlatanova, S., 3D GIS for Urban development, PhD thesis, ITC, The Netherlands, 2000.