Extraction and Integration of Window in a 3D Building Model from Ground View images

Sung Chun Lee and Ram Nevatia
Institute for Robotics and Intelligent Systems,
University of Southern California
Los Angeles, California 90089, USA
{sungchun|nevatia}@usc.edu

Abstract

Details of the building facades are needed for high quality fly-through visualization or simulation applications. Windows form a key structure in the detailed facade reconstruction. In this paper, given calibrated facade texture (i.e. the rectified texture), we extract and reconstruct the 3D window structure of the building. We automatically extract windows (rectangles in the rectified image) using a profile projection method, which exploits the regularity of the vertical and horizontal window placement. We classify the extracted windows using 2D dimensions and image texture information. The depth of the extracted windows is automatically computed using window classification information and image line features. A single ground view image is enough to compute 3D depths of the facade windows in our approach.

1 Introduction

Modeling and visualization of an urban area is important for many applications including entertainment and urban mission planning. Reconstruction of 3D building models is one of the key parts in the urban site modeling. Recently, modeling of more detailed 3D structure of a building is demanded for better quality applications, such as fly-through rendering and simulation for mission planning.

A variety of computer vision problems arise in the process of obtaining the detailed structure of 3D building. Focus of many efforts has been on extraction of 3D building models from aerial images [1, 2, 3, 4]. However, these methods largely provide roof models though some low resolution facades textures may also be captured as in work of Wang and Hanson [4]. For many applications, we need detailed descriptions of the facades (the vertical walls) and the structures of the objects such as windows in them; this is the focus of this paper.

Modeling facades poses additional problems to those of roof modeling. The features and structures in the facade need to be extracted and classified. The facades may contain several details that are not of interest, occlusion by vegetation and other objects is common and glass windows exhibit reflections or transparency.

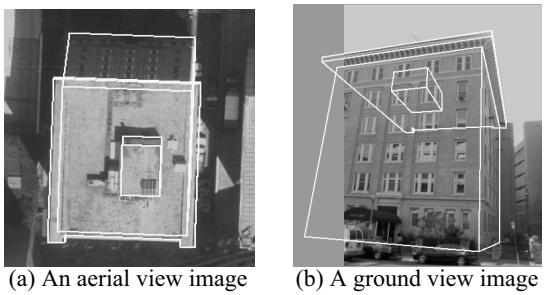
A classical work on facade modeling is described in [5]. This system requires considerable manual processing and captures only the facade texture and not its detailed structure (windows are not explicitly described). In more recent work, approaches with high resolution range data from ground have been developed.

Range data based approaches are attempted to reconstruct the detailed 3D building models [6, 7]. Their approach fuses different data to compensate for the accumulated errors of range data. However, the generated 3D building models are a mesh structure and do not make the structure explicit.

Integrated methods for different data sources have tried to get the 3D building models as well as the window descriptions [8, 9]. However, even though this approach exploits architectural knowledge such as repeating patterns of windows, it is limited to reconstructing the 3D building models that have only one type of window in a row. Model based approaches have been investigated to model the detailed 3D building structure [10, 11, 12]. Werner and Zisserman [10] recently described a method for reconstructing the detailed structure of a building and its facades by fitting primitive blocks. Performance of their method is highly dependent on the accuracy of line matching. Dick *at el.* [11, 12] reconstruct the 3D facade primitives using prior 3D primitive blocks based on a Bayesian framework. However, these approaches depend on strong priors, which makes their algorithms have a limitation on scalability.

We take a different approach. We assume that rough wireframe models are available from aerial image modeling [13] but they do not contain any facade textures; an example is shown in Figure 1 (a). Uncalibrated camera ground images are used for facade modeling; we align ground view images to the 3D building wireframe models by using a projective geometry method [14, 15] as the result of the projected 3D building is shown in Figure 1 (b). Then, we model the structures in the facade from these rectified camera images; this is the topic of the current paper.

Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)
1063-6919/04 $20.00 © 2004 **IEEE**



(a) An aerial view image (b) A ground view image

Figure 1. Overlaid 3D building models in an aerial view and a calibrated ground view image.

An overview of the proposed method is shown in Figure 2. There are two major steps to acquiring the detailed 3D structures of the windows: 2D feature (window) extraction and classification, and 3D reconstruction of the 2D windows as depicted two gray boxes in Figure 2. The rectified facade image textures are provided to the proposed system. Since the windows have regular shapes in the rectified facades, we extract window candidates by using an image projection profile based method. The classification of the window candidates is done by using width, height and image texture. Arch windows are identified from the classified windows. We compute the depth of the classified windows using a window plane sweeping method. The windows in the same class have the same window frame structures (*e.g.* sashes) and depth. We generate the 2D image features inside the windows and sweep them into the direction of the facade inner normal direction to search the correct depth of the window. Given 3D building wireframe models, we can construct the detailed 3D structures of the building windows using a minimum number of ground view image, including single if it covers the desired features.

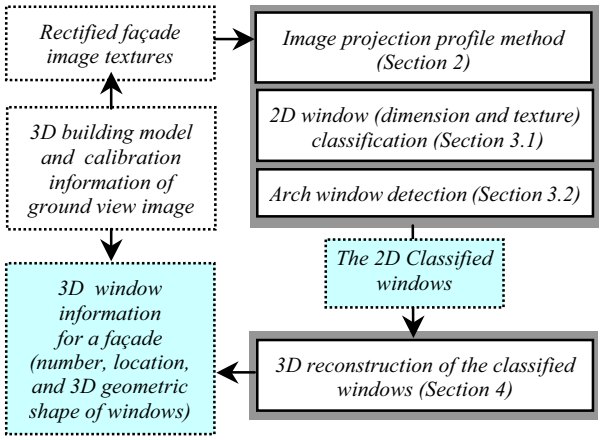


Figure 2. Overview of the proposed method (solid box:procedure, dotted box:input/output data).

2 Extraction of 2D window candidate

The major shapes of the building windows are assumed to be arch and rectangular in the rectified facade texture. In this section, we explain the extraction of the candidate areas (rectangles in the rectified facade texture for both arch and rectangular windows).

2.1 Acquisition of ground view image

We first obtain a rectified facade texture from the calibrated ground view images using our previous method [14, 15]. When multiple rectified images for a facade are available, we blend them using a median method that selects the median value among multiple intensity values for each pixel. Even if only one image is available, we can use it as long as it covers the entire facade.

2.2 Projection profile based approach

There have been efforts on finding patterns on the plane [16, 17]. However, they would not apply to facades with different patterns (shapes) of windows. We propose a projection profile based approach to extract 2D rectangles by exploiting the geometric property of 2D rectangles and alignment of the building windows.

We project horizontal (parallel to the ground) and vertical (perpendicular to the ground) image edges in horizontal and vertical directions to give a total of two projection profiles: a horizontal projection profile of the horizontal edges (H_h) and a vertical projection profiles of the vertical (V_v) edges as shown in Figure 3.

(a) Horizontal projection: H_h (b) Vertical projection: V_v

Figure 3. Two projection profiles.

Because the building windows are horizontally or vertically aligned, the image edges within the windows of the same column or row are accumulated at the same location of the projection profile histograms. There exist high frequency peaks in the horizontal and vertical histograms where the windows are located. The window candidates can be segmented by intersecting the vertical and horizontal projection profile histograms. Because there exist

image textures other than windows on a facade texture, we should choose appropriate threshold values to select those peaks that corresponds to the x and y position of windows.

We derive an automatic method of selecting a threshold value by fusing vertical and horizontal edge projection profiles. For example, to decide the width of the windows, we choose a threshold value to cut the profile, H_h. Note that good threshold candidates are located at the valleys in the profile projections H_h and V_v.

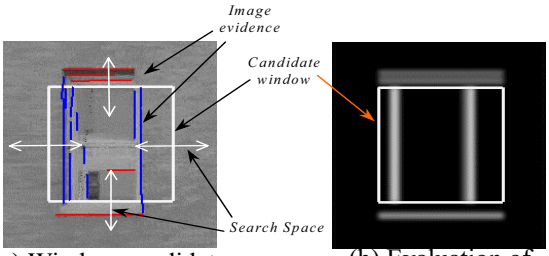
We select threshold candidates near the valley area of the projection and test each set of threshold candidates using an image based evaluation method. We pick a threshold from horizontal threshold candidate set and generate temporary horizontal lines of the windows. We compute image feature supports for each generated line. Image supporting evidence consists of line segments in the image, which confirm the generated line. The supporting image evidence (line segment) must form a small acute angle (angle threshold, a_t: *10 degree*) with the line and the perpendicular distance of the midpoint of the image line segment from the generated line should be within a distance threshold (d_t: *1% of min (image width, image height)*). These two (angle and distance) thresholds are chosen based on our experiments. Later stages of processing can correct some of the errors caused by incorrect choice of thresholds as shown later.

Once supporting image evidence is collected for the generated line, their coverages are calculated by projecting two end-points of the supporting edge onto a hypothesized line. Note that the overlapping coverages are counted only once and normalized by the length of the hypothesized line such that the coverage score is between 0 and 1. We choose the threshold value that has the best coverage score. We also apply this method to find vertical threshold value.

2.3 Image based refinement of candidate areas

The candidate windows obtained by using the projection profile method may not be highly accurate when there exist different shape of windows in the same column as shown in Figure 4 (a). We refine each candidate window by conducting a one dimensional search for the four sides respectively.

For each line of the window, we generate hypothesized lines by moving the line to its perpendicular direction and test the hypothesized line (l_h) using image supporting evidence ($\{l_i\}$, *i = 1 ... n, where n is the number of supporting evidence for l_h*). This time, we compute the coverage, $S(l_h)$ with weight of the distance between l_h and l_i as follows:



(a) Window candidate and image evidences (b) Evaluation of hypotheses

Figure 4. Search for good window frames.

$$S(l_h) = \sum_{i=1}^{n} c_i \, e^{-d_i^2/d_t} \qquad (1)$$

where c_i is the coverage score of l_i for l_h as explained in Section 2.2, d_i is the mid-point distance between l_i and l_h, and d_t is the distance threshold as mentioned above.

To compute more accurate position of the window boundary, we also collect negative image evidence. The negative image evidence ($\{l_j\}$, *j = 1 ... m, where m is the number of negative evidence for l_h*) must intersect with l_h within the image line segment (*$0.05 \le s \le 0.95$, s is an normalized position of the intersection point between l_j and l_h*). The contribution of a line l_j in the set of negative evidence lines to the negative score is the angle difference weighted by the length of l_j.

We compute the evaluation score for the hypothesized position by subtracting the negative evidence score from Equation (1). We choose the position where the hypothesized line has the best score for the window boundary line. We apply this method for the four directions.

The immediate results of extracting windows are shown in Figure 5. Figure 5 (a) shows an extracting result by the projection profile method and Figure 5 (b) shows a result of our image based refining method. The ground floor windows are poorly extracted due to too many occluding objects such as brushes.

(a) (b)

Figure 5. Immediate results of extracting windows.

3 Classification of 2D windows

Windows are different according to its size, appearance, and shape (rectangle or arch). Given the candidate windows (rectangles), we classify them using their 2D dimensional properties such as width or height and image texture information as will be explained in the Section 3.1. We might need refinement or realignment for the classified window group. Then, we apply a shape classification method to identify arch windows as described in the Section 3.2.

3.1 Classification

We first classify the extracted windows according to dimension and texture similarities. We use a normalized cross-correlation function, *CR(x, y)* for an image measurement of two windows *s* and *d* defined by the following equation:

$$CR(x,y)=\frac{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} I_s(x',y')I_d(x+x',y+y')}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} I_s(x',y')^2 \sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} I_d(x+x',y+y')^2}}$$

where I_s and I_d are image texture of window *s* and *d*, *I(x, y)* is the intensity value in pixel (*x, y*), *h* is the height of window, *w* is the width of window.

CR(x, y) characterizes the similarity between two images and its value is normalized so that it can be between 0 and 1. Image similarity, *IS(s, d)* between two windows *s* and *d* is chosen by the global maximum value of the cross-correlated result.

For dimension measure, we use the euclidean distance in the two dimensional space defined by the width and the height of window. If the distance is larger than distance threshold, $2d_t$, the two windows are clarified to be of different shape. Otherwise, we compute the dimensional similarity, *DS(s, d)* function for two windows, *s* and *d* as following:

$$DS(s,d) = \frac{1}{4}\left(\sum_{i=1}^{4}\frac{(S_s(l_i)+S_d(l_i))}{2}\right) e^{-d_e^2/(2d_t)}$$

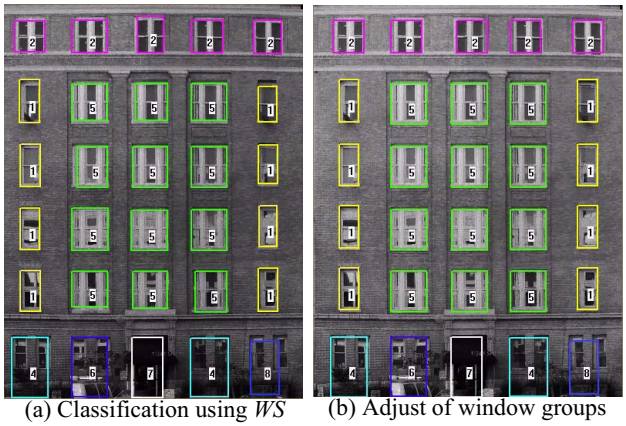
where $S_s(l)$ and $S_d(l)$ are coverage score function for each window boundary line *l* in the windows *s* and *d* respectively as defined in Equation (1), d_e is the euclidian distance in width-height space, and d_t is the distance threshold as mentioned above. Note that we multiply the normalized coverage score value with the distance similarity because the normalized coverage score implies the confidence probability of this similarity, *i.e.* it indicates how strongly image edges support this similarity.

The same class windows can have different image texture due to reflection or transparency of window glass as depicted in Figure 5 (*e.g.*, windows *a* and *b*). The size of windows in the same class can be different because of missing or noisy edges (e.g., windows *c* and *d* in Figure 5). This suggests that we should combine two classification criteria, *IS* and *DS* to classify the extracted windows. The window similarity, *WS* is defined as following:

$$WS(s, d) = (IS(s, d) + DS(s, d)) / 2 \qquad (2)$$

We chose to average the Dimensional Similarity and the Image Similarity for simplicity. More complex strategies, such as a Bayesian combination are possible. However, we find that simple averaging gives good results.

Grouping procedure is straightforward. For those windows that are not labeled yet, we compute *WS* with all other windows. If *WS* is less than s_t (*similarity threshold: 0.4*), then we label two windows differently. Otherwise, we label two windows with same class number. When two windows already have class numbers, we merge two window classes into one. The grouping window result is shown in Figure 6 (a). The same window class displays the same color and class numbers are shown in each window. After grouping windows, we adjust each window so that windows in the same class should be aligned vertically and horizontally, and have the median width and height out of the same window group as shown Figure 6 (b).



(a) Classification using *WS* (b) Adjust of window groups

Figure 6. Results of grouping windows.

3.2 Arch window detection

We define two primitive window shapes: arch and rectangular. Given the extracted rectangles of windows, we check whether the extracted one is *arch* or *rectangular* shape window. The arch window consists of an arch

(above) and a rectangle (below) part as depicted in Figure 7. Arch window classification result depends on the result of window detection. Thus, we have to refine the extracted windows before shape classification.

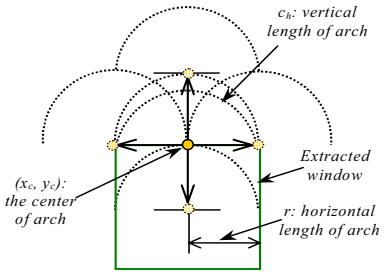


Figure 7. Required parameters to form an arch.

An arch-shaped window consists of the top arch and the bottom rectangular parts. The top arch is defined as a half ellipse by the following parametric equations:

$$x(t) = x_c + r\cos t$$
$$y(t) = y_c + c_h \sin t$$

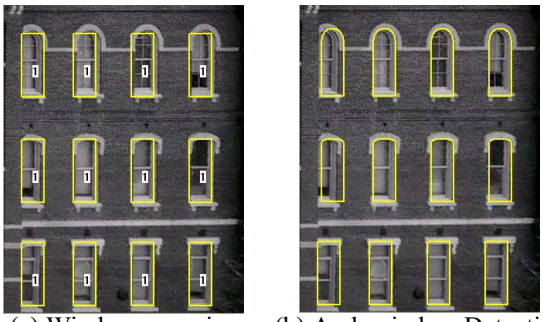
where $t = 0^o,..., 180^o$, (x_c, y_c) is the center of the arch, r is a horizontal length, and c_h is a vertical length, which defines the curvature of the arch.

We set r to be half of the width of the extracted window and find the other parameters, (x_c, y_c) and c_h as illustrated in Figure 7. We use a Hough transform in three dimensional parameter space (x_c, y_c, *and* c_h). One cell in this parameter space is mapped to a hypothesized arch in the image plane. We accumulate in each cell with the number of the image points that are located on the corresponding hypothesized arch and choose the maximum accumulated cell. If the vertical length, c_h of the maximum cell is higher than zero, we decide this window class as an arch window. Otherwise, this window class is set to a rectangular window.

Figure 8 shows a result of 2D classification. Since the arch window has a rectangular part, there is no significant influence of window shape (arch or rectangle) in window detection stage as shown in Figure 8 (a). The windows in the top and second floors are successfully detected as arch windows in Figure 8 (b).

4 3D Reconstruction of windows

Because the extracted windows are on the building facade surface, we can compute 3D position of the extracted windows using the known 3D building model information. However, the indented plane of the window such as for a recessed window, is not located on the facade surface. We aim to compute the applicable depth for the indented windows. This depth could be computed by



(a) Window grouping (b) Arch window Detection

Figure 8. 2D shape classification result.

using matches for the linear features within the extracted window in two or more ground views. However, we exploit the information that the windows in the same class group should have the same texture without noise or reflection. We have found it more robust to simply conduct a plane sweeping search for different depth values. Unlike previous approaches [7, 8, 10, 12], our method can work just a single or few ground view image to reconstruct 3D window structures.

Since glass has high reflectance or transparency property, there exist many noise image features within the window, corresponding to the surrounding objects. We need to remove the noise features before we reconstruct the 3D structure of the windows. We use the image features from the ground view image rather than the rectified image to avoid losing feature information due to the rectification process.

We project the extracted 2D windows (rectangles) into the image plane and collect image features within the projected windows. Then, we filter out the image features by using length, direction, and boundary constraints as illustrated in Figure 9.

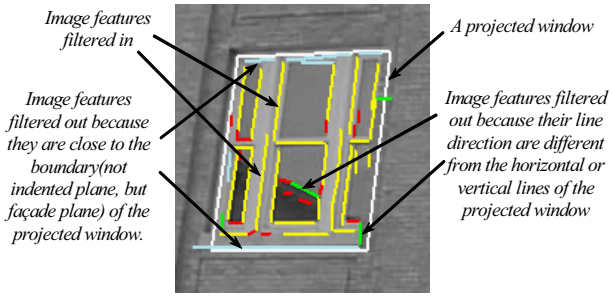


Figure 9. Filtered image features.

4.1 A plane sweeping method to find 3D depth

Windows in the same class are assumed to have the same 3D depth and window sash structure, which means that the projected 2D window of the same 3D window class should have the same image textures after the filter-

ing process. Then, the image features of a window should match with those of another window in the same window class. However, due to the perspective projection effect (*e.g.* the image of the window closer to the camera is larger than the farther one), direct image comparison between two windows on the image plane, is not reliable.

Instead of using the direct match, we derive a plane sweeping method for the best depth. From the known facade surface to the predefined maximum depth of the window, we defined the hypothesized indented plane using a hypothesized depth and the inner facade surface normal as follows:

$$ax + by + cz = d$$

$$d = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} p_x' \\ p_y' \\ p_z' \end{bmatrix}, \qquad \begin{bmatrix} p_x' \\ p_y' \\ p_z' \end{bmatrix} = d_h \begin{bmatrix} a & b & c \end{bmatrix}^T + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{3}$$

where $(a,b,c)^T$ is the inner facade surface normal vector, d_h is a hypothesized depth, (p_x, p_y, p_z) is a corner point of the facade, and (p_x', p_y', p_z') is a point on the hypothesized indented plane.

For each hypothesized indented plane, we do the followings:

- We compute a 3D line (l^i_n) by intersecting the indented plane with a plane formed by the camera center and an image line (e^i_n) of ith window ($n = 1,...E_i$, where E_i is the number of image features in ith window on the image plane) as illustrated in Figure 10.
- We obtain a new 3D line ($l^i_n{}'$) in the jth window by translating (l^i_n) as following:

$$l^i_n{}' = l^i_n + T_{ij}$$

  where T_{ij} is a translation vector between two 3D windows, w_i and w_j.
- We project the new 3D line ($l^i_n{}'$) into the image plane and get a projected image line ($e^i_n{}'$) as depicted in Figure 10.
- We check the projected image line ($e^i_n{}'$) with nearby image supporting evidence.

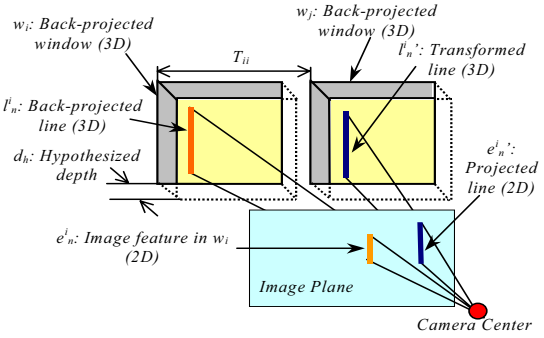


Figure 10. Generating a projected image line by transforming a image line from another window using a hypothesized depth.

The supporting evidence consists of nearby and overlapping image linear features as explained in Section 2.2. We check the hypothesized depth and accumulate the evaluation scores through k number of windows in the same window class. The depth providing the best score is selected as an indented depth for the window class.

When a building has many windows on its facade, only single ground view image is enough to find the correct depth because there exist a sufficient number of image features that we can find their matches using the window class information throughout the windows on the building facade.

5 Results and Discussion

We have tested our methods for extracting and classifying 2D windows and 3D reconstruction of the extracted windows on a number of examples. We show some results in this section.

5.1 2D window extraction results

We obtained building facade textures from Internet sites to test our method using four points plane homography. Figure 11 shows results of extracting and classifying 2D windows, and processing time. There are false alarms at the ground level of Figure 11 (b). Arch shaped windows are shown in Figure 11 (c).

(a) 335x631 : 19 secs (b) 300x449 : 10 secs (c) 300x411 : 13 secs

Figure 11. 2D window extraction and classification results (image size: processing time on Pentium II).

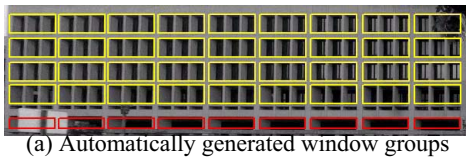
5.2 Semi-automatic window editing

After the automatic window extraction / classification process, we allow the user create or modify the 2D windows with few interactions by using underlying image features computed before. The adjustment operations for one window are adjusting boundary, moving, deleting, and copying/pasting a window.

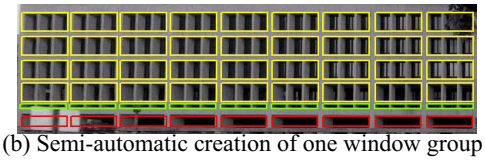
There also are semi-automatic adjustment operations for window groups such as splitting or creating windows. When the user selects a window to split, then the system

checks all the windows in the same window group to compute the proper number of windows within the window, the size of *unit window* (smaller window inside the window to be split) and the gap between the unit windows using supporting line segment evaluation as defined in Equation (1). In addition, when the user creates a window (simply draws a rectangle), the system checks the neighbor column or row of the created window to generate other similar windows in the row or column where there are no windows. After the dimension of the window is decided, we compute the image similarity, *CR* defined in Section 3.1 along the same row or column to find its similar windows.

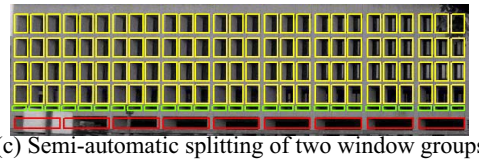
As shown in Figure 12 (a), the second row of windows from the ground are missed due to their small height and four rows of windows are merged due to narrow gap between windows. The user creates a window in the second row and the systems generates all other windows in the same row as depicted in Figure 12 (b). To split windows, the user selects any window in the same group and the system computes necessary parameters to split as shown in Figure 12 (c).



(a) Automatically generated window groups



(b) Semi-automatic creation of one window group



(c) Semi-automatic splitting of two window groups

Figure 12. Semi-automatic editing result.

5.3 3D window reconstruction results

We show a qualitative result of our plane sweeping plane method for finding 3D depth of each window group in the left image of Figure 13 (white rectangles are the window boundaries on the facade plane and red rectangles are the indented windows). The windows in the ground floor are adjusted by the user. The depths of the windows in the middle and the ground floor, are nicely recovered under the severe occluding or high reflection conditions.

To test our method in a quantitative aspect, we generated a synthesized 3D building model with known window depths (ground truth data). We captured arbitrary rendering views of the 3D building model to generate ground view images and calibrated them using our previous method [14, 15]. The right of Figure 13 shows a graphical result of our method. We project the computed indented windows into the image plane.

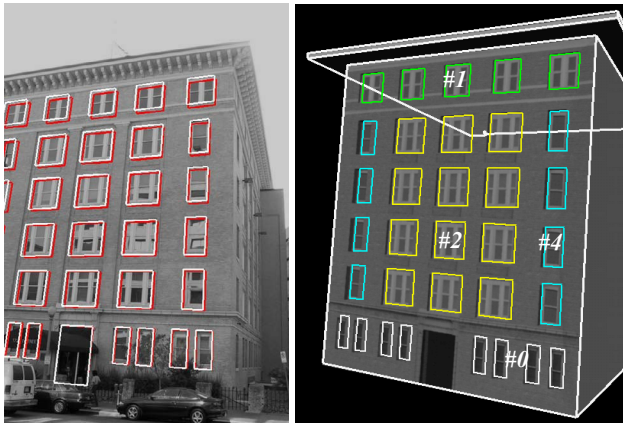


Figure 13. A qualitative result and a graphical result of the proposed sweeping plane method for finding 3D depth.

Table 1 shows quantitative analysis of accuracy of 3D depth inference as a function of the sweeping step by comparing the ground truth data with our results. Note that the smaller step, which means finer sweeping resolution, generates less errors, but requires more processing time. The windows in the middle (class #2) has the less errors than the other groups because it has more windows in the same class as shown in Figure 13.

TABLE 1. Depth errors according to search step (gt:ground truth, e:error, unit:meter).

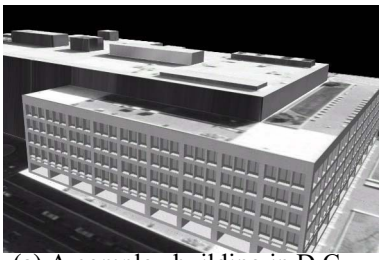
Search step	*Window Class #0 (gt depth=0.1)*	*Window Class #1 (gt depth=0.2)*	*Window Class #2 (gt depth=0.3)*	*Window Class #4 (gt depth=0.5)*
0.03	*0.12 (e=0.02)*	*0.12 (e=0.08)*	*0.27 (e=0.03)*	*0.42 (e=0.08)*
0.02	*0.12 (e=0.02)*	*0.14 (e=0.06)*	*0.28 (e=0.02)*	*0.44 (e=0.06)*
0.01	*0.119 (e=0.019)*	*0.139 (e=0.061)*	*0.28 (e=0.02)*	*0.43 (e=0.07)*
0.005	*0.105 (e=0.005)*	*0.14 (e=0.06)*	*0.275 (e=0.025)*	*0.43 (e=0.07)*
0.001	*0.107 (e=0.007)*	*0.144 (e=0.056)*	*0.275 (e=0.025)*	*0.431 (e=0.069)*

We generate a synthesized 3D building model with large recessed windows to reduce the number of visible image features. Errors between the computed depths and ground truth data are *0.08m, 0.06m, 0.02m*, and *0.08m* for each window class, *#0, #1, #2,* and *#4* respectively. In spite of a small number of visible image features, our method still shows the same results with Table 1 except window class *#0*. Since the window class *#0* has too few image features, the error becomes bigger.

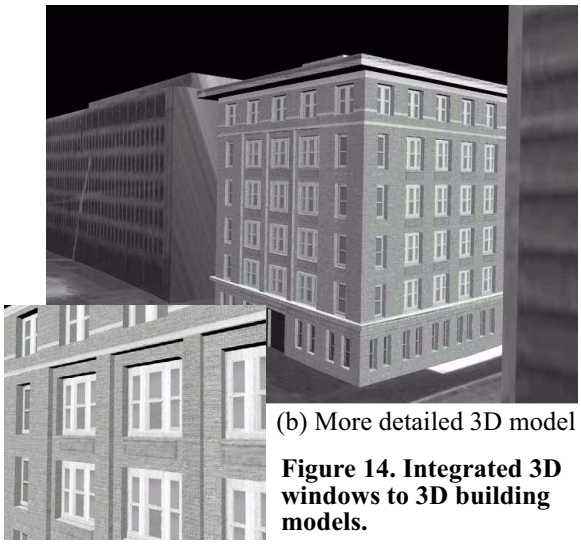
5.4 Integration result

One of application for reconstructing 3D windows is a fly-through rendering application. In our previous work

[13, 15], we extract 3D building models as well as facade structures such as indentations or protrusions as shown in Figure 14. We created the image texture for each window group and the background wall using the extracted window information. Then, we integrated the reconstructed 3D window models with 3D building models. Figure 14 (a) shows the result of a complex building in an urban site. Even though we only have one facade picture, we apply the reconstructed windows to another wall as shown in Figure 14 (a). Figure 14 (b) shows a high quality rendering result by our window reconstruction method. A close-up shot is also shown in Figure 14 (b).



(a) A complex building in D.C.



(b) More detailed 3D model

Figure 14. Integrated 3D windows to 3D building models.

6 Conclusion

In this paper, we have presented a method for reconstructing the 3D building windows from a single calibrated ground view image. We extract 2D windows by exploiting the properties of man-made building structure such as regularity and symmetry. The estimation of 3D depth for the extracted 2D windows is automatically computed by using the window class information and image features. Good performance is shown on some real examples but window groups with small number of rows may not be extracted as well due to insufficient image feature evidence.

References

[1] S. Noronha and R. Nevatia, Detection and modeling of buildings from multiple aerial images, *PAMI*, 23(5):501-518, 2001.

[2] Y. Hsieh, SiteCity: A Semi-Automated Site Modeling System, *CVPR*, 499-506, 1996.

[3] T. Strat, L. Quam, J. Mundy, R. Welty, W. Bremner, M. Horwedel, D. Hackett, and A. Hoogs, "The RADIUS Common Development Environment," *In Proc. of DARPA IUW*, 215-226, 1992.

[4] Wang and Hanson, Surface Texture and Microstructure Extraction from Multiple Aerial Images, *CVIU*, 83(1):1-37, 2001.

[5] P. E. Debevec, C. J. Taylor and J. Malik, Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach, *In SIGGRAPH*, 11-20, 1996.

[6] I. Stamos and P. K. Allen, Automatic Registration of 2-D with 3-D Imagery in Urban Environments, *CVPR*, 2:731-736, 2001.

[7] C. Fruh and A. Zakhor, 3D Model Generation for Cities Using Aerial Photographs and Ground Level Laser Scans, *CVPR*, 2001.

[8] S. Coorg and S. Teller, Extracting textured vertical facades from controlled close-range imagery, *CVPR*, 625-632, 1999.

[9] F. Taillandier, Texture and Relief Estimation from Multiple Georeferenced Images, *Master of Science Thesis*, DEA Algorithmique, Ecole Polytechnique, 2000.

[10] T. Werner and A. Zisserman, New Techniques for Automated Architecture Reconstruction from Photographs, *ECCV*, 2002.

[11] A. Dick, P.Torr, S. Ruffle, and R. Cipolla. Combining Single View Recognition and Multiple View Stereo for Architectural Scenes, *ICCV*, 2001.

[12] A. R. Dick, Philip H. S. Torr, and R. Cipolla, A Bayesian Estimation of Building Shape Using MCMC. *ECCV*, 2:852-866, 2002.

[13] S. C. Lee, A. Huertas, and R. Nevatia, Modeling 3-D complex buildings with user assistance, *WACV*, 170-177, 2000.

[14] S. C. Lee, S. K. Jung, and R. Nevatia, Automatic Integration of Facade Textures into 3D Building Models with a Projective Geometry Based Line Clustering, *Computer Graphics Forum (Euro Graphics)*, 21(3):511-519, 2002.

[15] S. C. Lee, S. K. Jung, and R. Nevatia, Integrating Ground and Aerial Views for Urban Site Modeling, *ICPR*, 4:107-112, 2002.

[16] Schaffalitzky, F. and Zisserman, A., Planar Grouping for Automatic Detection of Vanishing Lines and Points, *IVC*, 2000.

[17] Liu and Collins, A Computational Model for Repeated Pattern Perception using Frieze and Wallpaper Groups, *CVPR*, 537-544, 2000.