EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (2002), Number 3

Automatic Integration of Facade Textures into 3D Building Models with a Projective Geometry Based Line Clustering

Sung Chun Lee[†], Soon Ki Jung[‡] and Ram. Nevatia[†]

[†] Institute for Robotics and Intelligent Systems, and Computer Science Department, University of Southern California, CA, USA
[‡] Department of Computer Engineering, Kyungpook National Univeristy, Daegu, Korea

Abstract

Visualization of city scenes is important for many applications including entertainment and urban mission planning. Models covering wide areas can be efficiently constructed from aerial images. However, only roof details are visible from aerial views; ground views are needed to provide details of the building facades for high quality 'fly-through' visualization or simulation applications. We present an automatic method of integrating facade textures from ground view images into 3D building models for urban site modeling. We first segment the input image into building facade regions using a hybrid feature extraction method, which combines global feature extraction with Hough transform on an adaptively tessellated Gaussian Sphere and local region segmentation. We estimate the external camera parameters by using the corner points of the extracted facade regions to integrate the facade textures into the 3D building models. We validate our approach with a set of experiments on some urban sites.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Modeling packages

1. Introduction

Accurate 3D models of buildings in a city are needed for a variety of applications, such as fly-through rendering and simulation for mission planning. Many methods for extraction of such 3D models from aerial images have been developed in recent years [1, 2, 3]. Aerial images provide wide area coverage, however, the walls of buildings are either not visible or seen at a very low resolution. We need to utilize images taken from the ground to get accurate facade textures. We propose to integrate the use of such ground views with models obtained from aerial views. To do this, we need to estimate the global position and orientation of the camera used for taking the ground view images; then we can attach facade textures to 3D building model correctly.

In an urban area, it is hard to find ground views that capture an entire building since many buildings are close to each other, and narrow streets limit the field of view. Figure 1 depicts an example of a ground view image. The small field of view prevents the estimation of external camera parameters by using traditional methods, which use point correspondences, because not enough identifiable feature points such as corners of building are visible in a ground view image. Because of this constraint, a line based approach should be considered for pose estimation of ground view camera.

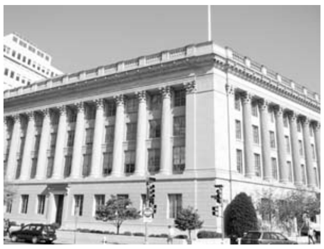


Figure 1: *A ground view image.*

1.1. Related Work

Some research has been done for 3D reconstruction of architecture by using 3D to 2D line correspondences [4, 5]. However, these approaches are applicable only for one or two buildings, rather than an entire urban site, as they require laborious user interactions for a large number of models. To reduce user interactions for the correspondences, automatically obtainable image features such as vanishing points that indicate the external and some internal parameters of camera are proposed [5, 6, 7, 8]. Previous methods describe techniques to calibrate the camera rotational parameters if there are more two or three orthogonal vanishing points [7, 11]. Stamos and Allen [11] use high precision range data from ground view to fit them into building facade planes through camera calibration with three orthogonal vanishing points. Coorg and Teller construct a large set of 3D building models by using spherical mosaics produced from accurately calibrated ground view cameras with a GPS device [10].

1.2. Overview

In our approach, 3D building models are obtained from aerial images by using a previous approach [1, 9] (the process of obtaining these models does not affect the rest of our operations; the 3D models could also be derived from LIDAR range sensors, for example). Aerial views provide not only the global (world) position and 3D geometry of buildings in a site, but also give the roof information such as texture and multiple layer structures as shown in Figure 2. We want to extract facade information of such buildings from ground view images.

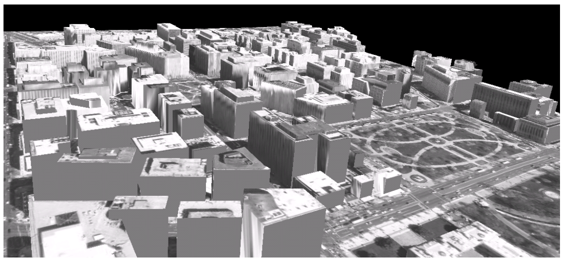


Figure 2: *3D building models of a D.C. site derived from aerial images.*

In this paper, we present an automatic method for integrating ground view images to 3D complex buildings in an urban area by using projective geometry. We first cluster lines having a common vanishing point and extract building facade regions by using spatial locality of the vertical and horizontal line clusters. Each facade gives us two vanishing points. However, since we already have 3D solid models from aerial views, the requirement of orthogonal vanishing points is not necessary in our approach. Furthermore, the roof corners of the extracted facade regions are used to estimate the position of the camera. For the automatic process of camera pose estimation, we derive multiple hypotheses of 3D to 2D correspondences, and search for the best match.

Section 2 explains how to extract facade image regions automatically by using vanishing points, followed by a pose estimation method with the extracted facade and vanishing points in the section 3. Some results and conclusion are discussed in section 4 and section 5.

2. Feature Extraction Methodology

Vanishing points from a set of parallel image lines have been shown to be useful for estimating camera parameters in previous work [5, 6, 7, 8]. Given vanishing points and a 3D building model, the world position and rotation of ground view images, can be recovered. We can exploit vanishing points, which are not necessarily orthogonal, as long as the angles between the corresponding lines can be known by referring to the 3D building model. We then recover the world position of ground view cameras by 3D to 2D surface correspondences.

2.1. Overview of vanishing point detection

Vanishing points can be extracted from a group of parallel lines in an image, which can be obtained by an automatic approach or given by a user. Many methods have used the concept of a Gaussian sphere [12, 13, 14, 15], which is one of global feature extraction methods such as the Hough transform. Shufelt [13] requires intrinsic camera parameters of images to use the orthogonality of 3D building models, which are not available directly from the uncalibrated ground view images due to possible variations of the focal length. Furthermore, Gaussian sphere approaches are applicable only for one simple structured building that has one or two surfaces, as a global feature approach does not consider the local proximity of lines in the image, *i.e.*, line clusters from multiple surfaces of buildings cannot be separated.

Other automatic vanishing point detection methods that operate in the image space instead of Gaussian sphere have been proposed [16]. Liebowitz and Zisserman [16] try to group the image lines that have the same dominant orientation. The intersection point of these lines with minimum distance error can be a vanishing point. Such methods are simple and use the spatial proximity well, but if there exist real parallel lines in the image, they fail to compute the intersection point. In addition, they are applicable only to a single surface at a time.

In this paper, we use a hybrid method of Gaussian sphere approach and image-based approach. First, each line segment is transformed into a unit vector on Gaussian sphere to get the dominant line directions. Then, The lines that belong to the chosen directions are grouped according to image proximities.

2.2. Global line clustering

A vanishing points detection technique using a Hough transform was introduced by Barnard [12], which transforms all image lines (or edges) into a Gaussian sphere space. Each image line has an interpretation plane, and the intersection of this plane with the Gaussian sphere forms a great circle as depicted in Figure 3. In the geometry of Gaussian sphere, the center of the sphere is the center of projection and its radius is normalized by camera focal length so that the metric information (such as angle) is preserved.

The Gaussian sphere is tessellated uniformly and a great circle corresponding to an image line is accumulated on the sphere, *i.e.* the strength of the line (usually its length) is accumulated into each cell around the circle. The lines on the same great circle contribute to the same group of cells. After processing all image lines, the cells located at dominant vanishing points such as horizontal or vertical ones should have high values.

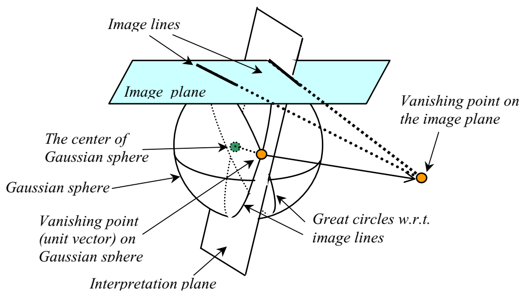


Image lines
Image plane
Vanishing point on the image plane
The center of Gaussian sphere
Gaussian sphere
Vanishing point (unit vector) on Gaussian sphere
Great circles w.r.t. image lines
Interpretation plane

Figure 3: *Gaussian sphere geometry.*

If the focal length is known, we construct a metric Gaussian sphere, so that two orthogonal vanishing points in 3D are also located orthogonally in the Gaussian sphere. Since we are interested in only the image location of the vanishing points rather than the location on the Gaussian sphere, we arbitrarily choose half of the image dimension as the distance between the centers of the Gaussian sphere and the image plane. Regardless of the distance between the image plane and the center of the Gaussian sphere, the image location of vanishing points should remain same. As illustrated in Figure 3, the image plane is moved to be adjacent to the surface of the Gaussian sphere so that the distance between the image plane and the center of the Gaussian sphere is the radius of the sphere.

By using half of the image dimension as the radius of the Gaussian sphere, we convert image coordinate vectors to Gaussian space vectors as followings:

$$g_x = x_i - r, \quad g_y = y_i - r, \quad g_z = r, \tag{1}$$

where $(x_i, y_i)^T$ is an image coordinate, $(g_x, g_y, g_z)^T$ is a Gaussian space coordinate vector, and r is the radius of the sphere. An image line has two end points, which are transformed to two Gaussian coordinate vectors and each corresponding great circle plane can be described by its normal vector:

$$n = (g_x^1 \;\; g_y^1 \;\; g_z^1)^T \times (g_x^2 \;\; g_y^2 \;\; g_z^2)^T, \tag{2}$$

where n is a normal vector of a great circle corresponding to an image line, $(g_x^1 \;\; g_y^1 \;\; g_z^1)^T$ and $(g_x^2 \;\; g_y^2 \;\; g_z^2)^T$ are Gaussian space vectors for the image line, and '$\times$' denotes a vector cross product.

To represent Gaussian sphere, we use a 'quad tree' structure to tessellate a quadrangular pyramid to form the sphere's surface. We use only a half sphere instead of the full sphere because great circles intersect each other at two points on the sphere and one of intersecting points should be on the half sphere. We first start with a quadrangular pyramid to cover a half sphere and keep tessellating each triangle of the pyramid until we get enough resolution (as shown in Figure 4, according to our experiments, five or six tessellations are sufficient)

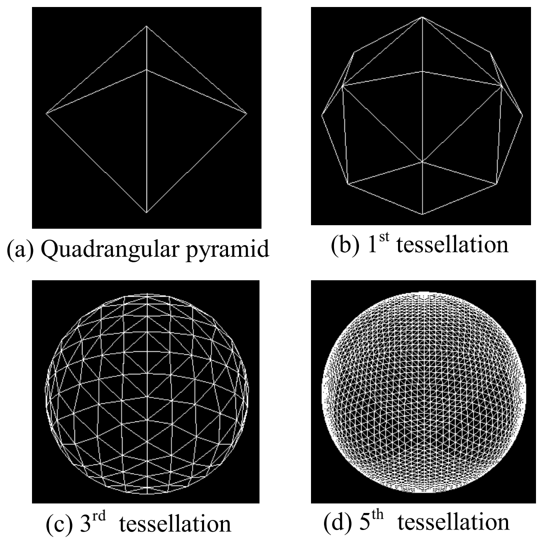


Figure 4: *Uniform tessellation of a half sphere.*

Due to the hierarchical property of a quad tree, we can save computation time by tessellating a triangle only when any great circle, *i.e.* any interpretation plane, intersects with this triangle. If not, we do not tessellate and the triangle becomes a leaf cell. In order to check the intersection of a great circle plane and a triangle plane, we make the following measurement: let m be a 3 bit mask variable, then

$$\text{great circle and a triangle have} \begin{cases} \text{No intersection, if } m = \text{'111' or } m = \text{'000'} \\ \text{Intersection, otherwise} \end{cases} \tag{3}$$

where ith bit of m: $m_i = \begin{cases} 1, & \text{if } n \cdot P_i \geq 0 \\ 0, & \text{if } n \cdot P_i < 0 \end{cases}$

n : the normal vector of a great circle

P_i : ith point(Gaussian vector) of a triangle, $1 \leq i \leq 3$

$\cdot$: vector inner product

As depicted in Figure 5, if the inner products of three

Gaussian points (vectors), P_i , of a triangle with the normal vector n all have positive or negative values, the triangle should be located above or below the great circle respectively, and this triangle becomes a leaf cell. Otherwise, this triangle intersects with the great circle and it becomes a non-leaf cell so that it can be tessellated in the next iteration.

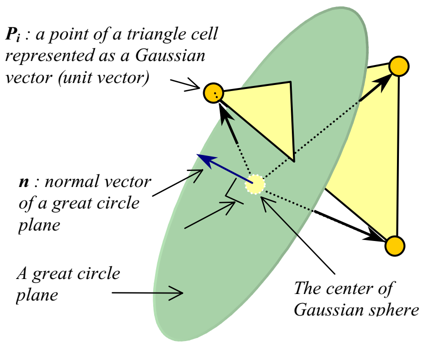

P_i : a point of a triangle cell represented as a Gaussian vector (unit vector)
n : normal vector of a great circle plane
A great circle plane
The center of Gaussian sphere

Figure 5: *Intersection between a great circle and a triangle.*

Figure 6 shows an adaptive tessellation of the sphere. We show just two lines on the Gaussian sphere to illustrate the benefit of the adaptive tessellation. Not much computation is required to go from 4th tessellation to 7th tessellation as shown in Figure 6.

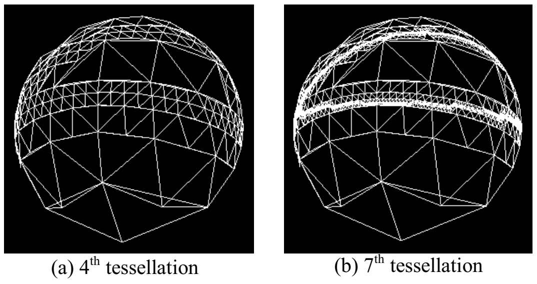


Figure 6: *Adaptive tessellation.*

If a triangle intersects with a great circle, which means that an image line passes this cell on the sphere, we accumulate the length of this line to this cell. When a non-leaf triangle is tessellated, it hands over its accumulation value to its child cells. After tessellation process is done to a sufficient resolution level, each cell has accumulation value from those lines that pass through it. Thus, vanishing points on the image plane should have large accumulation values on the Gaussian sphere. Figure 14 (see color plates) shows an example of the accumulated cells on the Gaussian sphere space: (brighter cells have higher accumulation values).

Since we use line length as an accumulating value, a pair of long horizontal and vertical lines, which are unrelated to form a vanishing point, could still create a bright noise cell where they are intersect. To avoid such noise cells, we first classify image lines into two groups based on their line directions in the image; horizontal and vertical line groups, and transform them to the separate Gaussian spheres. Horizontal group includes the image lines whose angles are close to the horizontal line (0° degree) as vertical group includes the others. Even though this separation process might not be completely accurate since the direction of image lines is variable under perspective projection, it is still effective in filtering potential noise cells. Figure 14(a) and (b) show the horizontal and vertical line groups respectively.

After the accumulation process is done, we have to search the vanishing point candidates that have significantly high accumulation value, through the Gaussian sphere surface. Many search algorithms for vanishing points on the Gaussian sphere have been developed [13, 14, 15]. Most of them exploit the orthogonality of vanishing points under the known focal length to find meaningful vanishing points, which is not applicable for our problem. Quan *et al.* [15] use a greedy approach, which selects the highest accumulation cell and removes it together with lines that contribute to that cell. To remove a line from the Gaussian sphere, its weight is subtracted from the cells along the great circle. This procedure is repeated until no more meaningful cells remain. This algorithm is simple but very effective for our application because we deal with building facades, which have only two dominant directional line groups.

The automatic vanishing points detection method based on the Gaussian sphere is summarized in Algorithm 1.

2.3. Local line clustering

We collect the image lines that are removed in the global line clustering process. Figure 15 (see color plates) illustrates the horizontal line groups from the global line clustering. The center point in Figure 15 is the location of the principal point, which is the orthocenter of the triangle formed by three orthogonal vanishing points in the image plane. The thick line in the middle of Figure 15 indicates the image location of the horizon, which connects two horizontal vanishing points. The image line of the horizon is related to the height of the camera, which is usually at a human's height in our approach. The image lines under the horizon are not meaningful in building modeling application because such lines would correspond to objects such as walking humans, cars and trees; we discard such lines.

Like other global feature extraction methods, the clustered lines may not have any spatial proximity. We need to segment the clustered lines by using the region locality to find the facade surfaces of buildings. A facade of architectural buildings consists of a horizontal and a vertical line group.

```
Algorithm 1 : Automatic Vanishing Points Detection
1. Classify image lines into two groups (horizontal and vertical groups).
2. Compute a Gaussian space vector of each line from each group by using equation (1).
3. Compute the normal vector of a great circle corresponding to each image line by using equation (2).
4. Perform the following tessellation and accumulation process:
    1) Take image lines and check any intersections with all triangles of the current tessellation by using equation (3). If intersections are found, add the line to the intersecting cell and accumulate line's weight to the cell.
    2) Label those triangles that do not intersect with any lines as leaf cells.
    3) Tessellate all non-leaf cells and hand over its accumulation value to its children cells.
    4) Repeat steps 1) through 3) until sufficient resolution acquired.
5. Find the maximum accumulating cell.
6. Mark those lines that intersect with the maximum cell.
7. Find those cells that have the marked lines in their line list and remove them and subtract the weights of the lines.
8. Repeat steps 5 through 7 until the accumulation value for the next maximum cell is less than a predefined threshold value, which is the minimum number of line weights to form a facade.
```

Since there is only one vertical line group in regular buildings, we do not use vertical line group to segment facade regions. Rather we segment facade regions by using only horizontal line groups.

The segmentation procedure with horizontal line groups is straightforward. In order to do region-based segmentation with line features, we use a voting method. We first partition the image with small square blocks whose dimension is predefined in proportion to the image dimension. We then label each block by voting result from nearby image lines. For each block, we collect the nearby lines that have one or two intersections. The line that has one intersection point has one of its end points within the intersecting block and the line that has two intersection points passes through that block. The collected lines might have different line clustering. As shown in Figure 7, two different line direction groups belong to one block. Each line votes for its line direction with its length. After voting, we label this block with the line group that has the highest voting count.

After segmentation, there exist some blocks that do not have any labels due to lack of line feature within it. These block regions can be either background (non-building object) or the regions that are located in the middle of building facade but have no lines within them. To discard background blocks and label building facade blocks, we scan the image in the y direction from top to bottom and fill unlabeled blocks that are located in between the labeled blocks.

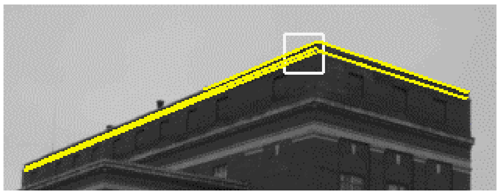


Figure 7: *An image block and lines intersected.*

The segmented regions are shown in Figure 8. The different regions in a surface of the small building in Figure 8 are generated because there is a small cubic structure in that area. No region is extracted in the leftmost part of Figure 8 because the lines of that area are not sufficient to form a building facade.

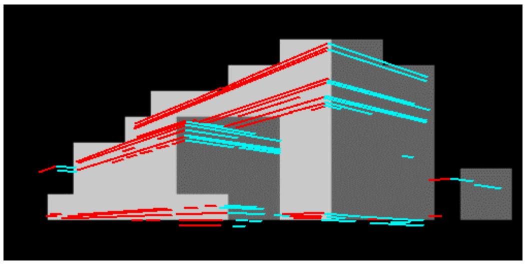
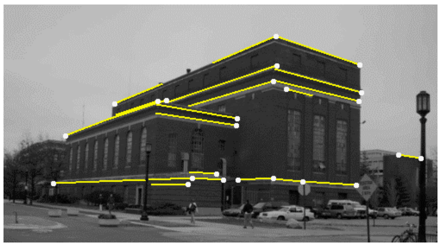


Figure 8: *Result of the segmented regions.*

2.4. Automatic generation of 2D correspondence candidates

After obtaining vanishing points and facade area in an image, we can estimate the rotational camera parameters of the ground view, as explained in the next section. We still need two 3D to 2D point correspondences to get the camera translation parameters without scale ambiguity. In this section, we explain how to obtain the 2D correspondence candidates automatically.

First we find the candidate blocks for the roof corner points. We search for the blocks that have differently labeled neighbor blocks, *i.e.* the boundary of the segmented regions. Then, we collect the blocks that have only one end point within it, *i.e.* corner point candidates. The lines that belong to the collected corner blocks indicate candidates for the roofline segments of buildings as shown in Figure 9(a). A line segment that has one or two corner point candidates, can be a 2D line segment correspondence candidate. One corner candidate block can have multiple 2D line segment correspondence candidates. Among the candidate line segments in a block, we pick the one that has the highest y coordinate value for one of its end points, as it is more likely to be a roofline segment. As depicted in Figure 9(a), some line segments are disconnected due to a poor edge detection at the beginning. We connect such segment candidates if the perpendicular distances from the end point to the other are small; Figure 9(b) shows the candidates after such processing. The evaluation of these candidates is discussed in the next section.


(a) The collected end points and lines

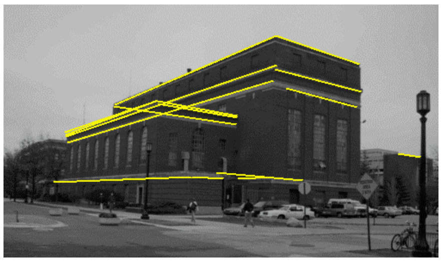

(b) The final 2D line correspondence candidates

Figure 9: *The automatically generated 2D line segment correspondence candidates.*

The automatic 2D line correspondence candidate generation method is summarized in Algorithm 2.

Algorithm 2 : Generation of 2D Line Segment Correspondence Candidates

```
1. Obtain significant vanishing points using Algorithm 1.
2. Remove the image lines below the horizon.
3. Cluster and classify the filtered image lines that have the
   same vanishing point (Global line clustering).
4. Partition the ground view image into small square
   blocks.
5. For each block, do the following:
   1) Collect image lines that intersect with this block.
   2) Vote the weight (length) of the collected lines for
      their directional class from step 3.
   3) Label this box with the class that has the largest vot-
      ing score.
6. Cluster those blocks which have the same label and are
   located nearby(Local line clustering).
7. Search for the boundary blocks of the segmented re-
   gions.
8. For each block boundary, do the following:
   1) Collect the image lines, which intersect with this
      block and have one of their end points within this
      block.
   2) Pick the line segment which has the highest y coordi-
      nate value (a roof boundary candidate) among the
      collected lines.
9. Connect disconnected candidate line segments.
```

3. Pose Estimation

In this section, we discuss how to recover the external camera parameters from the obtained vanishing points, not necessarily orthogonal, and the segmented facade surfaces. To compute the location of the ground view camera without scale factor ambiguity, we need at least two 3D to 2D correspondences. Because we already know the orientation information from the vanishing points, the candidate correspondences can be evaluated by error measurements between image edges and the predicted (projected) lines. We analyze case-by-case situations; three, two orthogonal, or three but non-orthogonal vanishing points in an image. More details may be found in Lee *et al.* [17].

3.1. Inference of orthogonal vanishing points

In the case that only one side of a building is visible, two vanishing points can be found from the sides, which are usually orthogonal. A third orthogonal vanishing point can then be recovered if we are given the principal point (or assume it to be in the center of the image), which is the orthocenter of the triangle form by three orthogonal vanishing points.

In case of non-rectangular rooftop buildings, the observed vanishing points from two different facades in the directions of say x and y are not necessarily orthogonal. However, as we know the angle between the two facades from the models computed from the aerial views, we can adjust the non-orthogonal vanishing points to estimate orthogonal ones if the position of camera center is known, as shown in Figure 10. In this figure, a newly inferred orthogonal vanishing point V_n is on the horizon line, which is the line-connecting x and y directional vanishing points, V_1 and V_2 in the image.

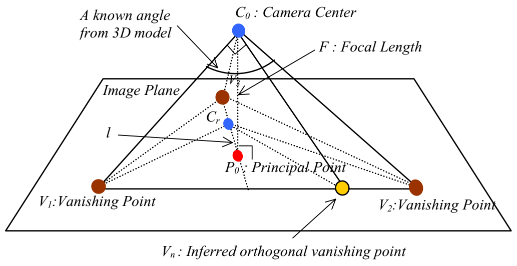


Figure 10: *Inferring a non-orthogonal vanishing point.*

Now imagine rotating the unknown camera center, C_0 to the image plane about the axis of the horizon line V_1V_2. The rotated point, say C_r, should be on the line L_3 as shown in Figure 10. The angles $V_1C_0V_2$ and $V_1C_rV_2$ are same, which are given from the 3D model. We conduct a binary search for the unknown point C_r through the line L_3 so that the angle $V_1C_rV_2$ equals the known value. Once the location of C_r is found, the new orthogonal vanishing point V_n can be located since V_n is on the horizon line and the angle $V_1C_rV_n$ is supposed to be perpendicular.

Figures 16 and 17 (see color plates) show some intermediate results for the non-orthogonal vanishing points case. As shown in Figure 16(a) and (b), there exists a non-rectangular shape roof-top building. Figure 16(c) and (d) illustrate how the 'greedy' search algorithm for vanishing points in the Gaussian sphere works in the presence of noise. Due to a significantly long image red line on the left side of the building in Figure 16(b), there are two potential vanishing points in the right side as indicated by two arrows in Figure 16(c). After removing the best cell with all intersected lines, the ambiguity disappears since the noise line is removed at the stage of the best cell removal operation as depicted in Figure 16(d). Because the three obtained vanishing points from Figure 16, are not actually orthogonal (refer Figure 16(a)), the obtained principal point is far off the center of the image as depicted in Figure 17(a). Figure 17(b) shows the correct principal point as a result of adjusting a non-orthogonal vanishing point. We can validate correctness of the inferred new orthogonal point (red point) on the horizon (white line) at the right bottom of the image by connecting another line from an adjacent building (see Figure 16(a)), which has the same directional vanishing point as the modeled building.

3.2. Estimation of camera rotation

Three orthogonal vanishing points give the camera external parameters relative to a 3D model and the internal parameters of focal length and principal point [5, 6, 7]. Since we can infer the locations of three orthogonal vanishing points regardless of the cases explained in Section 3.1, we can recover the internal parameters of the ground view camera assuming that the camera has no skew and that its aspect ratio is known.

Since three orthogonal vanishing points are the image projection of the points at infinity of 3D model coordinate vectors, the following equation can be derived:

$$[V_x\ V_y\ V_z] = M[I_x\ I_y\ I_z], \qquad (4)$$

where $I_x = (1,0,0,0)^T, I_y = (0,1,0,0)^T, I_z = (0,0,1,0)^T$, and V_x, V_y, V_z are 3×1 scaled vectors of vanishing points, and M is a projection matrix. Cipolla *et al.* [7] derive the following equation to get the external rotation matrix, R :

$$R = \begin{bmatrix} \lambda_1(u_1-u_0)/f & \lambda_2(u_2-u_0)/f & \lambda_3(u_3-u_0)/f \\ \lambda_1(v_1-v_0)/f & \lambda_2(v_2-v_0)/f & \lambda_3(v_3-v_0)/f \\ \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}, \qquad (5)$$

where $\lambda_1, \lambda_2, \lambda_3$ are scaling factors, (u_0, v_0) is the principal point, f is the focal length, and $(u_1,v_1), (u_2,v_2), (u_3,v_3)$ are x, y, and z directional vanishing points respectively. There exist an ambiguity in the signs of $\lambda_1, \lambda_2, \lambda_3$ in Cipolla *et al.* [7], which needs to be resolved.

Each column of the rotation matrix, R in Equation (5) represents a unit vector of the selected vanishing points in the camera coordinate system. Thus, each λ_i indicates the z coordinate of that vector in the camera coordinate system. Thus, when we know which model points are visible, we can infer the sign of λ_i.

In Figure 11, if a point U_i is used as a 3D correspondence, it means that a 2D correspondence of the point U_i is visible in 'Image Plane i'. In case of the 'Image Plane 1', the 3D vector corresponding to X_m directional vanishing point goes toward the negative Z_{c1} direction in the camera coordinate system, which implies a negative sign for λ_1. Signs for λ_2 and λ_3 are determined similarly with respect to Y_m and Z_m vanishing points. Therefore, the sign of each λ_i depends on the position of the computed correspondence point, which helps infer the relative location of the camera to the model origin.

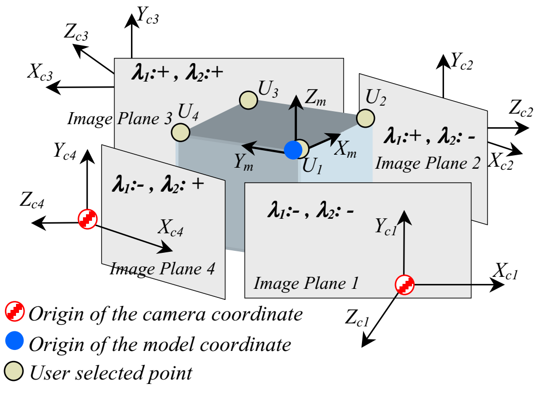


Figure 11: *Deciding the signs of* λ.

The model-to-camera rotation matrix R is recovered, the external rotation matrix, R_{wc} is derived by the following equation:

$$R_{wc} = R_{mc}R_{wm}, \qquad (6)$$

where R_{mc} is the rotation matrix R in Equation (5) and R_{wm} is the orientation of the 3D building model, computed from aerial views earlier.

3.3. Estimation of camera position

Given the external rotation matrix and a 3D to 2D point correspondence, the 3D position of the camera center must be on the 3D line, which intersects the 3D point in the model and has a directional vector from the center of projection to the corresponding 2D point. With two point correspondences, the exact position of the camera can be obtained by intersecting two lines.

3.4. Evaluation of correspondence hypotheses

When we have one 3D to 2D point correspondence between a 3D model and its 2D image, we can recover the rotational camera parameters as explained in the Section 3.2. Two point correspondences are required to estimate the camera position as explained the Section 3.3. We use one 3D to 2D line segment correspondence to get the two correspondence points. We generate 3D to 2D line segment correspondence hypotheses by considering all pairs between 2D line segment candidates (as described in the Section 2.4) and the 3D lines of a building model. Each hypothesis is evaluated as described below and the best one is selected.

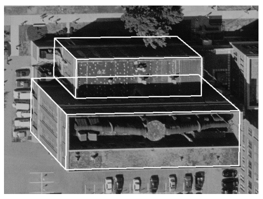
With a given correspondence hypothesis and the obtained vanishing points, a hypothesized camera projection can be generated. Under a hypothesized camera projection, we compute visible 3D points by checking intersections of the hypothesized camera rays (from the center of projection to each 3D points) and the surfaces of the 3D building model. The 3D points that belong to the surfaces having the closest intersection point with the hypothesized ray, are recognized as visible points. The set of hypothesized 3D visible points is projected to a ground view image to compute image evidence for it.

The evidence may be positive (supporting) or negative (conflicting) as provided by the underlying image features. Positive support edges are located near and parallel to the boundary of a roof hypothesis. Negative support edges are also located near the boundary of a roof hypothesis but intersect the roof boundary [1]. The line segment hypothesis providing the best score is selected as the correct 3D to 2D correspondence.

The automatic 3D to 2D line segment correspondence extraction method is summarized in Algorithm 3.

Algorithm 3 : Automatic 3D to 2D Line Segment Correspondence Extraction

```
1. Pick a 2D line from the 2D correspondence candidates
   generated in section 2.4.
2. Generate hypotheses of 3D to 2D line correspondences
   with the selected 2D line and a 3D model line out of all
   rooflines of the building.
3. Generate all possible hypotheses by repeating steps 1
   and 2 for all 2D line candidates.
4. Generate hypothesized camera models from the hypothesized
   line correspondences and the computed vanishing points by
   using the method explained in sections 3.2 and 3.3.
5. Compute 3D visible lines under the hypothesized camera model
   by using the method in section 3.4 and evaluate the
   hypothesized camera model by the following steps:
   1) Project all visible lines on the image.
   2) Find the nearby image lines which are parallel with
      the projected line.
   3) Find the nearby image lines which intersect with the
      projected line.
   4) Compute the positive and negative supporting evidence.
   5) Choose the hypothesis with the best supporting evidence.
```


(a) The given 3D building models from aerial views

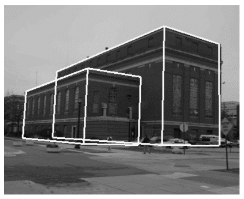

(b) The computed pose of a ground view image

Figure 12: *The reconstructed 3D pose of a ground view camera.*

4. Implementation and Experimental Results

We have tested our method with some buildings in a city and a college campus. We acquired 3D building models by using an interactive building modeling system [9]; a model is shown in Figure 12 (note that hidden lines have not been eliminated). For ground view processing, the buildings to be processed are selected by the user. The automatic system estimates the pose of the buildings in ground view images.

Figure 13 shows a result for a non-rectangular rooftop building. Since only one roof corner point is visible in the image, the user has to indicate another point to generate 2D line correspondence candidates. In this case, 2D line candidates include the user clicked point.

Finally, we show the integrated result of the ground view images and the aerial view image in Figure 18 (see color plates). A textured VRML model is generated from an aerial view, to which we integrate three ground view images of building B_1 shown in Figure 18(b). Figure 18(a) shows a rendered view from a chosen viewpoint using a standard VRML viewer. The blending of facade textures is done by the user though an automatic blending function could also be used [4]. The computed position and principal axes of three ground view cameras for the building B_1 are displayed on the bottom of the Figure 18(a). While other buildings have no or low-resolution textures from the aerial view, the building B_1 has high-resolution textures from the ground views.

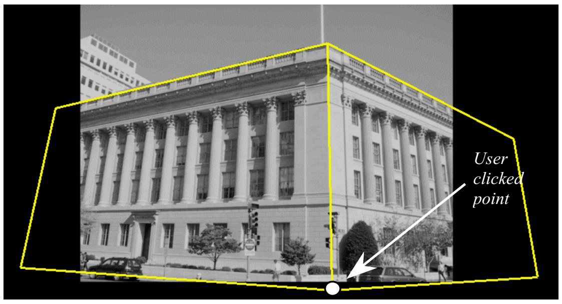


Figure 13: *A result of non-rectangular rooftop building.*

5. Conclusion

We have presented a method for automatically integrating ground view images and 3D models to obtain high resolution facade textures for 3D architectural models. We proposed a hybrid feature extraction method, which exploits a global line clustering based on the Gaussian sphere and a local line clustering based on a region-based segmentation approach. We also implemented an automatic method for generating and evaluating 3D to 2D line correspondences if at least two roof corners are visible in an image. The automatic system still requires some predefined values such as image sub-block size and the significant number of lines in Gaussian Sphere processing. In addition, we derived a novel pose estimation method for the ground view camera by using the automatically obtained vanishing points, not necessarily orthogonal, and the selected 3D to 2D line correspondence. A refinement process for 3D model parameters will be necessary not only because 3D models from aerial views are not highly accurate, but also because some facade detail structures are not visible from the aerial views.

Acknowledgements

This research was supported in part by U.S. Army STRICOM under contract N61339-02-C-0053.

References

1. S. Noronha and R. Nevatia. Detection and Modeling of Buildings from Multiple Aerial Images *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**(5):501–518, 2001.
2. Y. Hsieh, SiteCity: A Semi-Automated Site Modeling System. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 499–506, 1996.
3. A. Gruen and R. Nevatia (Editors). Special Issue on Automatic Building Extraction from Aerial Images. *Computer Vision and Image Understanding*, 1998.
4. P. E. Debevec, C. J. Taylor and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. *In Proceedings of SIGGRAPH*,11–20, 1996.
5. D. Liebowitz, A. Criminisi and A. Zisserman. Creating Architectural Models from Images. *Computer Graphics Forum*, **18**(3):39–50, 1999.
6. B. Caprile and V. Torre. Using Vanishing Points For Camera Calibration. *International Journal of Computer Vision*, 127–140, 1990.
7. R. Cipolla, T. Drummond and D.P. Robertson. Camera Calibration from Vanishing Points in Images of Architectural Scenes. *In Proceedings of British Machine Vision Conference*, **2**:382–391, 1999.
8. E. Guillou, D. Meneveaux, E. Maisel. and K. Bouatouch. Using Vanishing Points for Camera Calibration and Coarse 3D Reconstruction from A Single Image. *The Visual Computer*, **16**:396-410, 2000.
9. S. C. Lee, A. Huertas, and R. Nevatia. Modeling 3-D Complex Buildings With User Assistance. *Proceedings of IEEE Workshop on Applications of Computer Vision*, 170–177, 2000.
10. Satyan Coorg and Seth Teller. Extracting Textured Vertical Facades from Controlled Close-Range Imagery. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 625–632, 1999
11. Ioannis Stamos and Peter K. Allen. Automatic Registration of 2-D with 3-D Imagery in Urban Environments. *Proceedings of IEEE International Conference on Computer Vision*, **2**:731–736, 2001.
12. S. Barnard. Interpreting Perspective Images. *Artificial Intelligence*, **21**:435–462, 1983.
13. J. A. Shufelt. Performance Evaluation and Analysis of Vanishing Point Detection Techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(3):282–288, 1999.
14. F.A. van den Heuvel. Vanishing Point Detection for Architectural Photogrammetry. *International archives of photogrammetry and remote sensing*, **32**(5):652–659, 1998.
15. L. Quan and R. Mohr. Determining Perspective Structures Using Hierarchical Hough Transform. *Pattern Recognition Letter*, **9**:279–286, 1989.
16. D. Liebowitz and A. Zisserman. Metric Rectification for Perspective Images of Planes. *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 482–488, 1998.
17. Sung Chun Lee, Soon Ki Jung, and Ram. Nevatia. Integrating Ground and Aerial Views for Urban Site Modeling. *International Conference on Pattern Recognition*, 2002.