Extracting Textured Vertical Facades From Controlled Close-Range Imagery*

Satyan Coorg Seth Teller
Computer Graphics Group
MIT Laboratory for Computer Science
{satyan,seth}@graphics.lcs.mit.edu

Abstract

We are developing a system to extract geodetic, textured CAD models from thousands of initially uncontrolled, close-range ground and aerial images of urban scenes. Here we describe one component of the system, which operates after the imagery has been controlled or geo-referenced. This fully automatic component detects significant vertical facades in the scene, then extrudes them to meet an inferred, triangulated terrain and procedurally generated roof polygons. The algorithm then estimates for each surface a computer graphics texture, or diffuse reflectance map, from the many available observations of that surface.

We present the results of the algorithm on a complex dataset: nearly 4,000 high-resolution digital images of a small (200 meter square) office park, acquired from close range under highly varying lighting conditions, amidst significant occlusion due both to multiple inter-occluding structures and dense foliage. While the results are of less fidelity than that would be achievable by an interactive system, our algorithm is the first to be demonstrated on such a large, real-world dataset.

1 Introduction

Three-dimensional modeling of existing urban architecture has numerous applications, including virtual environments [7, 11], urban planning, military simulation, etc. It is clearly attractive to employ photographs in the modeling process: the accuracy of the 3-D model is enhanced, and the resulting environments appear visually realistic.

Automatic 3-D model recovery from photographs has long been one goal of computer vision research, leading to significant progress in designing algorithms that analyze a few images and recover 3-D structure from them. However, a drawback of most vision algorithms is their inability to scale to large sets of images, thereby limiting their applicability to 3-D modeling of extended regions. Though some recent research addresses this drawback (Section 1.2), practical methods for automatically extracting geodetic "computer graphics" 3-D models from imagery remain elusive.

An alternate approach has been to design semi-automatic modeling systems and allow a human to aid the reconstruction process (e.g., by marking and corresponding image features). Such systems yield high quality results [7], but require the user to process each image in the input dataset and each structure in the output dataset. Such systems therefore do not scale to large numbers of images or structures.

This paper presents a novel algorithm that automatically recovers 3-D information by analyzing a large set of images. The algorithm is based on three ideas:

- Each image is annotated with absolute position and orientation of the acquiring camera. Pose information provides useful geometric constraints that aid the reconstruction process. Also, pose information enables our algorithm to focus on processing only the (potentially small) portion of the data that is *relevant* to any 3-D region of interest.

- We focus here on extracting *vertical facades*, as they are common in urban scenes. This assumption yields an extraction algorithm based on a "detect-and-verify" paradigm.

- We exploit a *large number* of observations of each facade to design a simple and robust technique that synthesizes a single texture for the facade.

The output of our algorithm, a 3-D textured geometric model, can be input directly to a computer graphics rendering system.

*Funding for this research was provided in part by the Advanced Research Project Agency under contract DABT63-95-C-0009, Intel Corporation and the MIT Lincoln Laboratories.

1.1 Dataset Acquisition

Our dataset consists of nearly 4,000 images acquired from eighty-one distinct optical centers or "nodes." Apart from avoiding inclement weather and darkness, no other restriction (e.g., selection of diffuse lighting conditions) was made on the manner or time of acquisition. The image acquisition, spherical mosaicing, and exterior calibration are described elsewhere [6].

1.2 Related Work

Interactive modeling systems allow a user to identify geometric features in photographs and establish correspondences between them [2, 7]. These systems include a human to identifying features (for correspondence) and occluded pixels (for texture computation). This makes them impractical for use on large datasets, such as ours. Such systems are also difficult to assess algorithmically, as they require a "human-in-the-loop" to perform non-algorithmic tasks.

Mosaicing [4, 18, 20] seamlessly stitches together multiple images taken from the same viewpoint, for viewing from a synthetic camera constrained to lie at the same position as the acquiring camera. In our work, we use mosaics to organize multiple images to significant engineering advantage, but recover facade geometry and texture as well, providing the user full navigational freedom.

Stereo vision [8] recovers 3-D information from a few (usually two or three) pose-annotated images. This technique matches corresponding features (e.g., points or edges) across images, and locates 3-D features by triangulation. These techniques make an inherent trade-off between the inter-camera distance (baseline) and ease of matching: larger baselines allow more stable triangulation, and thus higher quality 3-D models, but matching across images from widely separated cameras is an extremely hard problem. *Multi-baseline* stereo (e.g., [12]) attempts to address this drawback by using several images simultaneously. However, in order to perform automatic matching, such algorithms require images to be taken from closely-spaced cameras under stable illumination – conditions difficult, if not impossible, to achieve in extended outdoor environments.

Structure from motion techniques [22, 19, 21, 1] recover both scene structure and camera motion by analyzing correspondences in a closely-spaced image sequence (e.g., frames from a video sequence). While these techniques correlate nearby images in the sequence, significant analysis must be performed to relate images that are farther apart.

Space-sweep techniques have been used recently [5, 15, 17] to perform matching and reconstruction from an arbitrary number of images. These world-space algorithms traverse the entire 3-D region of interest and identify likely locations of 3-D features. This paper employs a related space-sweep technique, but our algorithm handles an arbitrary number of general camera positions, works in outdoor scenes with widely varying illumination, and generates a 3-D model suited for graphics rendering.

1.3 Overview

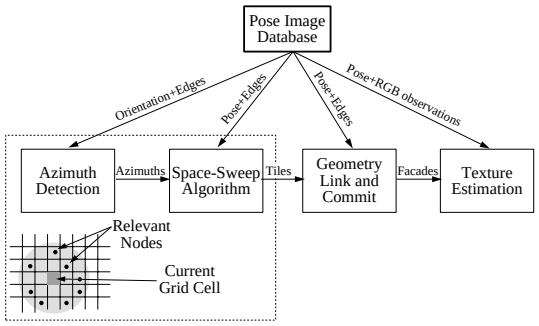


Figure 1: Overview of vertical facade extraction.

Figure 1 shows a high-level overview of our algorithm. The algorithm executes in communication with a pose image database, which stores raw data (e.g., pose and images) as well as derived data (e.g., detected edges).

First, the algorithm divides the 3-D region of interest into a 2-D XY grid based on a user-supplied grid size G. Grid-based subdivision enables restriction of subsequent processing to nodes that are *relevant* to the grid cell (e.g., within some world-space distance D). This subdivision also decouples different parts of the scene, making the reconstruction process more robust. For each grid cell, the algorithm computes *tiles* – pieces of vertical facade – in a two step process. Likely tile azimuths are detected based on a histogramming technique (Section 2). These are verified and located using a space-sweep technique (Section 3), populating the grid cell with only those 3-D tiles that are supported by sufficient image evidence.

Second, the tile geometry recovered is linked to form complete facades; and many spurious facades in the model are removed by a *facade commitment* process (Section 4). Textures for the generated vertical facade geometry are computed from multiple observations (Section 5).

We report results of the algorithm on our dataset in Section 6 and conclude in Section 7.

2 Azimuth Detection

This section describes a histogramming algorithm to identify likely azimuths of tiles, using horizontal line segments. Such line segments arise often in urban environments, e.g., from windows and facade boundaries.

2.1 Estimating Tile Azimuths

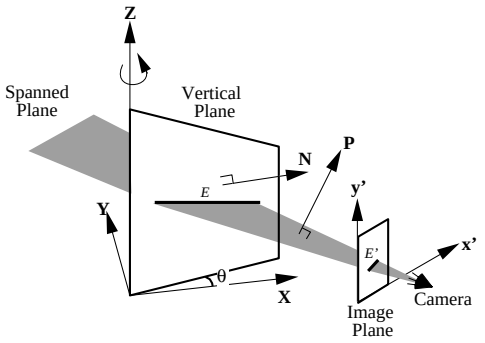


Figure 2: Deducing 3D tile azimuth from a 2D edge.

We estimate tile azimuths as follows. When pose information (specifically, orientation) is known, the direction of a horizontal edge is completely determined by the 2-D line equation of its projection (Figure 2). Let a 3-D horizontal line segment E on a tile project to a 2-D edge E' in some pose image. In the figure, the vertical plane through E makes an angle θ with the $\mathbf{X}$ axis, i.e., its *azimuth* is θ. The normal $\mathbf{N}$ to this plane is $[\sin\theta, -\cos\theta, 0]^T$. Let $\mathbf{P} = [p_x, p_y, p_z]^T$ be the normal to the plane spanned by E' and the camera. $\mathbf{P}$ (in global coordinates) is determined by the orientation of the camera (a 3×3 rotation matrix $\mathbf{R}$) and the 2-D image-space line equation $ax' + by' + c = 0$ of E' (where $a^2 + b^2 + c^2 = 1$):

$$\mathbf{P} = \mathbf{R}^{-1} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Then, the direction of E is given by:

$$\mathbf{P} \times \mathbf{N} = [p_z \cos\theta, -p_z \sin\theta, (-p_x \cos\theta - p_y \sin\theta)]$$

As E is horizontal, the z component vanishes. This yields two solutions for the orientation (when either $p_x \neq 0$ or $p_y \neq 0$):

$$\theta = \begin{cases} \tan^{-1} \frac{-p_x}{p_y} & \text{or} \\ \tan^{-1} \frac{-p_x}{p_y} + \pi \end{cases} \quad (1)$$

Of these, the correct azimuth is the one that faces the viewing direction. If $p_x = 0$ and $p_y = 0$, the camera has observed a horizontal line segment at the camera height, and no information can be determined. As most of the nodes in our dataset are taken from positions near ground level, this case arises very rarely.

Figure 3 shows the results of applying this technique to two nodes. In this figure, vertical edges were first identified by thresholding ($p_z < 0.01$). For the rest, an azimuth θ is estimated using Equation 1. Note that truly horizontal edges from the same facade are assigned the same azimuth, both within and across nodes. This property does not hold for edges which are not horizontal in world space; these tend to be uncorrelated both within a node and across different nodes. A similar technique has been used in photogrammetry to generate buildings from *monocular* (i.e., single) aerial images [14].

2.2 Histogramming Azimuths

Likely azimuths are identified through the following idea: 2-D edges arising from truly (i.e., world-space) horizontal line segments will be assigned identical azimuths in different nodes, whereas azimuths of non-horizontal edges will vary with node position. Thus histogramming reinforces true azimuths; the rest tend to be unsupported, as they are uncorrelated across nodes.

The following algorithm reports a set of likely tile azimuths in a grid cell C, from edges in C's relevant nodes $1 \ldots k$:

Azimuths $A = \phi$.
for node $i \in 1 \ldots k$ **do**
  Let C project to image-space region C' in node i.
  Find azimuth of each non-vertical edge in C'.
  Histogram azimuths (weighted by edge length)[1].
  Identify the most populated bucket and add its representative azimuth (e.g., median) to A.
endfor
Histogram A and report each bucket that contains at least three nodes.

Informally, the algorithm picks a dominant azimuth from each node, then reports azimuths that are supported by several nodes. These azimuths are then verified by the space-sweep algorithm described in the next section.

3 The Space-Sweep Algorithm

The space-sweep algorithm to locate and verify tiles is based on an *incidence counting* idea, related to that proposed by Collins [5]. If any *sparse* set of features in several nodes is projected into 3-D, regions with high *incidence* of such projections correspond to likely locations of 3-D features, for the following reasons. If a 3-D

[1] We use overlapping θ buckets of size 3 degrees.

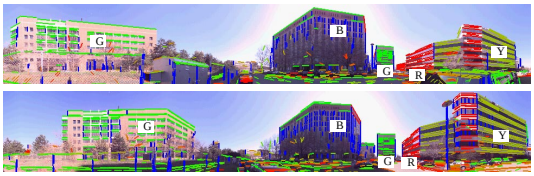
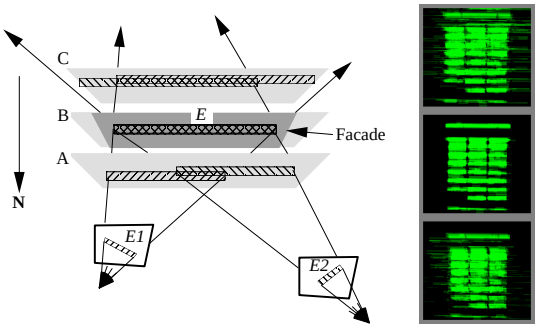


Figure 3: This figure shows two nodes (four faces of a cubical environment map) along with their (long) edges. Vertical edges are colored *blue*, and other edges are colored with (absolute values of) the tile normal derived from their azimuth (e.g., *red* is $[1,0,0]^T$, *green* is $[0,1,0]^T$, etc.).

feature is present in multiple nodes, then projections through the corresponding 2-D features pass near it, increasing its incidence count. Conversely, since the set of features are sparse, it is rare that unrelated projections pass through the same 3-D region by chance.

Given an azimuth θ, the sparse features in our algorithm are edges likely to lie on tiles with azimuth θ. Note that identifying such edges in each node is straightforward; they are those edges whose computed azimuth is θ (Figure 3).



(l) (r)

Figure 4: At left, three positions A, B, and C of a vertical plane. At right, correlation among projected line segments is indicated by brightness.

Using such edges, Figure 4 shows the application of incidence counting to tile location. Part (l) shows three planes with common normal **N** and different offsets, with projected (and blurred) edges $E1$ and $E2$ from two nodes. Note that correlation between horizontal line segments is greatest at the position of the plane corresponding to the tile location. We use a correlation function defined in Section 3.1, and a space-sweep algorithm described in Section 3.2 that locates tiles using the maxima of this function.

3.1 Correlation Function

Given a plane and its associated horizontal line segments, the correlation function computes a measure of the extent of overlap between different line segments on the plane. Each line segment L is blurred into a *rectangle* of half-width σ_L to alleviate small errors in camera pose and smooth the peaks of the correlation function[2].

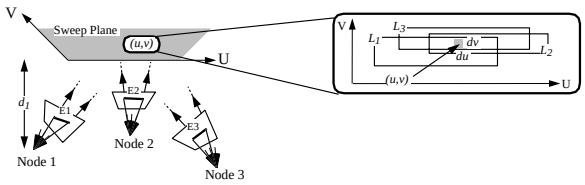
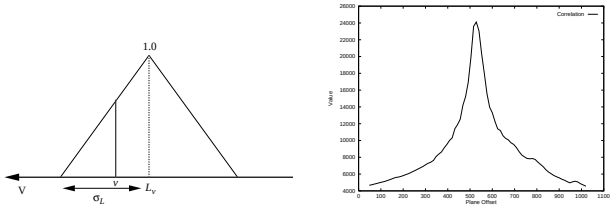


Figure 5: Parameters in the correlation function. The figure on the left shows edges $E1$, $E2$, and $E3$ from three nodes projected onto a plane. The figure on the right is a blowup of a small section of the plane. It depicts overlap of the rectangles L_1, L_2, and L_3 corresponding to the edges.

$$O = \int\int \sum_{i=1}^{k} w_i(u,v) \sum_{i=1}^{k} \frac{1}{d_i^2} du dv \qquad (2)$$

Consider the function defined in Equation 2 (see also Figure 5). In this equation, u and v range over the plane. The point (u,v) is inside rectangles $L_1 \ldots L_k$ arising from nodes $1 \ldots k$; the perpendicular distance from the plane to node i is given by d_i; and $w_i(u,v)$ is the weight contributed by L_i to plane element (u,v). In our implementation, we use a triangular filter for $w_i(u,v)$, as it is both efficient to evaluate and possesses a well-defined peak (Figure 6-a):

$$w_i(u,v) = \max(1 - \frac{|v - L_v|}{\sigma_L}, 0)$$



(a) (b)

Figure 6: Edge weights (a); Correlation (b).

[2]We use $\sigma_l = 0.01d$, where d is the perpendicular distance between the node generating l and the vertical plane.

The correlation function favors overlap between line segments arising from different nodes: "cross-terms" w_i/d_j^2 such that $i \neq j$ arise when L_i overlaps with L_j, and contribute to O. An example is shown in Figure 6-b, in which Equation 2 was evaluated for different plane offsets using the line segments shown in Figure 4-(b). Due to the combined effect of many line segments, the function has a well-defined peak at the facade's location.

The function is unbiased toward planes of larger area. Instead, the function downweights the area of each plane-element (u, v) by the squared-distance from the source node. This is advantageous, as such a bias would favor tiles farther away from the nodes. In effect, the function measures correlation in each node's image-space, and aggregates correlation to yield the total value.

Finally, the correlation function can be evaluated efficiently by exploiting the rectangular geometric structure inherent in Equation 2. The evaluation technique is a straightforward modification of a segment-tree based plane-sweep algorithm that computes the total area of m rectangles in optimal $O(m \log m)$ time [16].

3.2 Maxima Location; Tile Generation

Given a grid cell C, the azimuth θ of a tile, and a set of nodes $1 \ldots k$, the space-sweep algorithm locates tiles as follows:

```
// Project edges from each node onto canonical planes
for node i ∈ 1...k do
    Let C project to image-space region C' in node i.
    Let P_i be a plane with azimuth θ, 1 unit from node i.
    Project each edge in C' with azimuth θ onto P_i.
endfor
// Sweep plane through C locating and generating tiles
for offset d in C (with step size S) do
    Let plane P = (θ, d).
    Reproject all edges in P_1 ... P_k to P.
    // Identify high-incidence regions and
    // tiles corresponding to local maxima
    if O(d) > O(d + S) and O(d) > O(d − S) then
        Identify regions with incidence > K in P,
        and their corresponding node edges.
        Extrude high-incidence regions vertically to
        a ground plane, producing tile rectangles.
    endif
endfor
```

In the first phase, the algorithm identifies edges in each node that are likely to lie on the tile by comparing their azimuths with θ. For efficiency, such edges are projected to a canonical plane $\mathbf{P}_i$ (with azimuth θ) corresponding to node i. This intermediate projection permits a simple transformation – a scaling plus a 2-D translation – to be used to construct horizontal line segments on any other plane $\mathbf{P}$ with the same azimuth.

Next, the algorithm discretizes, using step size S, the set of all possible plane offsets corresponding to grid cell C. At each plane offset, it computes horizontal line segment positions and evaluates the correlation function. For each local maximum of the correlation function, it identifies regions on the plane that correspond to a tile by thresholding on incidence K, i.e., regions on the plane (if any) that overlap more than K rectangles (weighted using the triangular filter). In addition, it identifies node edges that *support* (i.e., give rise to) the tile, by thresholding on the extent of overlap[3] with identified 3-D regions on the tile. Finally, the 3-D region information generated is converted to rectangles by extruding them onto a *ground plane*, and combining overlapping rectangles to produce tiles. The ground plane is estimated by using the z values of the camera positions.

The complexity of this algorithm for a single grid cell is $O(\frac{G}{S}ke(\log k + \log e))$, where G is the grid size, S is the step size, k is the number of nodes, and e is number of edges of a node that lie in the cell's projection.

4 Geometry Link and Commit

The space-sweep algorithm of the previous section populates a set of grid cells with tiles likely to exist in the 3-D world. This section describes two techniques to enhance the quality of the generated 3-D model.

4.1 Facade Commitment

This section describes a technique to eliminate spurious facades present in the recovered 3-D model. Our technique is based on the idea of facade *commitment*, which enforces the following constraint: a facade committed to the model *precludes* edges that gave rise to it from supporting other facades. Such enforcement can result in the removal of other facades, if the number of observations supporting them falls below the incidence threshold K of Section 3.

Figure 7 shows (in 2-D) an example of how a spurious facade might arise from three real facades B, C, and D with the same normal $\mathbf{N}$. Such spurious facades can arise from a single facade as well, due to self-interaction of repeated texture on the facade.

Our spurious facade elimination technique consists of the following steps. First, the algorithm orders all facades using some criterion that favors real facades over spurious facades. Facades are then committed to

[3] We use a threshold of $0.8l$ for an edge with length l, i.e., if more than 80% of the edge overlaps with high incidence regions identified on the tile.

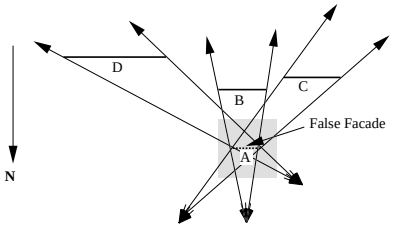


Figure 7: A spurious facade can result from interaction between unrelated facades with the same azimuth.

the model in this order. Edges giving rise to facades earlier in the ordering are removed from later facades. For example, in Figure 7, if the ordering favors either B, C, or D over A, it would preclude at least one of A's observations from supporting it, resulting in A's removal.

We considered two possible facade orderings:

1. Order facades according to *occlusion*, as in [17]. This ordering favors locations closer to the nodes, and tends to select spurious facades.

2. Order facades according to *higher reported observations* (measured by total length of horizontal segments on the facade). This ordering heuristic favors facades that are larger in area and/or have more line segments.

In practice, we have observed that the first ordering tends to break up facades into several pieces before their actual location, whereas the second, by favoring larger facades, tends to retain real facades. The results in Section 6 are computed using the second method.

5 Texture Estimation

We now describe an algorithm to synthesize a single (diffuse RGB) texture for each facade. Note that the set of nodes that observe the facade are known, having been reported by the facade extraction algorithm.

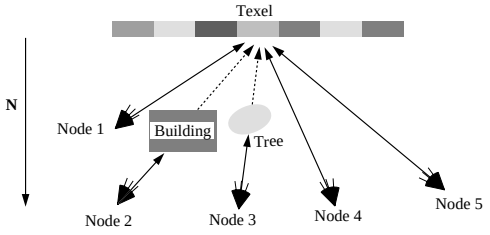


Figure 8: The problem of texture estimation.

Figure 8 illustrates the challenges inherent in texture estimation. Each observation reports different colors for a texel on the facade, due to occlusion, obliqueness, illumination variation, etc. Figure 9 shows these effects for four *rectified* facade images. Significant variations in pixel color due to changes in illumination and the effects of shadows and reflections, and aocclusion, make it difficult to determine (and use) a single "best" node, or even interpolate between various nodes based on viewing direction [7]. Instead, using median statistics, we combine the information present in all relevant nodes to compute a single facade texture.

Figure 9: Relevant portions of four nodes, auto-rectified to a facade.

5.1 Median Texture Computation

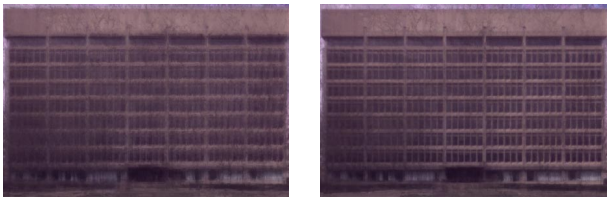
Our technique divides the facade into a (area dependent) number of texels, and estimates a single diffuse value for each texel. Since raw RGB color of the same texel can vary significantly in each node, we use a color space where at least some of the components are stable under different lighting conditions. The CIE xyY color representation [9] decouples chromaticity x, y from luminance Y. Under illumination by (predominantly) white sunlight, the luminance Y of a texel can vary significantly, but its chromaticity remains reasonably stable.

We compute the *median*[4] x and y value for each texel, as it is less sensitive to outliers than is simple averaging [10]. This is important in our application as outliers are common, typically arising from occlusion by foliage or other structures. Fortunately, such outliers do not appear at the same texel on different rectified nodes, due to parallax (e.g., [13]). We also compute a median luminance value for each texel. Even though luminance values vary significantly, we

[4]With values weighted by $\mathbf{N} \cdot \mathbf{V}$, where $\mathbf{N}$ is the facade's normal, and $\mathbf{V}$ is the vector direction from the texel center to the node. This down-weights nodes that view the facade obliquely.

have observed that the median luminance reasonably reflects the luminance of the texel under "average" lighting conditions. In any event, the differences in illumination are normalized away, so that the generated texture can be subjected to different simulated lighting conditions.

Figure 10-(a) shows the results of weighted median texture estimation. Note the automatic removal of most occlusion from the texture; the luminance pattern on the texture is also reasonably approximated. Compared to Figure 9, the median texture has less occlusion, fewer changes in luminance across the texture (e.g., due to shadows), and fewer view-dependent effects (e.g., reflection). The texture has inaccuracies arising from small errors in camera pose, unmodeled facade relief, and unmodeled unocclusion. We reduce the first of these effects by "sharpening" the texture by estimating an 8-parameter warp (as in [18]) that achieves a higher correlation between each source image and the median texture.



(a) (b)

Figure 10: (a) Median texture. (b) After sharpening.

Figure 10-(b) shows the result of sharpening. Note the significant improvement in the quality of the texture. Even though the color of each texel is computed independently, straight lines on the facade are clearly demarcated. It is possible for some blurring to persist in the texture, due to the non-planarity of the underlying building surface. One possible solution would be to construct disparity maps to capture such extrusions, as in [7].

6 Results

We have implemented the algorithm (and associated visualization) described in this paper in about 5000 lines of C++ code. The algorithm extracted all significant vertical facades in the office complex, as well as several neighboring facades. Details on the extraction process and algorithm execution times are provided below.

The facade extraction algorithm considered edges detected on six faces of a cubical environment map representing a node. Each face of the environment map was generated at 1024×1024 resolution by resampling the input images. Edge pixels were detected using the Canny edge detector [3] and converted to line segments by linking pixels with similar gradient orientation. Approximately 1000 edges were computed for each cube face (ignoring edges less than 10 pixels in length).

Area of 3-D region of interest	$\sim 500m \times 500m$
Grid Size G	$\sim 10m$
Far Distance D	$\sim 100m$
Step Size S	$\sim 0.1m$
Minimum Weighted Incidence K	3.0

Table 1: Parameters supplied to the extraction algorithm. All lengths are given in meters.

Some of the important parameters supplied to the algorithm are listed in Table 1. The grid size G should be approximately equal to the size of the smallest facade, to avoid interaction between different facades during tile reconstruction. We use a grid size of 10 meters for this dataset. The minimum weighted incidence value of 3.0 usually implies that a facade must be observed by at least five or six nodes to be successfully extracted.

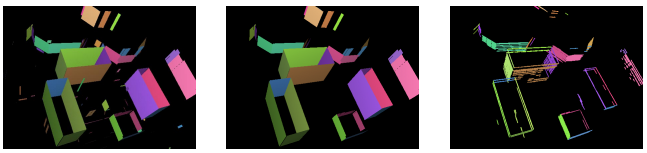
The facade extraction algorithm took about seven hours on an SGI O2 workstation with one R10000 processor, most of which was spent in the space-sweep algorithm. The space-sweep algorithm recovered approximately 2000 tiles. The model consists of about 140 facades after tile linking and facade commitment (Figure 11-(a)). After removing facades with area less than $100m^2$, the model consists of about 40 facades (Figure 11-(b)). The horizontal lines used to generate this model are shown in Figure 11-(c). Figure 13 shows the extracted facade geometry in wireframe, colocated with the input data (shown as texture-mapped spheres).

The recovered scene geometry was further processed using the following steps:

- A 2-D Delaunay triangulation of the camera XY positions was computed, and an approximation to the ground terrain was constructed using ground heights obtained from camera poses.
- Facades with end-points in the same grid cell were linked, modifying facade boundaries to the edge formed by intersecting adjacent facades. Also, a "roof" was added to each connected set of facades after extruding facade heights to the maximum height in the set.

Textures were computed to $0.1m$ resolution, with the

largest texture containing 1024×512 pixels. Texture computation took about ten hours. Textures for non-vertical geometry (i.e., roof and ground polygons) were extracted from a single aerial image of the complex, for which the exterior orientation was manually deduced.



(a) (b) (c)

Figure 11: (a) Recovered vertical facades. (b) After removal of small facades. (c) Horizontal edges that generated this model.

Figure 12: An aerial view of the final model.

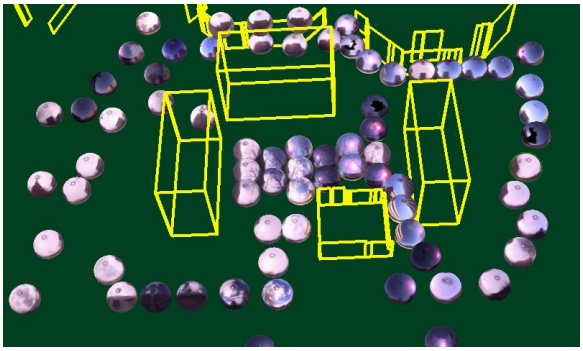


Figure 13: Pose imagery; extracted vertical facades.

7 Conclusion

This paper described an algorithm that extracts urban vertical by detecting likely vertical facade orientations from horizontal edges, and locating these using space-sweep. In addition, we described a simple and robust technique for computing a texture-map for each recovered facade. We presented results on a large pose image dataset consisting of over four thousand images. To our knowledge, ours is the first system to automatically analyze such a large set of images, and produce a realistic 3-D model suitable for computer graphics rendering.

Our algorithm has other desirable properties for automatic model extraction. It uses all information from relevant images, yet scales to an arbitrary number of images and model size. It exploits geometric constraints inherent in the 3-D environment. Finally, it is robust with respect to occlusion and changes in illumination. We believe that these properties will be crucial for future systems that perform practical, large-scale reconstruction.

Clearly our algorithm has many significant limitations, for example its unsophisticated handling of lighting and facade relief. However, we have chosen to emphasize end-to-end system architecture and scaling properties for the time being. Later each of the specific techniques for geometry and texture estimation will be elaborated.

References

[1] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, June 1999. to appear.

[2] S. Becker and V. M. Bove. Semiautomatic 3-D model extraction from uncalibrated 2-D camera views. In *Proceedings of Visual Data Exploration and Analysis II, SPIE Vol. 2410*, pages 447–461, 1995.

[3] F. J. Canny. A computational approach to edge detection. *IEEE Trans PAMI*, 8(6):679–698, 1986.

[4] S. E. Chen. Quicktime VR – an image-based approach to virtual environment navigation. In *SIGGRAPH '95 Conference Proceedings*, pages 29–38, Aug. 1995.

[5] R. Collins. A space-sweep approach to true multi-image matching. In *CVPR96*, pages 358–363, 1996.

[6] S. Coorg, N. Master, and S. Teller. Acquisition of a large pose-mosaic dataset. In *CVPR '98*, pages 872–878, 1998.

[7] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96 Conference Proceedings*, pages 11–20, Aug. 1996.

[8] U. R. Dhond and J. K. Aggarwal. Structure from stereo – A review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, Nov.-Dec. 1989.

[9] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1990.

[10] P. J. Huber. *Robust Statistics*. Wiley, 1981.

[11] W. Jepson, R. Liggett, and S. Friedman. Virtual modeling of urban environments. *PRESENCE*, 5.1, March 1996.

[12] S. B. Kang and R. Szeliski. 3-D scene recovery using omnidirectional multibaseline stereo. In *International Conference on Computer Vision and Pattern Recognition*, pages 364–370, San Francisco, CA, June 1996.

[13] R. Kumar, P. Anandan, and K. Hanna. Shape recovery from multiple views: a parallax based approach. In *ARPA Image Understanding Workshop, Monterey, CA*, Nov. 1994.

[14] D. M. McKeown, C. McGlone, S. D. Cochran, Y. C. Hsieh, M. Roux, and J. Shufelt. Automatic cartographic feature extraction using photogrammetric principles. In *Digital Photogrammetry*, pages 195–212. ASPRS, 1997.

[15] H. P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Robotics Institute, CMU, 1996.

[16] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, 1985.

[17] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR97*, pages 1067–1073, 1997.

[18] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, Mar. 1996.

[19] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using non-linear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

[20] R. Szeliski and H. Shum. Creating full-view panoramic mosaics and texture-mapped 3D models. In *SIGGRAPH '97 Conference Proceedings*, pages 251–258, Aug. 1997.

[21] C. J. Taylor and D. J. Kriegman. Structure and motion from line segments in multiple images. *PAMI*, 17(11):1021–1032, November 1995.

[22] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:137–154, 1992.