Image-based Street-side City Modeling

Jianxiong Xiao Tian Fang Peng Zhao Maxime Lhuillier* Long Quan

The Hong Kong University of Science and Technology *LASMEA - Université Blaise Pascal

Figure 1: *Two close-ups of the parts 1 and 2 of a modeled city area shown in the first two rows. All the models are automatically generated from input images, exemplified by the bottom row. The close-up of the part 3 is shown in Figure 15.*

Abstract

We propose an automatic approach to generate street-side 3D photo-realistic models from images captured along the streets at ground level. We first develop a multi-view semantic segmentation method that recognizes and segments each image at pixel level into semantically meaningful areas, each labeled with a specific object class, such as building, sky, ground, vegetation and car. A partition scheme is then introduced to separate buildings into independent blocks using the major line structures of the scene. Finally, for each block, we propose an inverse patch-based orthographic composition and structure analysis method for façade modeling that efficiently regularizes the noisy and missing reconstructed 3D data. Our system has the distinct advantage of producing visually compelling results by imposing strong priors of building regularity. We demonstrate the fully automatic system on a typical city example to validate our methodology.

Keywords: Image-based modeling, street view, street-side, building modeling, façade modeling, city modeling, 3D reconstruction.

1 Introduction

Current models of cities are often obtained from aerial images as demonstrated by Google Earth and Microsoft Virtual Earth 3D platforms. However, these methods using aerial images cannot produce photo-realistic models at ground level. As a transition solution, Google Street-View, Microsoft Live Street-Side and the like display the captured 2D panorama-like images with fixed view-points. Obviously, it is insufficient for applications that require true 3D photo-realistic models to enable user interactions with 3D environment. Researchers have proposed many methods to generate 3D models from images. Unfortunately, the interactive methods [Debevec et al. 1996; Müller et al. 2007; Xiao et al. 2008; Sinha et al. 2008] typically require significant user interactions, which cannot be easily deployed in large-scale modeling tasks; the automatic methods [Pollefeys et al. 2008; Cornelis et al. 2008; Werner and Zisserman 2002] focused on the early stages of the modeling pipeline, and have not yet been able to produce regular mesh for buildings.

1.1 Related work

There is a large literature on image-based city modeling. We review several studies according to the input images and the user interaction without being exhaustive.

Single-view methods. Oh et al. [2001] presented an interactive system to create models from a single image by manually assigning the depth based on a painting metaphor. Müller et al. [2007] relied

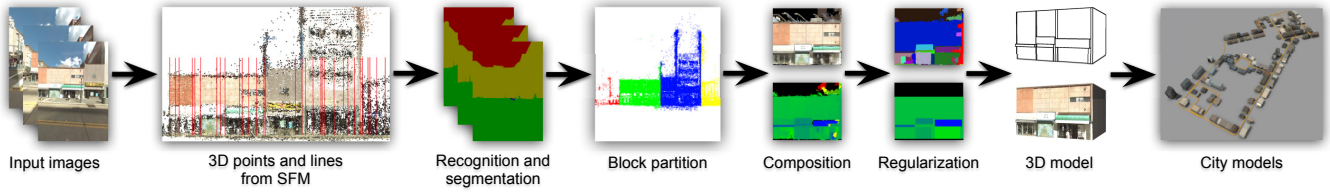


Figure 2: *Overview of our automatic street-side modeling approach.*

on repetitive patterns to discover façade structures, and obtained depth from manual input. Generally, these methods need intensive user interactions to produce visual pleasing results. Hoiem et al. [2005] proposed surface layout estimation for modeling. Saxena et al. [2009] learned the mapping information between image features and depth directly. Barinova et al. [2008] made used of manhattan structure for man-made building to divide the model fitting problem into chain graph inference. However, these approaches can only produce a rough shape for modeling objects without lots of details.

Interactive multi-view methods. Façade, developed by Debevec et al. [1996], is a seminal work in this category. They used line segment features in images and polyhedral blocks as 3D primitives to interactively register images and to reconstruct blocks with view-dependent texture mapping. However, the required manual selection of features and correspondences in different views is tedious, which makes it difficult to be scaled up when the number of images grows. Van den Hengel et al. [2007] used a sketching approach in one or more images to model a general object. But it is difficult to use this approach for detail modeling even with intensive interaction. Xiao et al. [2008] proposed to approximate orthographic image by fronto-parallel reference image for each façade during automatic detail reconstruction and interactive user refinement. Therefore, their approach requires an accurate initialization and boundary for each façade as input, probably manually specified by the user. Sinha et al. [2008] used registered multiple views and extracted the major directions by vanishing points. The significant user interactions required by these two methods for good results make them difficult to adopt in large-scale city modeling applications.

Automatic multi-view methods. Dick et al. [2004] developed a 3D modeling architectural modeling method for short image sequences. The user is required to provide intensive architectural rules for the Bayesian inference. Many researchers realized the importance of line features in man-made scenes. Werner and Zisserman [2002] used line segments for building reconstruction from registered images by sparse points. Schindler et al. [2006] proposed the use of line features for both structure from motion and modeling. However, line features tend to be sparse and geometrically less stable than points. In our work, reconstructed 3D lines are used to partition a reconstructed sequences into independent building blocks which will be individually modeled and to align building blocks regularly. A more systematic approach to modeling urban environments using video cameras has been investigated by several teams [Pollefeys et al. 2008; Cornelis et al. 2008]. They have been very successful in developing real-time video registration and focused on the global reconstruction of dense stereo results from the registered images. Our approach does not try to obtain dense stereo reconstruction, but focuses on the identification and modeling of buildings from the semi-dense reconstructions. Finally, the work by Zebedin et al. [2006] is representative of city modeling from aerial images, which is complementary to our approach from street-level images.

1.2 Overview

We propose in this paper an automatic approach to reconstruct 3D models of buildings and façades from street-side images. The image sequence is reconstructed using a structure from motion algorithm to produce a set of semi-dense points and camera poses.

Approach From the reconstructed sequence of the input images, there are three major stages during city modeling. First, in Section 3, each input image is segmented per pixel by a supervised learning method into semantically meaningful regions labeled as building, sky, ground, vegetation and car. The classified pixels are then optimized across multiple registered views to produce a coherent semantic segmentation. Then, in Section 4, the whole sequence is partitioned into building blocks that can be modeled independently. The coordinate frame is further aligned with the major orthogonal directions of each block. Finally, in Section 5, we propose an inverse orthographic composition and shape-based analysis method that efficiently regularizes the missing and noisy 3D data with strong architectural priors.

Interdependency Each step can provide very helpful information for later steps. The semantic segmentation in Section 3 can help the removal of line segments that are out of recognized building regions in Section 4. And it can identify the occluding regions of the façade by filtering out non-building 3D points for inverse orthographic depth and texture composition in Section 5.1. After the semantic segmentation results are mapped from the input image space to the orthographic space in Section 5.2, they are the most important cues for boundary regularization, especially the upper boundary optimization in Section 5.4. When the model is produced, the texture re-fetching from input images in Section 6.2 can also use the segmented regions to filter out occluding objects. On the other hand, the block partition in Section 4 provides us a way to divide the data into block level for further process, and also gives accuracy boundaries for the façades.

Assumption For the city and input images, our approach only assume that building facades have two major directions, vertical and horizontal, which are true for most buildings except some special landmarks. We utilize generic features from bank filters to train a recognizer from examples and optimized in multi-view to recognize sky regions without blue sky assumption. The final boundary between the sky and the buildings is robustly regularized by optimization. Buildings may be attached together or separated for a long distance, as the semantic segmentation can indicate the presence and absence of buildings. It is unnecessary to assume that buildings are perpendicular to the ground plane in our approach, as buildings are automatically aligned to the reconstructed vertical line direction.

Contribution Our approach contributes to the state-of-the-art in the following ways. The first is a supervised multi-view semantic segmentation that recognizes and segments each input street-side

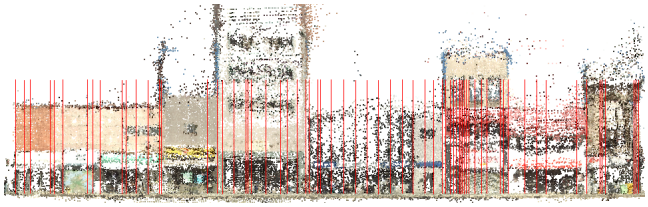


Figure 3: *Preprocessing. Reconstructed 3D points and vertical lines (red).*

image into areas according to different object classes of interest. The second is a systematic partition scheme to separate buildings into independent blocks using the major man-made line structures of the scene. The third is a façade structure analysis and modeling method to produce visually pleasing building models automatically. The last is a robust automatic system assembled from all these reliable components.

2 Preprocessing

The street-side images are captured by a camera mounted on a moving vehicle along the street and facing the building façades. The vehicle is equipped with GPS/INS (Inertial Navigation System) that is calibrated with the camera system.

Points The structure from motion for a sequence of images is now standard [Hartley and Zisserman 2004]. We use a semi-dense structure from motion [Lhuillier and Quan 2005] in our current implementation to automatically compute semi-dense point clouds and camera positions. The advantage of the quasi-dense approach is that it provides a sufficient density of points that are globally and optimally triangulated in a bundle-like approach. The availability of pose data from GPS/INS per view further improves the robustness of structure from motion and facilitates the large-scale modeling task. In the remainder of the paper, we assume that a reconstructed sequence is a set of semi-dense reconstructed 3D points and a set of input images with registered camera poses.

Lines Canny edge detection [Canny 1986] is performed on each image, and connected edge points are linked together to form line segments. We then identify two groups of the line segments: vertical line segments and horizontal ones. The grouping [Sinha et al. 2008] is carried out by checking whether they go through the common vanishing point using a RANSAC method. Since we have a semi-dense point matching information between each pair of images from the previous computation of SFM, the matching of the detected line segments can be obtained. The pair-wise matching of line segments is then extended to the whole sequence. As the camera is moving laterally on the ground, it is difficult to reconstruct the horizontal lines in 3D space due to lack of the horizontal parallax. Therefore, we only reconstruct vertical lines which can be tracked over more than three views. Finally, we keep the 3D vertical lines whose directions are consistent with each other inside RANSAC framework, and remove other outlier vertical lines.

3 Building segmentation

For a reconstructed sequence of images, we are interested in recognizing and segmenting the building regions from all images. First, in Section 3.1, we train discriminative classifiers to learn the mapping from features to object class. Then, in Section 3.2, multiple view information is used to improve the segmentation accuracy and consistency.

3.1 Supervised class recognition

We first train a pixel-level classifier from a labeled image database to recognize and distinguish five object classes, including building, sky, ground, vegetation and car.

Feature To characterize the image feature, we use textons which have been proved to be effective in categorizing materials and general object classes [Winn et al. 2005]. A 17-dimensional filter-bank, including 3 Gaussians, 4 Laplacian of Gaussians (LoG) and 4 first order derivatives of Gaussians, is used to compute the response on both training and testing images at pixel level. The textons are then obtained from the centroids by K-means clustering on the responses of the filter-bank. Since the nearby images in the testing sequence are very similar, to save computation time and memory space, we do not run the texton clustering over the whole sequence. We currently pick up only one out of six images for obtaining the clustered textons. After the textons are identified, the texture-layout descriptor [Shotton et al. 2009] is adopted to extract the features for classifier training, because it has been proved to be successful in recognizing and segmenting images of general classes. The each dimension of the descriptor corresponds a pair $[r, t]$ of an image region r and a texton t. The region r relative to a given pixel location is a rectangle chosen at random within a rectangular window of ± 100 pixels. The response $v_{[r,t]}(i)$ at the pixel location i is the proportion of pixels under regions $r + i$ that have the texton t, i.e. $v_{[r,t]}(i) = \sum_{j \in (r+i)} [T_j = t] / \text{size}(r)$.

Classifier We employ the Joint Boost algorithm [Torralba et al. 2007; Shotton et al. 2009], which iteratively selects discriminative texture-layout filters as weak learners, and combines them into a strong classifier of the form $H(l, i) = \sum_m h_i^m(l)$. Each weak learner $h_i(l)$ is a decision stump based on the response $v_{[r,t]}(i)$ of the form

$$h_i(l) = \begin{cases} a\left[v_{[r,t]}(i) > \theta\right] + b & l \in \mathcal{C} \\ k^l & l \notin \mathcal{C} \end{cases}.$$

For those classes that share the feature $l \in \mathcal{C}$, the weak learner gives $h_i(l) \in \{a + b, b\}$ depending on the comparison of feature response to a threshold θ. For classes not sharing the feature $l \notin \mathcal{C}$, the constant k^l makes sure that unequal numbers of training examples of each class do not adversely affect the learning procedure. We use sub-sampling and random feature selection techniques for the iterative boosting [Shotton et al. 2009]. The estimated confidence value can be reinterpreted as a probability distribution using softmax transformation:

$$P_g(l, i) = \frac{\exp(H(l, i))}{\sum_k \exp(H(l, k))}.$$

For performance and speed, the classifier will not be trained from the full labeled data that might be huge. We choose a subset of labeled images that are closest to the given testing sequence to train the classifier, in order to guarantee the learning of reliable and transferable knowledge. We use the gist descriptor [Oliva and Torralba 2006] to characterize the distance between an input image and a labeled image, because the descriptor has been shown to work well for retrieving images of similar scenes in semantics, structure, and geo-locations. We create a gist descriptor for each image with 4 by 4 spatial resolution where each bin contains the average response to steerable filters in 4 scales with 8,8,4 and 4 orientations respectively in that image region. After the distances between the labeled

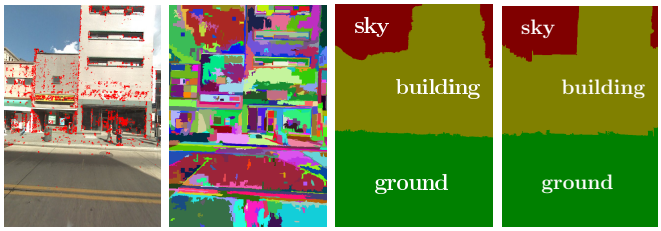


Figure 4: *Recognition and segmentation. (a) One input image. (b) The over-segmented patches. (c) The recognition per pixel. (d) The segmentation.*

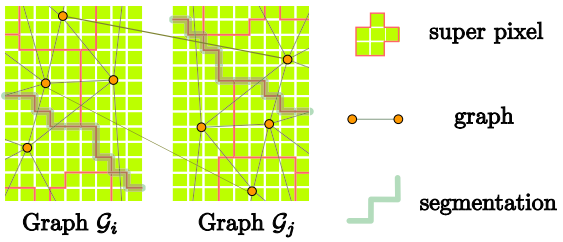


Figure 5: *Graph topology for multi-view semantic segmentation.*

images and input images of the testing sequence are computed, we then choose the 20 closest labeled images from the database as the training data for the sequence by nearest neighbor classification.

Location prior A camera is usually kept approximately straight in capturing. It is therefore possible to learn the approximate location priors of each object class. In a street-side image, for example, the sky always appears in the upper part of the image, the ground in the lower part, and the buildings in-between. Thus, we can use the labeled data to compute the accumulated frequencies of different object classes $P_l(l,i)$. Moreover, the camera moves laterally along the street in the capturing of street-side images. A pixel at the same height in the image space should have the same chance of belonging to the same class. With this observation, we only need to accumulate the frequencies in the vertical direction of the image space.

3.2 Multi-view semantic segmentation

The per-pixel recognition produces a semantic segmentation of each input image. But the segmentation is noisy and needs to be optimized in a coherent manner for the entire reconstructed sequence. Since the testing sequence has been reconstructed by SFM, we utilize the point matching information among multiple views to impose segmentation consistency.

Graph topology Each image I_i is first over-segmented using the method by [Felzenszwalb and Huttenlocher 2004]. Then we build a graph $\mathcal{G}_i = \langle \mathcal{V}_i, \mathcal{E}_i \rangle$ on the over-segmentation patches for each image. Each vertex $v \in \mathcal{V}_i$ in the graph is an image patch or a super-pixel in the over-segmentation, while the edges $\mathcal{E}_i$ denote the neighboring relationships between super-pixels. Then, the graphs $\{\mathcal{G}_i\}$ from multiple images in the same sequence are merged into a large graph $\mathcal{G}$ by adding edges between two super-pixels in correspondence but from different images. The super-pixels p_i and p_j in images I_i and I_j are in correspondence, if and only if there is at least one feature track $\mathbf{t} = \langle (x_u, y_u, i), (x_v, y_v, j), \dots \rangle$ with projection (x_u, y_u) lying inside the super-pixel p_i in image I_i, and (x_v, y_v) inside p_j in I_j. To limit the graph size, there is at most only one edge e_{ij} between any super-pixel p_i and p_j in the final graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, which is shown in Figure 5.

Adaptive features For object segmentation with fine boundaries, we prefer to use color cues to characterize the local appearance. In our model, the color distribution of all pixels in the image is approximated by a mixture model of m Gaussians in the color space with mean $\mathbf{u}_k$ and covariance $\mathbf{\Sigma}_k$. At the beginning, all pixel colors in all images of the same sequence are taken as input data points, and K-means is used to initialize a mixture of 512 Gaussians in RGB space. Let γ_{kl} denote the probability that the k-th Gaussian belongs to class l. The probability of vertex p_i having label l is

$$P_a(l,i) = \sum_{k=1}^{m} \gamma_{kl} \mathcal{N}(\mathbf{c}_i | \mathbf{u}_k, \mathbf{\Sigma}_k).$$

To compute γ, the probability $P_g(l,i)$ is used solely in a greedy way to obtain an initial segmentation $\{l_i\}$ as shown in Figure 4(c). This initial segmentation $\{l_i\}$ is then used to train a Maximal Likelihood estimate for γ from

$$\gamma_{kl} \propto \sum_{p_i \in \mathcal{V}} [l_i = k]\, p(\mathbf{c}_i | \mathbf{u}_k, \mathbf{\Sigma}_k)$$

under the constraint $\sum_k \gamma_{kl} = 1$. Now, combining the costs from both the local adaptive feature and the global feature, we define the data cost as

$$\psi_i(l_i) = -\log P_a(l,i) - \lambda_l \log P_l(l,i) - \lambda_g \log P_g(l,i).$$

Smoothing terms For an edge $e_{ij} \in \mathcal{E}_k$ in the same image I_k, the smoothness cost is

$$\psi_{ij}(l_i, l_j) = [l_i \neq l_j] \cdot g(i,j)$$

with $g(i,j) = 1/(\zeta \|\mathbf{c}_i - \mathbf{c}_j\|^2 + 1)$, where $\|\mathbf{c}_i - \mathbf{c}_j\|^2$ is the L_2-Norm of the RGB color difference of two super-pixels p_i and p_j. Note that $[l_i \neq l_j]$ allows capturing the gradient information only along the segmentation boundary. In other words, ψ_{ij} is penalizing the assignment to the different labels of the adjacent nodes.

For an edge $e_{ij} \in \mathcal{E}$ across two images, the smoothness cost is

$$\psi_{ij}(l_i, l_j) = [l_i \neq l_j] \cdot \lambda |\mathbf{T}|\, g(i,j),$$

where $\mathbf{T} = \{\mathbf{t} = \langle (x_u, y_u, i), (x_v, y_v, j), \dots \rangle\}$ is the set of all feature tracks with projection (x_u, y_u) inside the super-pixel p_i in image I_i, and (x_v, y_v) inside p_j in I_j. This definition favors two super-pixels having more matching tracks to have the same label, as the cost of having different labels is higher when $|\mathbf{T}|$ is larger.

Optimization With the constructed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, the labeling problem is to assign a unique label l_i to each node $p_i \in \mathcal{V}$. The solution $L = \{l_i\}$ can be obtained by minimizing the Gibbs energy [Geman and Geman 1984]:

$$E(L) = \sum_{p_i \in \mathcal{V}} \psi_i(l_i) + \rho \sum_{e_{ij} \in \mathcal{E}} \psi_{ij}(l_i, l_j).$$

Since the cost terms we defined satisfy the metric requirement, Graph Cut alpha expansion [Boykov et al. 2001] is used to obtain a local optimized label configuration L within a constant factor of the global minimum.

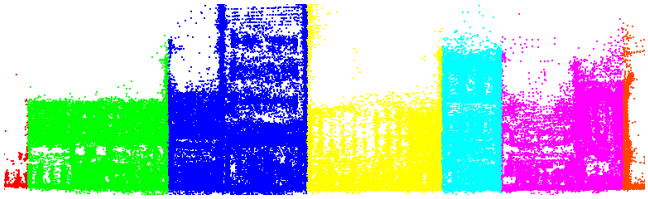


Figure 6: *Building block partition. Different blocks are shown by different colors.*

Algorithm 1 Inverse Orthographic Patching

```
for each image I_k visible to the façade do
    for each super pixel p_i ∈ I_k do
        if normal direction of p_i parallel with z-axis then
            for each pixel (x, y) in the bounding box do
                X ← (x, y, z_i)^T                ▷ z_i is the depth of p_i
                compute projection (u, v) of X to Camera i
                if super pixel index of (u, v) in I_k = k then
                    accumulate depth z_i, color, segmentation
```

4 Block partition

The reconstructed sequence needs to be partitioned into independent building blocks for each block to be individually modeled. However, the definition of building block is not unique, in that a block may contain a fraction or any number of physical buildings as long as they share a common dominant plane. As an urban scene is characterized by plenty of man-made structures of vertical and horizontal lines, we use the vertical lines to partition the sequence into blocks because they are stable and distinct separators for our purpose. Moreover, a local alignment process is used to place building blocks regularly. Such local alignment makes the analysis and implementation of the model reconstruction algorithm more straightforward.

4.1 Global vertical alignment

We first remove the line segments that are projected out of the segmented building regions from the previous section. From all the remaining vertical line segments, we compute the global vertical direction of gravity by taking the median direction of all reconstructed 3D vertical lines, found during the preprocessing stage in Section 2. Then, we align the y-axis of coordinate system for the reconstructed sequence with the estimated vertical direction.

4.2 Block separator

To separate the entire scene into natural building blocks, we find that the vertical lines are an important cue as a block separator, but there are too many vertical lines that tend to yield an over-partition. Therefore, we need to choose a subset of vertical lines as the block separators using the following heuristics.

Intersection A vertical line segment is a block separator if its extended line does not meet any horizontal line segments within the same façade. This heuristic is only true if the façade is flat. We compute a score for each vertical line segment L by accumulating the number of intersections with all horizontal line segments in each image $N(L) = \frac{1}{m}\sum_{k=1}^{m}\Gamma_k$ where Γ_k is the number of intersections in the k-th image and m is the number of correspondences of the line L in the sequence.

Height A vertical line is a potential block separator if its left block and right block have different heights. We calculate the height for its left and right blocks as follows: (1) In every image where the vertical line is visible, we fit two line segments to the upper and lower boundary of the corresponding building region respectively. (2) A height is estimated as the Euclidean distance of the mid-points of these two best-fit line segments. (3) The height of a block is taken as the median of all its estimated heights in corresponding images.

Texture Two buildings often have different textures, so a good block separator is the one that gives very different texture distributions for its left and right blocks. For each block, we build an average color histogram h_0 from multiple views. To make use of the spatial information, each block is further downsampled $t-1$ times to compute several normalized color histograms $h_1, ..., h_{t-1}$. Thus, each block corresponds to a vector of multi-resolution histograms. The dissimilarity of two histogram vector h^{left} and h^{right} of neighboring blocks is defined as $D_t\left(h^{left}, h^{right}\right) = \frac{1}{t}\sum_{k=0}^{t-1} d(h_k^{left}, h_k^{right})$ where $d(\cdot,\cdot)$ is the Kullback-Leibler divergence.

With these heuristics, we first sort all vertical lines by increasing number of intersections N, and retain the first half of the vertical lines. Then, we select all vertical lines that result in more than 35% in height difference for its left and right blocks. After that, we sort again all remaining vertical lines, now by decreasing texture difference D_t, and we select only the first half of the vertical lines. This selection procedure is repeated until each block is in a predefined range (from 6 to 30 meters in the current implementation).

4.3 Local horizontal alignment

After the global vertical alignment in the y-axis, the desired façade plane of the block is vertical, but it may not be parallel to the xy-plane of the coordinate frame. We automatically compute the vanishing point of the horizontal lines in the most fronto-parallel image of the block sequence to obtain a rotation around the y-axis for alignment of the x-axis with the horizontal direction. Note that this is done locally for each block if there are sufficient horizontal lines in the chosen image. After these operations, each independent façade is facing the negative z axis with x axis as horizontal direction from left to right, and y axis as vertical direction from top to down in their local coordinate system respectively.

5 Façade modeling

Since the semantic segmentation has identified the region of interest, and the block partition has separated the data into façade level, the remaining task is to model each façade. The reconstructed 3D points are often noisy or missing due to varying textureness as well as matching and reconstruction errors. Therefore, we introduce a building regularization method in the orthographic view of the façade for structure analysis and modeling. We first filter out the irrelevant 3D points by semantic segmentation and block separator. Orthographic depth map and texture image are composed from multiple views in Section 5.1, and provide the working image space for later stages. In Section 5.2, the structure elements on each façade are identified and modeled. When the identification of structure elements is not perfect, a backup solution is introduced in Section 5.3 to rediscover these elements, if they repetitively appear. Now, after the details inside each façade have been modeled, the boundaries of the façade are regularized in Section 5.4 to produce the final model.

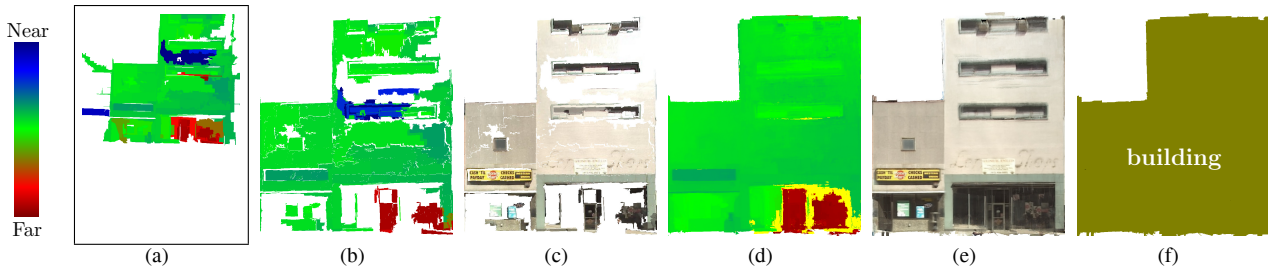


Figure 7: *Inverse orthographic composition. (a) Depth map in input image space. (b) Partial orthographic depth map from one view. (c) Partial orthographic texture from one view. (d) Composed orthographic depth map (unreliably estimated pixels are in yellow). (e) Composed orthographic texture. (f) Composed orthographic building region.*

5.1 Inverse orthographic composition

Each input image of the building block is over-segmented into patches using [Felzenszwalb and Huttenlocher 2004]. The patch size is a trade-off between accuracy and robustness. We choose 700 pixels as the minimum patch size for our images at a resolution of 640×905 pixels to favor relatively large patches since the reconstructed 3D points from images are noise.

Patch reconstruction The normal vector and center position of each p_i are estimated from the set of 3D points $\mathcal{P}_i = \{(x_k, y_k, z_k)\}$, which have projections inside p_i. As the local coordinate frame of the block is aligned with the three major orthogonal directions of the building, the computation is straightforward. Let σ_x^i, σ_y^i and σ_z^i be the standard deviations of all 3D points in $\mathcal{P}_i$ in three directions. We first compute the normalized standard deviations $\hat{\sigma}_x^i = \frac{\bar{s}_x}{s_x^i}\sigma_x^i$, $\hat{\sigma}_y^i = \frac{\bar{s}_y}{s_y^i}\sigma_y^i$, where s_x^i and s_y^i are the horizontal and vertical sizes of the bounding box of the patch in the input images, and their median respectively across all patches $\bar{s}_x = \text{median}_i s_x^i$, $\bar{s}_y = \text{median}_i s_y^i$. The normalization avoids bias to a small patch. The patch p_i is regarded as parallel to the façade base plane if σ_z is smaller than σ_x^i and σ_y^i. And, all these parallel patches with small σ_z contribute to the composition of an orthographic view of the façade. The orientation of such a patch p_i is aligned with the z-axis, and its position set at the depth $z_i = \text{median}_{(x_j, y_j, z_j) \in p_i} z_j$. One example is shown in Figure 7(a).

Orthographic composition To simplify the representation for irregular shapes of the patches, we deploy a discrete 2D orthographic space on the xy-plane to create an orthographic view O of the façade. The size and position of O on the xy-plane are determined by the bounding box of the 3D points of the block, and the resolution of O is a parameter that is actually set not to exceed 1024×1024. Each patch is mapped from its original image space onto this orthographic space as illustrated in Figure 7 from (a) to (b). We use an inverse orthographic mapping algorithm shown in Algorithm 1 to avoid gaps. Theoretically, the warped textures of all patches create a true orthoimage O as each used patch has a known depth and is parallel with the base plane.

For each pixel v_i of the orthoimage O, we accumulate a set of depth values $\{z_j\}$, a set corresponding of color values $\{\mathbf{c}_j\}$ and a set of segmentation labels $\{l_j\}$. The depth of this pixel is set to the median of $\{z_j\}$ whose index is $\kappa = \arg\text{median}_j z_j$. Since the depth determines the texture color and segmentation label, we take $\mathbf{c}_\kappa$ and l_κ as the estimated color and label for the pixel. In practice, we accept a small set of estimated points around z_κ and take their mean as the color value in the texture composition. As the content of images are highly overlapped, if a pixel is observed only once from one image, it is very likely that it comes from an incorrect reconstruction. It will thus be rejected in the depth fusion process. Moreover, all pixels $\{v_i\}$ with multiple observations $\{\{z_j\}_i\}$ are sorted in non-decreasing order according to their standard deviation $\varsigma_i = sd(\{z_j\})$ of depth sets. After that, we define $\varsigma(\eta)$ to be the $\eta|\{v_i\}|$-th element in the sorted $\{\varsigma_i\}$. We declare the pixel v_i to be unreliable if $\varsigma_i > \varsigma(\eta)$. The value of η comes from the estimated confidence of the depth measurements. We currently scale the value by the ratio of the number of 3D points and the total pixel number of O.

Note that when we reconstruct the patches, we do not use the semantic segmentation results in the input image space for two reasons. The first is that the patches used in reconstruction are much larger in size than those used for semantic segmentation, this may lead to an inconsistent labeling. Though it is possible to estimate a unique label for a patch, it may downgrade the semantic segmentation accuracy. The second is that the possible errors in the semantic segmentation may over-reject patches, which compromises the quality of the depth estimation. Therefore, we reconstruct the depth first and transfer the segmentation results from the input image space to the orthographic view with pixel-level accuracy, shown in Figure 7(f). After that, we remove the non-building pixels in the orthoimage according to the segmentation label. Our composition algorithm for the orthographic depth map is functionally close to the depth map fusion techniques such as [Curless and Levoy 1996]. But our technique is robust as we use the architectural prior of orthogonality that preserves structural discontinuity without over-smoothing.

5.2 Structure analysis and regularization

From the composed orthographic depth map and texture image for each façade, we want to identify the structural elements at different depths of the façade to enrich the façade geometry. To cope with the irregular, noisy and missing depth estimations on the façade, a strong regularization from the architecture priors is therefore required. Most of buildings are governed by vertical and horizontal lines and form naturally rectangular shapes. We restrict the prior shape of each distinct structure element to be a rectangle, such as the typical extruding signboard in Figure 7.

Joint segmentation We use a bottom-up, graph-based segmentation framework [Felzenszwalb and Huttenlocher 2004] to jointly segment the orthographic texture and depth maps into regions, where each region is considered as a distinct element within the façade. The proposed shape-based segmentation method jointly utilizes texture and depth information, and enables the fully automatic façade structure analysis. Xiao et al. [2008] also proposed a functional equivalent top-down recursive sub-division method. However, it has been shown in [Xiao et al. 2008] to be inefficient to produce satisfactory result without any user interaction.

(a) (b) (c) (d) (e) (f)

Figure 8: *Structure analysis and regularization for modeling. (a) The façade segmentation. (b) The data cost of boundary regularization. The cost is color-coded from high at red to low at blue via green as the middle. (c) The regularized depth map. (d) The texture-mapped façade. (e) The texture-mapped block. (f) The block geometry.*

A graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined on the orthoimage image O with all pixels as vertices $\mathcal{V}$ and edges $\mathcal{E}$ connecting neighboring pixels. To encourage horizontal and vertical cut, we use 4-neighborhood system to construct $\mathcal{E}$. The weight function for an edge connecting two pixels with reliable depth estimations is based both on the color distance and normalized depth difference

$$w\left((v_i, v_j)\right) = \|\mathbf{c}_i - \mathbf{c}_j\|^2 \cdot \left(\frac{z_i - z_j}{\varsigma(\eta)}\right)^2,$$

where $\|\mathbf{c}_i - \mathbf{c}_j\|^2$ is the L_2-Norm of the RGB color difference of two pixels v_i and v_j. We slightly pre-filter the texture image using a Gaussian of small variance before computing the edge weights. The weight for an edge connecting two pixels without reliable depth estimations is set to 0 to force them to have the same label. We do not construct an edge between a pixel with a reliable depth and a pixel without a reliable depth, as the weight cannot be defined.

We first sort $\mathcal{E}$ by non-decreasing edge weight w. Starting with an initial segmentation in which each vertex v_i is in its own component, the algorithm repeats for each edge $e_q = (v_i, v_j)$ in order for the following process: If v_i and v_j are in disjoint components $C_i \neq C_j$, and $w(e_q)$ is small compared with the internal difference of both those components, $w(e_q) \leq MInt(C_i, C_j)$, then the two components are merged. The minimum internal difference is defined as

$$MInt(C_1, C_2) = \min\left(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)\right),$$

where the internal difference of a component C is the largest weight in the minimum spanning tree of the component

$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

The non-negative threshold function $\tau(C)$ is defined on each component C. The difference in this threshold function between two components must be greater than their internal difference for an evidence of a boundary between them. Since we favor a rectangular shape for each region, the threshold function $\tau(C)$ is defined by the divergence $\vartheta(C)$ between the component C and a rectangle, which is the portion of the bounding box B_C with respect to the component C, $\vartheta(C) = |B_C| / |C|$. For small components, $Int(C)$ is not a good estimate of the local characteristics of the data. Therefore, we let the threshold function be adaptive based on the component size,

$$\tau(C) = \left(\frac{\varrho}{|C|}\right)^{\vartheta(C)},$$

where ϱ is a constant and is set to 3.2 in our prototype. τ is large for components that do not fit a rectangle, and two components with large τ are more likely to be merged. A larger ϱ favors larger components, as we require stronger evidence of a boundary for smaller components.

Once the segmentation is accomplished, the depth values for all pixels in C_i of each reliable component C_i are set to the median. The depth of the largest region is regarded as the depth of the base plane for the façade. Moreover, an unreliable component C_i smaller than a particular size, i.e. smaller than 4% of the current façade area, is merged to its only reliable neighboring component if such a neighboring component exists.

Shape regularization Except for the base plane of the façade, we fit a rectangle to each element on the façade. For an element $C = \{v_i = (x_i, y_i)\}$, we first obtain the median position $(x_{\text{med}}, y_{\text{med}})$ by $x_{\text{med}} = \text{median}_i x_i$ and $y_{\text{med}} = \text{median}_i y_i$. We then remove outlier points that are $|x_i - x_{\text{med}}| > 2.8\sigma_x$ or $|y_i - y_{\text{med}}| > 2.8\sigma_y$, where $\sigma_x = \sum_i |x_i - x_{\text{med}}| / |C|$ and $\sigma_y = \sum_i |y_i - y_{\text{med}}| / |C|$. Furthermore, we reject the points that are in the 1% region of the left, right, top and bottom according to their ranking of x and y coordinates in the remaining point set. In this way, we obtain a reliable subset C_{sub} of C. We define the bounding box $B_{C_{\text{sub}}}$ of C_{sub} as the fitting rectangle of C. The fitting confidence is then defined as

$$f_C = \frac{B_{C_{\text{sub}}} \cap C}{B_{C_{\text{sub}}} \cup C}.$$

In the end, we only retain the rectangle as distinct façade element if its confidence $f_C > 0.72$ and the rectangle size is not too small.

The rectangular elements are automatically snapped into the nearest vertical and horizontal mode positions of the accumulated Sobel responses on the composed texture image, if their distances are less than 2% of the width and height of the current façade. The detected rectangles can be nested within each other. When producing the final 3D model, we first pop up the larger element from the base plane and then the smaller element within the larger element. If two rectangles overlap but do not contain each other, we first pop up the one that is closest to the base plane.

5.3 Repetitive pattern rediscovery

Structure elements are automatically reconstructed in the previous section. However, when the depth composition quality is not good enough due to poor image matching, reflective materials or low image quality, only a few of them could be successfully recovered. For repetitive elements of the façade, we can now systematically launch a re-discovery process using the discovered elements as templates in the orthographic texture image domain. The idea of taking advantage of repetitive nature of the elements has been explored in [Müller et al. 2007; Xiao et al. 2008].

We use the Sum of Squared Differences (SSD) on RGB channels for template matching. Unlike [Xiao et al. 2008] operating in 2D search space, we use a two-step method to search twice in 1D, shown in Figure 9. We first search in horizontal direction for a template B_i

and obtain a set of matches $\mathbf{B}_i$ by extracting the local minima under a threshold. Then, we use both B_i and $\mathbf{B}_i$ together as the template to search for the local minima along the vertical direction. This leads to more efficient and robust matching, and automatic alignment of the elements. A re-discovered element by template matching inherits the depth of the template.

When there are more than one structure elements discovered previously by joint segmentation representing the same kind of structure elements, we also need to cluster the re-discovered elements using a bottom-up hierarchical merging mechanism. Two templates B_i and B_j obtained by joint segmentation with sets of matching candidates $\mathcal{M}_i$ and $\mathcal{M}_j$ are merged into the same class, if one template is sufficiently similar to any element of the candidates of the other template. Here, the similarity between two elements is defined as the ratio of the intersection area by the union area of the two elements. The merging process consists of averaging element sizes between $\mathcal{M}_i \cup \{B_i\}$ and $\mathcal{M}_j \cup \{B_j\}$, as well as computing the average positions for overlapped elements in $\mathcal{M}_i \cup \{B_i\}$ and $\mathcal{M}_j \cup \{B_j\}$.

5.4 Boundary regularization

The boundaries of the façade of a block are further regularized to favor sharp change and penalize serration. We use the same method as for shape regularization of structure elements to compute the bounding box $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ of the façade. Finally, we further optimize the upper boundary of the façade, as we cannot guarantee that a building block is indeed a single building with the same height during block partition.

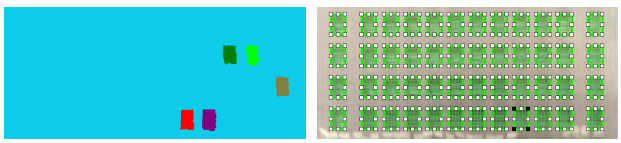
Illustrated in Figure 10, we lay out a 1D Markov random field on the horizontal direction of the orthoimage. Each $x_i \in [x_{\min}, x_{\max}]$ defines a vertex, and an edge is added for two neighboring vertices. The label l_i of x_i corresponds to the position of the boundary, and $l_i \in [y_{\min}, y_{\max}]$ for all x_i. Therefore, one label configuration of the MRF corresponds to one façade boundary. Now, we utilize all texture, depth and segmentation information to define the cost.

The data cost is defined according to the horizontal Sobel responses

$$\phi_i(l_j) = 1 - \frac{\text{HorizontalSobel}(i,j)}{2\max_{xy}\text{HorizontalSobel}(x,y)}.$$

Furthermore, if l_j is close to the top boundary r_i of reliable depth map, $|l_j - r_i| < \beta$, where β is empirically set to $0.05(y_{\max} - y_{\min} + 1)$, we update the cost by multiplying it with $(|l_j - r_i| + \epsilon)/(\beta + \epsilon)$. Similarly, if l_j is close to the top boundary s_i of segmentation $|l_j - s_i| < \beta$, we update the cost by multiplying it with $(|l_j - s_i| + \epsilon)/(\beta + \epsilon)$. For the façades whose boundaries are not in the viewing field of any input image, we snap the façade boundary to the top boundary of the bounding box, and empirically update $\phi_i(y_{\min})$ by multiplying it with 0.8. Figure 8(b) shows one example of defined data cost.

The height of the façade upper boundary usually changes in the regions with strong vertical edge responses. We thus



(a) The façade segmentation (b) Matching results using the violet template in (a)

Figure 9: *Repetitive pattern rediscovery.*

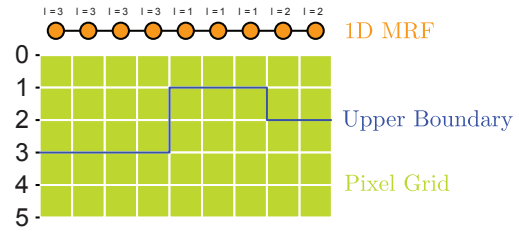
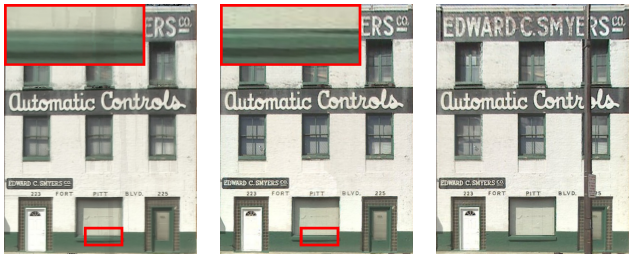


Figure 10: *An example of MRF to optimize façade upper boundary.*



(a) (b) (c)

Figure 11: *Texture optimization. (a) The original orthographic texture image. (b) The optimized texture image. (c) A direct texture composition. The optimized texture image in (b) is more clear than the original orthographic texture image in (a), and has no texture from occluding objects, such as the one contained in (c).*

accumulate vertical Sobel responses at each x_i into $V_i = \sum_{y\in[y_{\min},y_{\max}]} \text{VerSobel}(i,y)$, and define the smoothness term to be

$$\phi_{i,i+1}(l_i, l_{i+1}) = \mu\,|l_i - l_{i+1}|\left(1 - \frac{V_i + V_{i+1}}{2\max_j V_j}\right),$$

where μ is a controllable parameter.

The boundary is optimized by minimizing a Gibbs energy [Geman and Geman 1984]

$$E(L) = \sum_{x_i\in[x_{\min},x_{\max}]} \phi_i(l_i) + \sum_{x_i\in[x_{\min},x_{\max}-1]} \phi_{i,i+1}(l_i, l_{i+1}),$$

where ϕ_i is the data cost and $\phi_{i,i+1}$ is the smoothing cost. The exact inference can be obtained with a global optimum by methods such as belief propagation [Pearl 1982].

6 Post-processing

After the model for each façade is computed, the mesh is produced, and the texture is optimized.

6.1 Model production

Each façade is the front side of the building block. We can extend a façade in the z-direction into a box with a constant depth (the default constant is set to 18 meters in the current implementation) to represent the geometry of the building block, as illustrated in Figure 12(f).

All the blocks of a sequence are then assembled into the street side model. The texture mapping is done by visibility checking using z-buffer ordering. The side face of each block can be automatically textured as illustrated in Figure 12 if it is not blocked by the neighboring buildings.

6.2 Texture optimization

The orthographic texture for each front façade by Algorithm 1 is a true orthographic texture map. But as a texture image, it suffers from the artifacts of color discontinuities, blur and gaps, as each pixel has been independently computed as the color of the median depth from all visible views. However, it does provide very robust and reliable information for the true texture, and contain almost no outlier from occluding objects. Therefore, we re-compute an optimized texture image for each front façade, regarding the original orthographic texture image as a good reference.

Suppose that each façade has N visible views. Each visible view is used to compute a partial texture image for all visible points of the façade. Then we obtained N partial texture images for the façade. Next, we define a difference measurement as the squared sum of differences between each pixel of the partial texture images and the original orthographic texture image at the same coordinate. This is the data term for a Markov Random Field on the orthographic texture image grid. The smoothing term is defined to be the reciprocal of the color difference between each neighboring pair of pixels on the original orthographic texture image. The desired orthographic texture image is computed using Graph-Cut alpha-expansion [Boykov et al. 2001]. If seam artifacts are serious, Poisson blending [Pérez et al. 2003] can be used as post-process. Figure 11 shows the comparative results of this process. Figure 11(c) also shows a direct texture warping from most fronto-parallel image as in [Xiao et al. 2008], which fails to remove the occluding objects, i.e. the telegraph pole in this case.

7 Experiment and Discussion

We have implemented our system and tested on the street-side images of downtown Pittsburgh from Google. These images have been used in Google Street View to create seamless panoramic views. Therefore, the same kind of images is currently available for a huge number of cities around the whole world, which have been captured without online human control and with noises and glares. The image resolution is 640×905. The entire sequence of 10,498 images in Pittsburgh is broken down into a few shorter sequences roughly every 100 consecutive images. Then, each sequence is reconstructed using the structure from motion algorithm to produce a set of semi-dense points and camera poses. The cameras are then geo-registered back to the GPS coordinate frame. The result corresponding to this sequence is geo-registered back to the global earth coordinate using available GPS data. Since the sequence is not very long, there is no obvious drift effect, and we don't explicitly handle loop closing.

7.1 Implementation details

The implementation is in unoptimized C++ code, and the parameters are manually tuned on a set of 5 façades. The whole system consists of three major components: SFM, segmentation, and modeling, and it is completely modular. We use the code from [Oliva and Torralba 2006] for gist feature extraction, the code from [Shotton et al. 2009] for Joint Boost classification, the code from [Boykov et al. 2001] for Graph Cut alpha expansion in MRF optimization, the code from [Felzenszwalb and Huttenlocher 2004] for joint graph-based segmentation in structure analysis and over-segmentation. Note that after block partition, the façade analysis and modeling component works in a rectified orthographic space, which involves only simple 2D array operations in implementation.

The resulted model is represented by pushed-and-popped rectangles. As shown in Figure 8(f) and Figure 12(e), the parallepipeds-like division is the quadrilateral tessellation of the base plane mesh by "rectangle map", which is inspired by the trapezoid map algorithm [de Berg et al. 2008]. A rectangle is then properly extruded according to the depth map. The results may seem to have some "parallepipeds", while some of them may have the same depth as neighboring, depending precisely upon their reconstructed depth values.

For a portion of Pittsburgh, we reconstructed 202 building blocks from 10,498 images. On a small cluster composed by 15 normal desktop PCs, the results are produced automatically in 23 hours, including approximately 2 hours for SFM, 19 hours for segmentation, and 2 hours for partition and modeling. Figure 12 shows different examples of blocks and the intermediate results. Figure 15 shows a few close-up views of the final model. All presented results in the paper and in the accompanying video are "**as is**" without any manual touch-up. For rendering, each building block is represented in two levels of detail. The first level has only the façade base plane. The second level contains the augmented elements of the façade.

In the semantic segmentation, we hand-labeled 173 images by uniformly sampling images from our data set to create the initial database of labeled street-side images. Some example labeled data is shown in the accompanying video. Each sequence is recognized and segmented independently. For testing, we do not use any labeled images if they come from the same sequence in order to fairly demonstrate the real performance on unseen sequences.

Our method is remarkably robust for modeling as the minor errors or failure cases do not create visually disturbing artifacts. The distinct elements such as windows and doors within the façade may not always be reconstructed due to lack of reliable 3D points. They are often smoothed to the façade base plane with satisfactory textures as the depth variation is small. Most of the artifacts are from the texture. Many of the trees and people are not removed from the textures on the first floor of the buildings seen in Figure 15. These could be corrected if an interactive segmentation and inpainting is used. There are some artifacts on the façade boundaries if the background buildings are not separated from the foreground buildings, shown in the middle of Figure 15. Some other modeling examples are also shown in Figure 12. Note that there are places where the top of the buildings are chopped off, because they are clipped in the input images.

7.2 Comparative studies

Comparison with semi-automatic methods There are several semi-automatic image-based modeling methods [Debevec et al. 1996; Xiao et al. 2008]. The work in [Xiao et al. 2008] is the most recent representative approach targeting a single façade modeling with interactive initialization and editing. Although our method is fully automatic for the entire street-side city modeling pipeline, we could compare the functional equivalent component of the automatic structure analysis of a given façade in Section 5.2 of this paper and the automatic part of [Xiao et al. 2008] in Sections 6 and 7. The results are compared in Figure 14. Note that for comparison, we manually specify the initial planes with accurate boundaries, as shown in Figure 14(g). Since they made the very strong assumption that the depth variation is small for each façade, we have to specify two initial planes due to the large depth difference in this example. The input images are captured by hand-held DSLR camera at high resolution. For this data set, with simple normalization of image space to [-1,+1], we can use exactly the same set of parameters as for Pittsburgh data set. The results clearly indicate that our façade analysis is more robust and results in superior results before any manual retouching. For repetitive patterns such as windows, they relies on user specification or trained model to identify the template (the first region) for each façade and match around to find more occurrences of the template. In our approach, since our shape-based

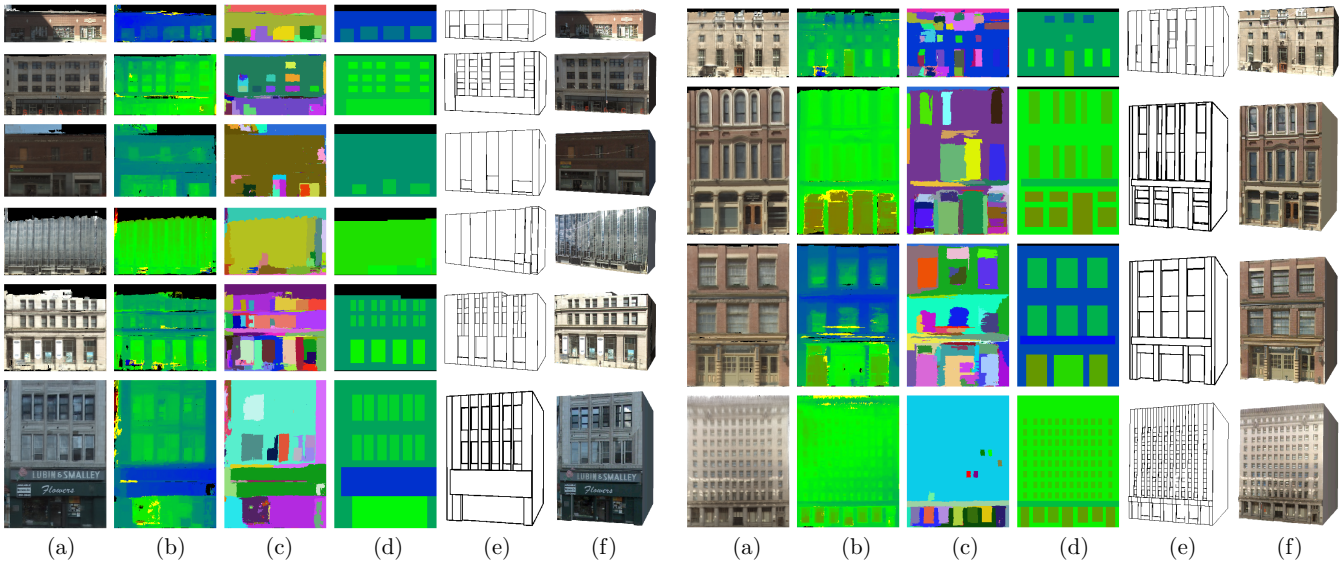


Figure 12: *Modeling examples of various blocks. (a) The orthographic texture. (b) The orthographic color-coded depth map (yellow pixel is unreliable). (c) The façade segmentation. (d) The regularized depth map. (e) The geometry. (f) The textured model.*

(a) Orthographic texture (b) 3D model

Figure 13: *A challenging non-rectangular case. The slope-shape roof is approximated as a step-shape structure.*

joint segmentation can automatically identify several templates, it is unnecessary to specify the template manually or to train a window recognition model. Therefore, it reduces the manual efforts or system complexity, and can perform more robustly since the correct recognition of windows is non-trivial.

Comparison with sensor-rich methods Using 3D scanners for city modeling is definitely an alternative for street-side city modeling [Stamos and Allen 2002; Frueh and Zakhor 2003]. The work in [Frueh and Zakhor 2003] is one of the most recent representative works that generate 3D street-side models from 3D scans captured along the streets at the ground level. Additionally, they also capture aerial views for the building tops that we do not model. The scanned data clearly has a higher density and accuracy in geometry than the reconstructed data from SFM. An approach integrating the images and scans could be envisaged to leverage the advantage of image analysis developed in our method and the high quality of 3D point clouds from the scanns. Our image analysis accepts any available 3D points if the 3D data is registered with the images. The key to the success of the integrated approach is therefore the registration of the scanned data and the images. The potential challenge is the inconsistency of the registered data.

7.3 Limitations

There are a few limitations reflecting on the current implementation of the approach.

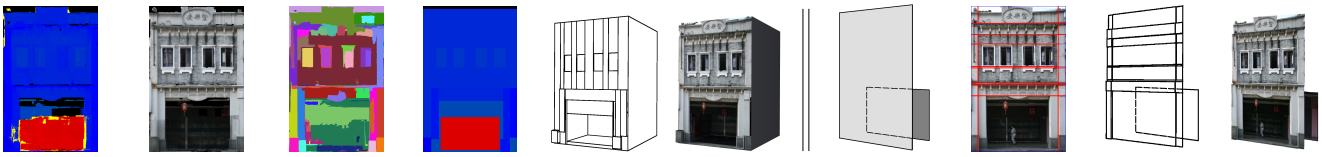
Rectilinear structure assumption As for any problem, a more flexible model with many degrees of freedom is difficult to be solved in practice. Therefore, imposing reasonable priors of building regularity is the trade-off that we have to make for robustness and automation. Rectilinear structure assumption, or equivalent Manhattan-world assumption [Coughlan and Yuille 1999], is universal for usual man-made buildings. For more complex buildings such as landmarks, the rectangular assumption of buildings can still be a first-level approximation of arbitrary surfaces. Moreover, at the given scale of street-side city reconstruction we are targeting in this paper, the assumption is sufficient, as what the final results demonstrated. For instances, some non-rectangular windows in the center of Figure 1 are well-approximated by rectangles without obvious artifact at the scale of street view. Figure 13 is another example that the roof shape directly conflicts with our assumption. With our 1D Markov random field regularization, the roof is approximated as a step-shape structure.

Camera viewing field The upper parts of large buildings are not modeled due to the limited viewing field of a ground-based camera. We could envisage to integrate aerial images as suggested in [Früh and Zakhor 2003], or deploy a multiple camera system with one of them pointing upward.

Potential interactive editing Our approach is fully automatic for all presented results. However, the method in [Xiao et al. 2008] does provide a very convenient user interface for manual operations. Nevertheless, since our rectangular representation is just a special case of DAG graph used in their method, our method can be seamlessly used together with the user interface provided by them for later manual process if necessary.

8 Conclusion

We have proposed a completely automatic image-based modeling approach that takes a sequence of overlapping images captured along the street and produces the complete photo-realistic 3D models. The main contributions are: a multiple view semantic segmentation method to identify the object classes of interest, a systematic partition of buildings into independent blocks using the man-made vertical and horizontal lines, and a robust façade modeling with



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 14: *Comparison with [Xiao et al. 2008] . Our results are presented from (a) to (f) with the same legend as in Figure 12. The automatic results provided by [Xiao et al. 2008] are presented from (g) to (j). (g) is the manually specified initial façades. (h) is the automatic subdivision result. (i) is the automatic geometry of the façades before interactive retouching. (j) is the textured façades.*

pushed and pulled rectangular shapes. More importantly, the components are assembled into a robust and fully automatic system. The approach has been successfully demonstrated on large amount of data.

There are a few limitations to the current implementation of the system, but they can be improved within the same framework. For example, we could incorporate the 3D information in the semantic segmentation. Furthermore, using the grammar rules extracted from the reconstructed models to synthesize missing parts procedurally is also an interesting further direction.

Acknowledgements

The work is supported by Hong Kong RGC Grants 618908 and 619107. We thank Google for the data and a Research Gift, and Honghui Zhang for helps in segmentation.

References

BARINOVA, O., KONUSHIN, V., YAKUBENKO, A., LIM, H., AND KONUSHIN, A. 2008. Fast automatic single-view 3-d reconstruction of urban scenes. In *Proceedings of the European Conference on Computer Vision*, 100–113.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11, 1222–1239.

CANNY, J. F. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 679–714.

CORNELIS, N., LEIBE, B., CORNELIS, K., AND GOOL, L. V. 2008. 3D urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision 78*, 2, 121–141.

COUGHLAN, J., AND YUILLE, A. 1999. Manhattan world: compass direction from a single image by bayesian inference. In *Proceeding of IEEE International Conference in Computer Vision*, vol. 2, 941–947.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH 96*, ACM Press / ACM SIGGRAPH, H. Rushmeier, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 303–312.

DE BERG, M., CHEONG, O., VAN KREVELD, M., AND OVERMARS, M. 2008. *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, Berlin.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of SIGGRAPH 96*, ACM Press / ACM SIGGRAPH, H. Rushmeier, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 11–20.

DICK, A., TORR, P., AND CIPOLLA, R. 2004. Modelling and interpretation of architecture from several images. *International Journal of Computer Vision 60*, 2, 111–134.

FELZENSZWALB, P., AND HUTTENLOCHER, D. 2004. Efficient graph-based image segmentation. *International Journal of Computer Vision 59*, 2, 167–181.

FRUEH, C., AND ZAKHOR, A. 2003. Automated reconstruction of building facades for virtual walk-thrus. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*, ACM, New York, NY, USA, 1–1.

FRÜH, C., AND ZAKHOR, A. 2003. Constructing 3d city models by merging ground-based and airborne views. In *Proceedings of IEEE Conference Computer Vision and Pattern Recognition*, 562–569.

GEMAN, S., AND GEMAN, D. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 6, 721–741.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Automatic photo pop-up. *ACM Transactions on Graphics 24*, 3 (Aug.), 577–584.

LHUILLIER, M., AND QUAN, L. 2005. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 418–433.

MÜLLER, P., ZENG, G., WONKA, P., AND GOOL, L. V. 2007. Image-based procedural modeling of façades. *ACM Transactions on Graphics* (Aug.), 85:1–85:10.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 433–442.

OLIVA, A., AND TORRALBA, A. 2006. Building the gist of a scene: the role of global image features in recognition. *Progress in Brain Research 155*, Part 2, 23–36.

PEARL, J. 1982. Reverend bayes on inference engines: a distributed hierarchical approach. In *Proceedings of AAAI National Conference on AI*, 133–136.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3 (Aug.), 313–318.

POLLEFEYS, M., NISTÉR, D., FRAHM, J., AKBARZADEH, A., MORDOHAI, P., CLIPP, B., ENGELS, C., GALLUP, D., KIM,

Figure 15: *Two close-up street-side views of the city models automatically generated from the images shown on the bottom.*

S., MERRELL, P., SALMI, C., SINHA, S., TALTON, B., WANG, L., YANG, Q., STEWNIUS, H., YANG, R., WELCH, G., AND TOWLES, H. 2008. Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision 78*, 2, 143–167.

SAXENA, A., SUN, M., AND NG, A. Y. 2009. Make3d: learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 5, 824–840.

SCHINDLER, G., KRISHNAMURTHY, P., AND DELLAERT, F. 2006. Line-based structure from motion for urban environments. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, 846–853.

SHOTTON, J., WINN, J., ROTHER, C., AND CRIMINISI, A. 2009. TextonBoost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision 81*, 1, 2–23.

SINHA, S. N., STEEDLY, D., SZELISKI, R., AGRAWALA, M., AND POLLEFEYS, M. 2008. Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics 27*, 5 (Dec.), 159:1–159:10.

STAMOS, I., AND ALLEN, P. K. 2002. Geometry and texture recovery of scenes of large scale. *Computer Vision and Image Understanding 88*, 2, 94–118.

TORRALBA, A., MURPHY, K., AND FREEMAN, W. 2007. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 5, 854–869.

VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. S. 2007. VideoTrace: rapid interactive scene modelling from video. *ACM Transactions on Graphics 26*, 3 (Aug.), 86:1–86:5.

WERNER, T., AND ZISSERMAN, A. 2002. Model selection for automated architectural reconstruction from multiple views. In *Proceedings of the British Machine Vision Conference*, 53–62.

WINN, J., CRIMINISI, A., AND MINKA, T. 2005. Object categorization by learned universal visual dictionary. In *Proceedings of IEEE International Conference in Computer Vision*, vol. 2, 1800–1807.

XIAO, J., FANG, T., TAN, P., ZHAO, P., OFEK, E., AND QUAN, L. 2008. Image-based façade modeling. *ACM Transactions on Graphics 27*, 5 (Dec.), 161:1–161:10.

ZEBEDIN, L., KLAUS, A., GRUBER-GEYMAYER, B., AND KARNER, K. 2006. Towards 3D map generation from digital aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing 60*, 6 (Sep.), 413–427.