Automatic Urban Modeling using Volumetric Reconstruction with Surface Graph Cuts

Ignacio Garcia-Dorado Ilke Demir Daniel G. Aliaga
Purdue University, USA

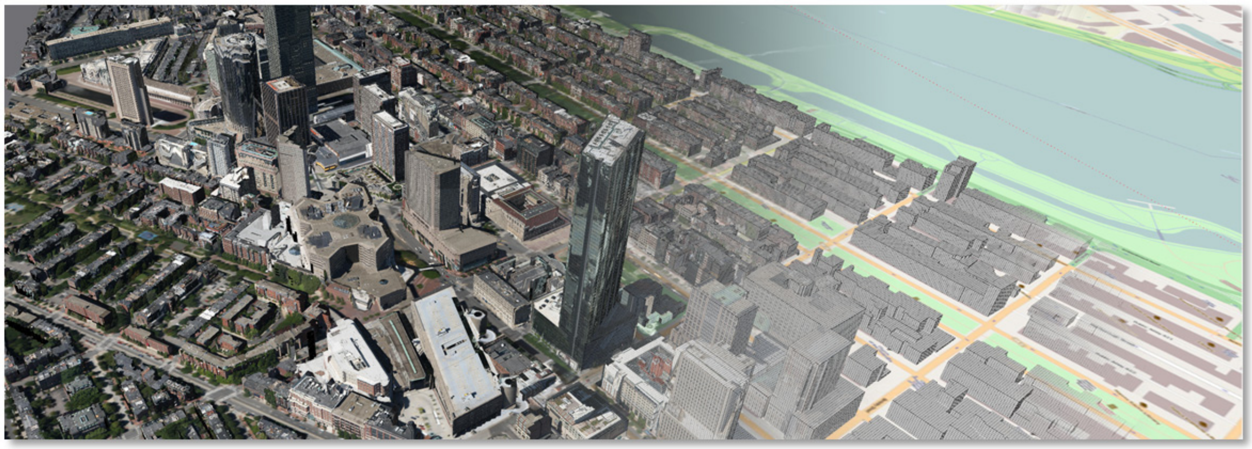


Figure 1. Urban Modeling. A complex urban area (left) is automatically obtained using volumetric reconstruction with surface graph cuts (middle) computed from aerial imagery and GIS-style parcel/building data (right). Our methodology uses photo-consistency to robustly recreate 2.5D building structures and surface graph cuts to assemble seamless and coherent textures despite occlusion, geometry, and calibration errors.

Abstract— The demand for 3D city-scale models has been significantly increased due to the proliferation of urban planning, city navigation, and virtual reality applications. We present an approach to automatically reconstruct buildings densely spanning a large urban area. Our method takes as input calibrated aerial images and available GIS meta-data. Our computational pipeline computes a per-building 2.5D volumetric reconstruction by exploiting photo-consistency where it is highly sampled amongst the aerial images. Our building surface graph cut method overcomes errors of occlusion, geometry, and calibration in order to stitch together aerial images and yield a visually coherent texture-mapped result. Our comparisons show similar quality to the manually modeled buildings of Google Earth, and show improvements over naive texture mapping and over space-carving methods. We have tested our algorithms with a 12 square kilometer area of Boston, MA (USA), using 4667 images (i.e., 280GB of raw image data) and producing 1785 buildings.

Index Terms — automatic, urban, photo-consistency, graph cuts, volumetric reconstruction.

1 Introduction

We present a method for automatic reconstruction of buildings densely spanning a city or portion thereof. The demand for such 3D volumetric content has been significantly increased due to the proliferation of urban planning, city navigation, and virtual reality applications (Figure 1). Nevertheless, automatic widespread reconstruction of urban areas is still an elusive target. Services, such as Google Earth/Maps, Apple Maps, Bing Maps, and OpenStreetMap have fomented the capture and availability of ubiquitous urban imagery and geographic information system (GIS) style data. Using LIDAR data is one option for city modeling however it still has challenges and is not always available. Ground-level imagery provides high resolution but such images are usually scattered and incomplete. Aerial images provide extensive and uniform coverage of large areas, albeit at lower resolution, and are widely available for most cities. Hence, to reconstruct large urban areas we focus on aerial imagery.

There have been several fundamental approaches for producing urban volumetric reconstructions. In contrast to partial (or facade-level) reconstructions (e.g., Müller et al. [24], Xiao et al. [43]), we seek to automatically create texture-mapped building envelopes spanning a large-portion of a city (i.e., akin to the crowd-sourced created models visible in Google Earth) – such complete models are suitable 3D content for the aforementioned graphics and visualization applications. Inverse procedural modeling approaches pursue generating parameterized 2D and 3D models from observations (e.g., Stava et al. [35], Bokeloh et al. [3], Park et al. [27]), but have not been demonstrated for large-scale urban areas due to the inherent complexity and ambiguity in the inversion process. Relevant volumetric recon-

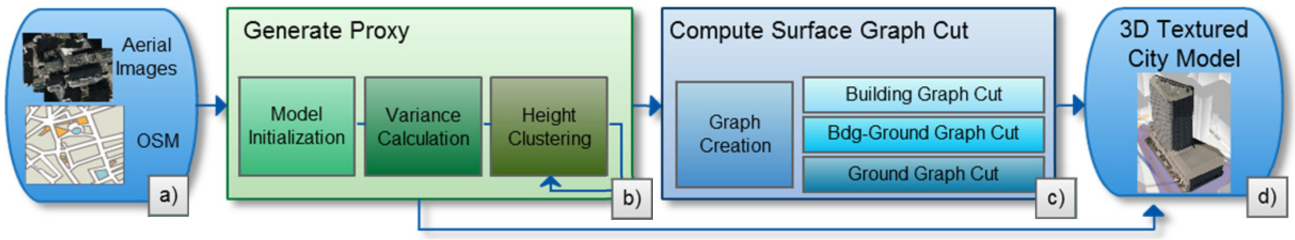


Figure 2. System Pipeline. Our system uses (a) aerial images and GIS-like input data to (b) compute a geometric proxy, (c) generate surface graph cuts, and (d) assemble textured 3D building models of large urban areas.

struction methodologies from image-based modeling and computer vision can be loosely divided into i) space carving and similar techniques (e.g., Kutulakos and Seitz [13], Matusik et al. [22], Montenegro et al. [23], Lazebnik et al. [17], Shalom et al. [32]) and ii) volumetric graph cuts (e.g., Vogiatzis et al. [41]). All of these methodologies exploit, in some form, photo-consistency, visibility constraints, and smoothness assumptions.

However, for our targeted large areas with high building density and thus a high-level of occlusion, we cannot assume a dense, complete, and un-occluded sampling of all building and ground surfaces. These facts about the input data spawn three important challenges. First, although in a typical aerial capture process each building might be at least partially observed in 25-50 images, parts of each facade might only be seen by a few images (and sometimes none at all). This relatively sparse sampling of the building walls hinders photo-consistency measures. Further, the limited visibility and high-level of occlusion also encumbers the silhouette usage and robust foreground/background segmentation for space carving and hampers the determination of the initial geometry (e.g., visual hull) for volumetric graph cuts. Second, since the captured images of building and ground surfaces may be plagued with the projections of nearby buildings, obtaining occlusion-free projective texture mapping (i.e., texture mapping without neighboring buildings unwillingly appearing on other buildings) would require very accurate geometry. Third, obtaining such very accurate geometry is hindered by camera calibration error and by the grazing angle observations of the building facades. Naïve projective and view-dependent texture mapping would produce strong visual discontinuities or would compensate for the inaccuracies by using significant blending/blurring.

Our solution circumvents the aforementioned challenges by exploiting the following inspirations.

- Buildings are, by and large, individual 2.5D structures; thus we assume each successive floor up the building is equal to or contained within the contour of the previous floor.
- Since aerial images mostly sample the roof structures of a building, we exploit photo-consistency only for determining the roof structure; for the building walls, we exploit the 2.5D assumption and stitch together the visual observations using a surface graph-cut based technique (a surface graph cut is a 2D manifold in 3D space that has been stitched together using a solution to the minimum-cost graph-cut problem); our surface graph cut assembles a seamless and visually-coherent texture-mapping of the buildings and ground surfaces despite an imperfect building proxy, projected occlusions, and camera calibration errors.
- To solve the chicken-and-egg dilemma of needing to know the geometry to solve for visibility (and needing to know visibility to solve for geometry), we exploit the assumption of having approximate GIS data (e.g., building outlines) in order to formulate simple building shape estimates which we enhance.

Our approach builds upon voxel occupancy and graph cuts (e.g., Kwatra et al. [14]) to automatically and robustly yield large-scale 3D urban reconstructions. Our largest example includes 1785 reconstructed and texture-mapped buildings spanning more than 12 square kilometers. Our system pipeline (Figure 2) takes as input a set of pre-calibrated high-resolution aerial images captured from a multi-camera cluster flying over a city (courtesy of C3Technologies), approximate building outlines extracted from a GIS provider (i.e., OpenStreetMap (OSM)) and rough initial building heights per city zone.

A coarse initial building geometry is subdivided into voxels which are then refined. Improved building outlines, heights, and roof structures are obtained by using a photo-consistency and clustering algorithm. Then, we use surface graph cuts to add the remaining visual details to the building walls and to the ground. The roof structure is sampled by many images. Thus, texture mapping the roof voxels to display additional visual details can be straightforwardly done by selecting the most head-on observations. However, the building walls are sparsely sampled. Hence, in order to create a complete, coherent, and occlusion-free colored appearance, we texture-map wall voxels using the aerial images for which a satisfactory graph cut with the roof and with the adjacent building walls is produced. Further, we solve two other surface graph cut problems in order to provide a smooth visual transition between the building walls and the ground surface as well to produce a top-down high-resolution ground surface image that is free of unwanted projections of building geometry and shadows.

Altogether, our method exploits photo-consistency only where it is highly sampled (thus less susceptible to outliers and noise) and uses a graph-cut based algorithm to stitch together a visually plausible result for the rest of the building surfaces and for the ground surface. Our examples are from a large metropolitan area (i.e., Boston, MA in USA) using a dataset of 4667 aerial images and conservative initial building outlines and height estimates (e.g., often overestimates of 50%). Our comparisons show that our results are significantly better than texturing a space-carving/visual-hull result and similar quality to crowd-sourced manual modeling efforts.

Our main contributions are

- a robust voxel- and photo-consistency based method for estimating building roof geometry,
- a surface graph cut method to not only stitch textures, but also reduce, or eliminate, artifacts due to incorrect texture overlapping, missing texture fragments, incorrect camera pose, or an inaccurate geometric proxy,
- a graph-cut based method for generating a top-down aerial view of a city free of unwanted building projections appearing on the ground, despite such projections being present in all captured aerial images, and
- a complete automatic framework that generates closed, low polygon count, textured buildings and ground that are ready-to-use in 3D city modeling and computer graphics applications.

2 Related Work

In this section, we relate our work to urban modeling approaches in procedural modeling, image-based algorithms, LIDAR-based methods, and volumetric reconstructions including graph cuts. Image-based algorithms, from computer graphics, computer vision, and photogrammetry, have generated very compelling urban reconstruction results. A recent survey by Musiaski et al. [25] provides an overview of numerous urban reconstruction techniques. Some representative works have created individual facades from images (e.g., Müller et al. [24], Xiao et al. [43], Teboul et al. [37]), individual buildings and statues (e.g., Lafarge et al. [15], Vanegas et al. [40]) and point cloud reconstructions (e.g., Liao et al. [20]). However, these methods have not produced volumetric building models (e.g., complete texture-mapped building envelopes) of large city areas. Approaches have also been proposed that use large online photo communities to perform reconstructions of popular areas (e.g., Goesele et al. [10], Agarwal et al. [1], Frahm et al. [6]). But, these results are fragmented and cannot necessarily produce all buildings in a given target area.

Numerous methods exploit LIDAR data sources. For example, Nan et al. [26] and Zheng et al. [44] provide interactive tools to improve partial scans of individual building models. Zhou and Neumann [45] provide striking results by extending dual contouring to 2.5D building structures. Poulis et al. [30] present an automatic method to reconstruct 2.5D buildings from aerial images and LIDAR data. They propose a framework using i) 2.5D graph-cuts, ii) automatic and interactive segmentation, and iii) automatic identification and reconstruction of linear roof types. Lafarge and Mallet [16] segment data into ground, buildings, vegetation, and clutter. Then, buildings are formed by fitting points to a collection of template primitives. In general, these methods, and similar ones, rely on the availability of high-resolution point cloud data, sometimes make assumptions of the roof/building geometry, and some do not produce colored/textured models – a naïve projective texture-mapping using the available aerial images will not necessarily produce good results, as shown in our results section. Shen et al. [33] presents an adaptive partitioning of unorganized LIDAR data to find high-level facade structure repetitions. This method can be used to consolidate facades but it is not designed to recover geometry. Toshev et al. [38] detect building structures from city-scale 3D point clouds and construct a hierarchical representation for high-level tasks. Also Golovinskiy et al. [11] present another approach to recognize objects in 3D urban LIDAR data using specialized clustering and graph-cut segmentation. However, reconstruction is not the focus of these last three methods.

Some methods focus on the registration of aerial images with LIDAR data or with 3D models. For example, Ding et al. [5] describe a new feature called 2DOC based on 2D corners that corresponds to orthogonal structure corners in 3D. Wang et al. [42] improve the registration by using a novel feature called 3CS which uses sets of connected lines. To create a robust registration, they first overestimate the number of line segments and then perform a RANSAC-based refinement. Frueh et al. [7] automatically texture detailed 3D models. They improve the texture discontinuities of each triangle using a classification approach and reduce the graphic card memory footprint using an atlas approach. They present nice results but with clearly visible seams between ground-base and airborne textures.

Volumetric reconstruction via space carving, graph cuts, and related methods have also received significant attention. Methods, such as space carving [13] and image-based visual hulls [22] assume the presence of many images observing the silhouette of the object. Such observations are in general not possible using aerial images of dense urban environments. Another option is using a set of ground-level images to reconstruct the facades of buildings (e.g. Gallup et al. [8] uses a high resolution video with a priori calibrated street level video and per-pixel depth map as input; Frahm et al. [6] uses a scattered set of images; Grzeszczuk et al. [12] reconstructs building facades from street level images without significant occlusions) but it is impossible to fully sample all facades and all roofs of all buildings in a large urban area. Pollard et al. [29] present a voxel-based volumetric method to detect changes in a 3D scene. Despite presenting some similar inspiration, this approach is designed to detect changes instead of find similarities.

Graph cuts have been extensively used in computer graphics (e.g., texture synthesis Kwatra et al. [14], Lefebvre et al. [18]). For volumetric reconstructions, graph cuts are applied to 3D subdivisions of space and combined with stereo processing (e.g., Vogiatzis et al. [41], Sinha and Pollefeys [34], Tran and Davis [39]). Nevertheless, these methods rely on high photo-consistency over the entire building surface and require an initial building geometry, such as the visual hull. Using aerial images to obtain the visual hull as well as sufficient samples for robust photo-consistency metrics over the entire building surface is challenging for dense urban environments. In our work, we also use graph cuts, but we define a surface graph cut that lies on building roofs and walls and on the ground surface. Further, each graph node is positioned and oriented in 3D space but is only connected to its neighboring surface elements.

Lempitsky and Ivanov [19] also use graph-cut optimizations, as well as gradient-domain techniques, to address the problem of texture fragmentation on a 3D surface. They assume i) all textures completely see the object, ii) there are no occlusions, iii) all images have the same quality (=importance), and iv) the cameras are perfectly calibrated. These assumptions allow them to simplify their cost function to only use the direction of the corresponding view and the surface normal and to discard any duplicated or overlapping texture segments. Allene et al. [2] alleviate the aforementioned equal image-quality assumption by including optimization terms to measure the effective texture resolution and the color continuity at edges between faces assigned to different (textured) images. Moreover, they use per-pixel blending to minimize the difference due to lighting conditions. In contrast, our approach tackles the problem when occlusions are frequent, camera poses are not contiguous nor have similar angles, cameras are not perfectly calibrated, and the proxy model is not guaranteed to be accurate.

Alternative approaches have been proposed. Gao et al. [9] directly operates on the points and splats/combines results to an output image without obtaining a geometric model. Mathias et al. [21] use structure-from-motion, image-based analysis, and shape grammars. The reconstruction results are promising however a grammar is required, which thus lacks automation for large-scale deployment. A related semi-automatic approach is that of Taillandier et al. [36]. However, their method has several requirements which make it not adequate for many urban areas: they only handle square buildings with slanted roofs; they require having an accurate outline of the building and not just the parcel contour or a rough approximation.

In contrast to previous methods, our work focuses on automatically obtaining complete (e.g., closed) building models of urban tall building areas (e.g., downtown, office buildings, financial districts) spanning multiple square kilometers and rely only on aerial imagery and commonly available GIS data for cities around the world. In addition to estimating a building proxy, our method enables the creation of plausible texture-mapped building models using stitched together imagery, even in the presence of imperfect geometric proxy estimates and imperfect camera calibration. Some commercial ventures, such

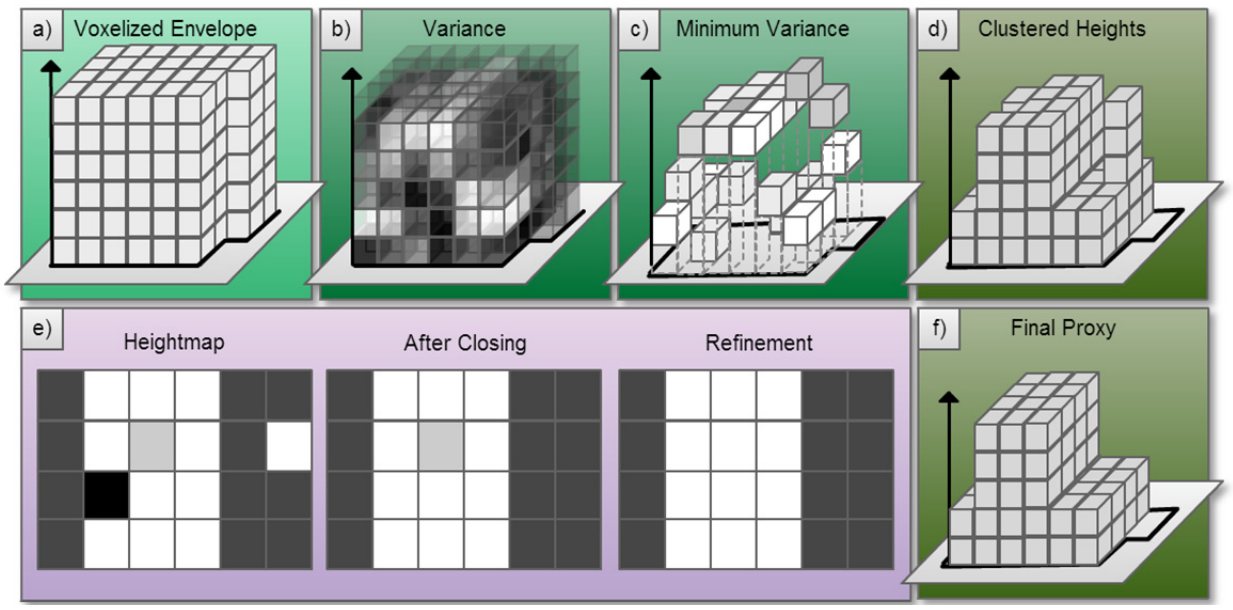


Figure 3. Building Volumes. We show the steps of our volumetric building reconstruction. a) An initial model is divided into voxels. b) The per-voxel variance of our weighted photo-consistency measure is computed. c) The most consistent voxel per column is chosen, potentially reducing building height. d) The voxels are clustered by height, e) placed in a height-map, and filtered. f) The final proxy model is obtained.

as C3 Technologies (purchased by Apple), pursue similar 3D reconstruction objectives but to our knowledge use manual-intervention and wide-baseline stereo to obtain building models, thus making widespread deployment challenging.

3 OVERVIEW

We identify two main tasks to reconstruct a city: i) find the building geometry (Figure 2b), and ii) texture the models (Figure 2c). For the first task, we could discretize the space of the whole city and try to find the geometry at the same time, but this approach would not scale since the number of voxels is linear with the size of the city (e.g., in our case it would be $O(10^8)$ voxels). Given the individual nature of each building (i.e. a building can be seen as an independent model surrounded by streets), we simplify the problem by processing each building individually. For each building, we first initialize the building with a set of voxels using the GIS data (Section 4.1) and find the photo consistency between aerial images (Section 4.2). Then, we find 2.5D building geometry (Section 4.3).

For the second task, we could use a standard view-dependent texture mapping, but this would assemble imagery by blending together fragments from many different images. Such a method does not exploit the internal consistency of each capture image and might create seams along image transitions. In contrast, we use graph cuts to stitch together imagery from as few images as possible so as to exploit internal consistency as well as produce seamless texture mapping. Given that the complexity of graph cuts (solved using the min-cut algorithm) is $O(VE^2)$, it would not scale to city level (in our case it would be $O(10^{19})$). Therefore, we also process each building individually. However, this does not completely solve the problem since the ground should be also textured, which in turn necessitates a smooth transition between building and ground surfaces. To overcome the problems of this task, we use graph cut for three different purposes: i) texturing building surface (Section 5.2), ii) improving the transition building-ground surfaces (Section 5.3), and iii) finding an optimized ground surface (Section 5.4).

4 VOLUMETRIC BUILDING PROXY

We first describe our algorithm for computing a per-building volumetric proxy. Our method initializes each building model as a grid of voxels, calculates a weighted photo-consistency measure per voxel, and clusters the voxels of minimum variance. The output is a 3D un-textured proxy model.

4.1 Appearance Editing

Each individual building is initialized as a 3D array of voxels (Figure 3a). The voxels are obtained by subdividing a vertical extrusion of a coarse estimate of the 2D building footprint. Given a building of size $[\mathrm{b_x}, \mathrm{b_y}, \mathrm{b_z}]$ and a voxel size r, we label each voxel $\mathrm{v_i}$ for $\mathrm{i} \in [1, \mathrm{N}]$ and $\mathrm{N} = \left(\frac{\mathrm{b_x}}{r}\right) * \left(\frac{\mathrm{b_y}}{r}\right) * \left(\frac{\mathrm{b_z}}{r}\right)$. For notational brevity, we assume $\mathrm{v_i}$ also refers to the 3D position of the middle of voxel i. The upper bound for N is when a voxel of size $\mathrm{r_0}$ corresponds roughly to one projected pixel. In practice, we choose values of $\mathrm{r} > \mathrm{r_0}$ in order to reduce the per-building computation time which is important when processing city-scale environments.

Building footprints and building heights, or estimates thereof, are frequently present in a city GIS's and in some navigation service databases. With regards to building footprints, one option is to use the shape of the enclosing parcel which is roughly of the same shape for dense urban areas. In our case, we make use of the increasingly popular open data repository OpenStreetMap.org. It contains top-down street, parcel, and approximate building outlines for a very large number of cities worldwide (as seen in Figure 2a and in our video). We extract building outline estimates from images such as this using image processing; in particular, we detect a loop of edges per parcel and form a closed polyline. For building heights, if not available in the GIS, we make zonal estimates (e.g., residential zone apartments are given a constant height, high-rise zones are given a higher constant height, etc.); however, the building height should be conservative (e.g., we frequently overestimate height by 50%). Photo-consistency will enable finding the actual roof heights and building outlines.

We must also establish an initial surface normal per voxel. After inspecting many buildings, we found that a good prior is to represent a building as a half ellipsoid (Figure 4, bottom). At this stage in the pipeline, the voxel normal is solely used to determine the subset of the aerial images that potentially “see” the voxel. This approximation does not directly affect the resulting building geometry but rather helps select which images are used in later stages. Because it is not known yet which voxels will be on the building surface, normals are computed for all voxels of the initial model (i.e., interior and exterior voxels). Given a building, we first fit the upper-half of an ellipsoid to the building by computing values for the ellipsoid radius and ellipsoid coefficients a, b and c. Then, given voxel v_i, we compute the voxel's initial normal as

$$n_i = \left(\frac{2v_{i_x}}{a^2}, \frac{2v_{i_y}}{b^2}, \frac{2v_{i_z}}{c^2}\right) \quad (1)$$

Given voxel positions and normals, we obtain the color c_{ik} for voxel i observed by camera k. To support different voxels sizes (both when voxel-to-camera distance varies amongst the aerial images and when purposefully working with larger voxels to increase reconstruction performance), we project the voxel onto camera image k, estimate the size s_{ik} of voxel i on camera image k and grab a Gaussian weighted footprint of pixels as

$$c_{ik} = \sum_{t \in [(-s_{ik}, -s_{ik}), (s_{ik}, s_{ik})]} (proj_k(v_i) + t) e^{-\|t - proj_k(v_i)\|^2 / 2\sigma_c} \quad (2)$$

where $proj_k(\cdot)$ returns the projection of its argument onto camera image k and σ_c is the standard deviation of the Gaussian. Given s_{ik}, σ_c is obtained by the known estimate $0.3\left(\frac{s_{ik}}{2} - 1\right) + 0.8$

4.2 Variance Calculation

Starting with the initial model of a building, we search for a subset of voxels that are photo-consistent amongst the aerial images observing the building. We assume strong photo-consistency for a voxel implies it is on the actual building surface. As the measure of photo-consistency, we use the weighted variance of the color of a voxel's projection on different aerial images (Figure 3b).

In preliminary experiments, we investigated several measures for evaluating whether a voxel is on the building surface. We attempted using color-based segmentation of aerial images and/or the weighted sum of the measures of photo-consistency, local surface planarity, and local supportability (i.e., probability that a voxel is needed because another higher-up voxel will be selected). However, we observed that the various variants of this combined metric were not robust to noise/errors and in practice over-constrained voxel selection. This is primarily due to the relatively sparse (and often at grazing angles) sampling of building walls. As mentioned in the introduction, we did however observe many visual samples and significant photo-consistency amongst voxels on the building roof surface which led us to mostly rely on them.

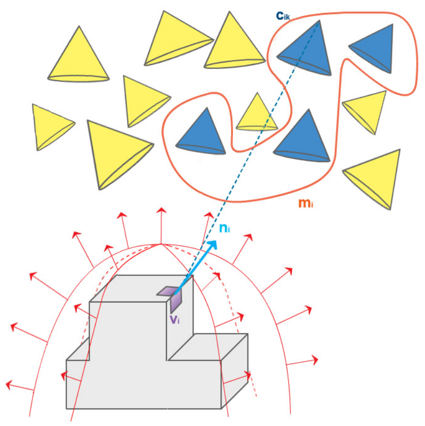


Figure 4. Variance Calculation. Using the initial voxel normals n_i for a voxel v_i, we determine the variance of our weighted photo-consistency measure of the subset of cameras, such as c_{ik}, that best see the voxel.

Our method transforms all aerial images to HSL color space and uses the H and S channels. We use only the H and S channels in order to ignore the effect of changing daylight illumination and, to a lesser degree, the effect of shadows. Further, we explicitly weigh variance by the inverted building height of a voxel. Hence, given an approximately tied variance, the vertically higher voxel is chosen. Numerically, our voxel variance measure is computed as

$$m_i = \frac{v_{i_z} + b_z/2}{b_z} \left(\sum_{k=1}^{s_i} c_{ik}^2 - \left(\sum_{k=1}^{s_i} c_{ik}^2 \right)^2 / s_i \right) \quad (3)$$

where it is assumed the building is centered at the origin, the first term computes the ratio of the voxel's vertical height (assumed to be along the z-axis) to the building's z size, and s_i is the number of camera images that have a line of sight to voxel i.

In order to improve the variance calculation, we use the initial footprints to account for the potential occlusion of neighboring buildings. Specifically we create a mask m_k by rendering the building from the point of view of a camera k pointing towards the building; the building is rendered in white and the background in black. When computing color c_{ik}, we check in the corresponding mask whether the image pixel is white (unoccluded) and should be used, or black (occluded) and should be discarded.

4.3 Height Clustering

In aerial images, roofs are expected to be viewed by more cameras than facades (i.e., more photo-consistent). Thus, we find the height of each column by searching for the column's voxel with the lowest variance m_i. We use this information to assemble the final building proxy (Figure 3c). If we observe the building from a side, the voxels should collectively exhibit a compact distribution around the different heights of the buildings. Hence, we can use 1D k-means clustering to find those different building roof heights (e.g., $k = \{1,2,3, \dots\}$). Since the optimum value for k is not known a priori, we estimate it using a heuristic that works well in practice. Starting at $k = 1$, we increase k until we find that at $k + 1$ the clustering error reduces by no more than c_e percent. In preliminary experiments, such a clustering algorithm worked well, yielding buildings with 1 to 5 different roof heights. For our results, we set $c_e = 0.3$. After clustering, our method selects the voxel per column whose height is closest to the corresponding cluster's mean height (Figure 3d).

After clustering, we place all voxel heights into a height map image. Starting with the uppermost cluster, our algorithm performs a per-cluster morphological close operation [28] (i.e., dilate and then erode) in order to remove small islands of the current cluster type and to fill-in small gaps. We also perform an in-filling refinement step to remove any remaining single-voxel holes with no height/cluster assignment (i.e., we find the most popular cluster assignment of the neighboring voxels and assign that value to the missing voxel).

Voxels physically below the filtered minimum variance voxels are marked. Then, all exterior surface voxels are selected as being part of the building envelope (Figure 3f). Although voxels are small, we reduce jaggedness by adding quadrilaterals to connect corners of adjacent voxels on an off-axis (i.e., diagonal) building surface. It is worth noting that the final proxy's outline will not necessarily match that of the initial conservative building estimate.

Finally, the voxel normal is recomputed for each exterior surface voxel by summing up the vectors from the voxel center to each existing neighboring voxel, reversing the normal direction, and normalizing the vector. Afterwards, the normals of all voxels are averaged using a Gaussian weighting of the nearby voxel normals. Succinctly, this is computed as:

$$n_i = \sum_j \gamma_{ij} norm\left(-\sum_k \delta_{jk}\left(p_j - p_k\right)\right) e^{-\|p_i-p_j\|^2 / 2\sigma_n} \quad (4)$$

where σ_n is the standard deviation of the desired smoothing neighborhood, $\delta_{jk} = 1$ if v_j and v_k are adjacent, $norm(\cdot)$ returns the vector normalized version of its argument, and $\gamma_{ij} = 1$ if v_i and v_j are within $2\sigma_n$ voxels of each other (e.g., 95% of the neighbors that affect normal averaging are considered).

5 Surface Graph Cuts

In this section, we define surface graph cuts as well as describe our multiple uses of them. Graph cuts can be used to solve problems such as image stitching and image segmentation. To solve the stitching problem, a 2D graph is created where each vertex represents a pixel and edges connect adjacent pixels with a calculated weight (e.g., the color difference). The best stitching possible will be the one that minimizes the visible transitions (i.e., the minimum cut through overlapping areas -- Figure 5h). We extend this idea to not just define a flat image but instead pixels on the 3D surface formed by the visible faces of the building, the interface between building and ground surfaces, and the ground surface. Conceptually, this can be viewed as covering the building with pieces of cloth. Each image is a piece of a cloth that partially covers the building. We try to cover the whole building with the least noticeable transitions. The challenge is in choosing cloths and in how to cut them.

5.1 Definition

A surface graph contains the visible faces of a volumetric building proxy (Figure 5a) and/or of the surrounding ground. Since, for reconstruction performance reasons, we typically chose a voxel size that projects to larger than one pixel, each exterior (i.e., visible from the outside) voxel face is subdivided into SxS subfaces (in Figure 5b, $S = 2$) to ensure the final model is textured at near the original image resolution despite using lower-resolution voxels. In each visible face of each voxel, we place a SxS array of vertices (Figure 5c). Each vertex v_a is then connected to its neighbor v_b by an edge e_{ab} (Figure 5d) to form the 3D graph where a graph-cut will be applied. Thus, the surface graph $G = \{V, E\}$ is composed of vertices $V = \{v_a\}$ for $a \in S^2 N_S$ (where N_S are the faces of the voxels that are in the surface) and edges $= \{e_{ab}: v_a$ and v_b are adjacent$\}$.

Within each graph vertex v_a, our system stores

- q_a: 3D position of the graph vertex,
- n_a: surface normal in the vicinity of the subface,
- k_a: camera image id to use for this voxel,
- c_a: current color of the graph vertex, and
- p_a: potential color of the graph vertex.

A graph cut defines a smooth visual transition between two adjacent surface *patches*. Each of the two patches is a subset of the surface graph that has the same camera image id (Figure 5e and Figure 5f). These two patches overlap in a region (Figure 5g). The graph cut process will find the trajectory, in this overlapping area, along which the sum of the color differences between the corresponding pixels of the two source camera images is minimal (Figure 5h).

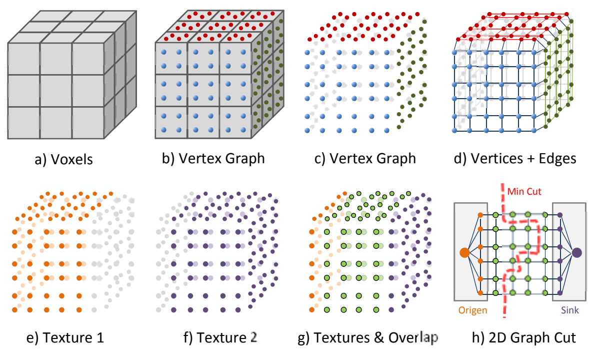


Figure 5. Surface Graph Cut. a) Voxels, b) voxels showing graph vertices, c) vertices, d) vertices with edges, e) vertices seen by an image 1, f) vertices seen by an image 2, g) vertices that see image 1 and image 2 are in green and are where graph cut will be applied, and h) a 2D graph cut .

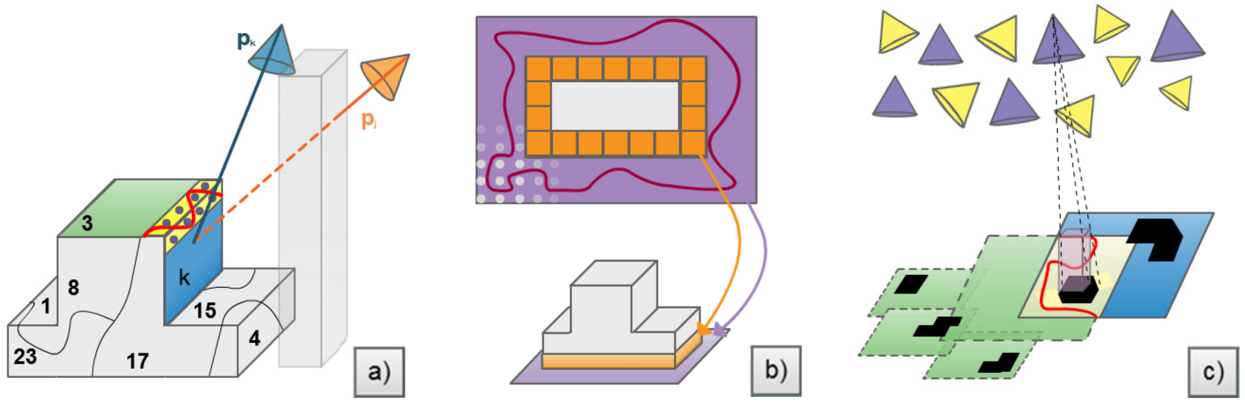


Figure 6. Applications of Surface Graph Cuts. a) We show several patches over a building surface. Each patch is obtained by grouping adjacent subfaces best observed by the same camera while taking visibility into account (e.g., patch k is best observed by camera p_k because p_j is occluded. In this step, patch 3 and k are joined as in Figure 5h. b) Another surface graph cut is defined and computed at the boundary of the building with the ground surface. c) Finally, a ground surface graph cut is also performed so as to obtain a seamless and free-of-projected-buildings ground texture.

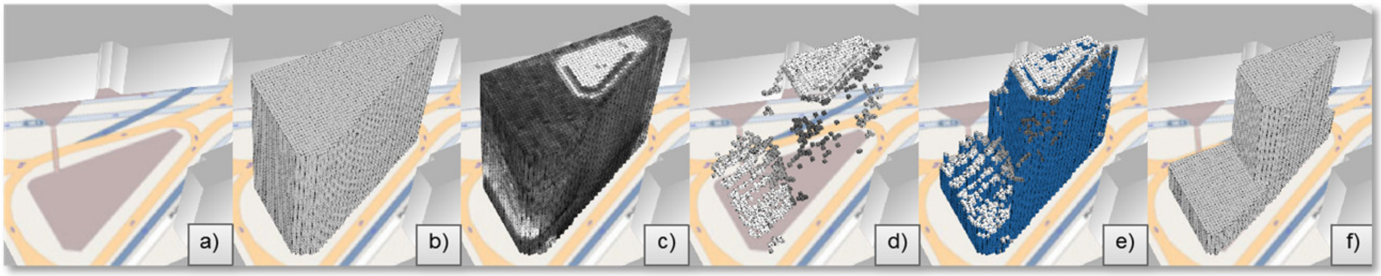


Figure 7. Volumetric Reconstruction Pipeline. We show example images from a volumetric building reconstruction. a) OpenStreetMap input image. b) Voxelized-version of the extruded building footprint. c) Per-voxel weighted photo-consistency variance (white = low variance). d) Selection of per-column voxel with lowest variance. e) Vertical support added beneath each per-column selected voxel. f) Final proxy after clustering and filtering.

To avoid re-creating the graph for each texture, we create the graph just once and update the weights, origin, and sink before calling the min-cut procedure. To efficiently compute our large min-cuts (e.g., O(10^6 vertices)), we use the augmenting path algorithm of Boykov and Kolmogorov [4] which in practice is significantly faster than other methods. To calculate the cost, we perform color differences in perceptually linear LAB color space in order to improve the perceived transition from one texture to another and not just reduce the numerical color difference (i.e., reducing the Euclidian distance in this color space maps to a perceptual improvement). We define the matching quality cost C between two adjacent vertices s and t that belong to two different patches P_1 and P_2 as

$$\mathrm{C}(\mathrm{s},\mathrm{t},P_1,P_2) = \left|\left|P_1(\mathrm{s}) - P_2(\mathrm{s})\right|\right| + \left|\left|P_1(\mathrm{t}) - P_2(\mathrm{t})\right|\right| \quad (5)$$

where $P_i(*)$ evaluates to a LAB color.

5.2 Building Surface

We solve the graph cut problem for the building surfaces resulting in the best seamlessly stitched texture-map over the building surface (Figure 6a). First, we compute which cameras are visible from each graph vertex and choose the visible camera that best samples the vertex's surface fragment. Since we have a very large number of vertices (e.g., over 100,000 per building), we use the graphics card to quickly determine which are visible from each of a nearby set of camera viewpoints. For efficiency, we render each voxel as a color-coded quadrilateral. From all the cameras k, at position g_k, that see a particular voxel and all its subfaces/vertices, the camera k_a for which $(g_k - q_a) \cdot n_a$ is maximal is chosen; e.g., $k_a = k$.

Second, spatially-adjacent vertices with the same camera image id are grouped into patches and sorted by size. To assist with reducing the effect of image-to-image illumination changes and calibration errors, we wish to have as few textures and graph cuts as possible. Thus, we group same-image-id vertices. We also sort them by area from largest to smallest because the largest group is mostly likely to reference the best aerial image. Empirically, buildings are stitched together from 3-10 different aerial images.

Third, our method assembles the surface graph cut starting with the largest patch. Given the current processed vertices, the system iteratively searches for the largest adjacent patch. An overlapping frontier is defined within the two patches. Although we could use the entire overlapping area to find the graph cut, we limit the overlapping area so as to keep most of the current processed vertices intact. Before calling min-cut, we update the weight of each edge: the vertices that have been processed are connected to the origin of the min-cut and their weights are set to infinity (i.e., to not be cut), the edges of the vertices within the transition region are updated with the cost C (between the current color and the new potential one), and vertices that belong to the potential texture but do not overlap, are connected to the sink and their weights set to infinity (i.e., also to not be cut) -- as in Figure 5h. Our system uses min-cut to search for the cut that minimizes the visual image transition from one patch to another one. This step is repeated iteratively until all vertices have an assigned camera image id.

We choose this greedy approach over other global optimizations because i) we do not try to minimize the transition between vertices but between patches, ii) a global (patch) optimization would require an exhaustive/stochastic exploration, iii) it is guaranteed to converge, and iv) it fits the requirement to keep the number of patches as small as possible to minimize the change-of-illumination issue.

5.3 Building-Ground Surface

Next, we solve the graph cut problem for connecting the building surfaces to the ground surface (Figure 6b). We extend the building surface by generating a ring of voxels around the ground-level height of the building such that the top most face of each extended voxel coincides with the existing ground surface (i.e., the voxel center is essentially slightly "below ground"). Even though the width of the ring can be altered, we use a constant value for all our examples. For each of the newly created voxels, we assign a camera image id to it. This is done by finding the closest voxel on the building surface and copying the camera image id to each of the S^2 graph vertices of the new voxel. To build the local ground surface, we use the same extended building surface vertices but calculate their color using the improved ground surface image (see next section).

We define a single building-ground graph (per building) with a source node inside the footprint and the exterior ring of voxels connected to a sink node. The graph cut computes the smoothest transition from building wall textures to ground surface images. The cut is constrained to lie on the ground surface so as to prevent the building textures from changing.

5.4 Ground Surface

To produce an improved top-down view ground surface image, we stitch together the most downward facing aerial images (Figure 6c). In a manner similar to 2D texture and image synthesis, the aerial images are pieced together sequentially in random order -- the order does not matter as long as the ground surface is fully sampled. Since the graph of one ground image is very large (e.g., over a million graph vertices), only a subset of the overlap region between the currently stitched image and a new image is used. A graph cut is calculated within the overlap region and stored.

To avoid the appearance of building surfaces projected onto the ground outside of the building footprint, we make use of the building proxies. We render each building proxy "in black" from each image's center of projection and onto the aerial image. Then, we explicitly prevent the graph cut from using, or "going through", the building by placing very large cost penalties when choosing to transition to a building pixel. Although it is not guaranteed that all ground surface points are observed, unobstructed, from an aerial viewpoint, in practice it is possible. The final result is one single coherent, occlusion-free top-down image of the city.

6 Implementation

Our system implementation includes several optimizations to improve computation time, memory usage, and rendering performance. In our dataset, one aerial image pixel projects to roughly 0.5 meters. To reduce the proxy computation time, we typically choose a voxel size of $r = 2$ to 4 meters. To compensate for this subsampling during graph-cut based texturing, we subdivide voxel faces with $S = 4$ to 8, thus returning to approximately one pixel resolution. The graph cut computation time is increased but only in the overlap regions.

Rather than having each building require access to multiple textures, we create one custom texture atlas per building. This design choice is also motivated by the fact that we cannot load all the aerial images needed for a zone of the city into texture memory, and even less all aerial images (e.g., all aerial images amount to about 280GB of raw image data which even with texture compression cannot be loaded into texture memory). A typical building's texture atlas requires 1MB of space and contains the entire pixel data needed for the building surface graph cuts and the building-floor surface graph cuts. The ground surface is composed of a grid of texture-mapped quadrilaterals. All texture atlases and ground textures are loaded/unloaded on demand by the system.

The system parameters are tuned once and are used the same values for the reconstruction of all buildings. To use a completely different set of images, it would take 10-30 minutes to manually find the optimal values. The list of these parameters is: c_e which is the percentage height clustering error (Sec 5.3): a low value makes the buildings have too many levels, and a high value may cause building details to be missed -- as long as noise is not high, our system is not sensitive to this value; r is the voxels size (Sec 6.1) -- it is a tradeoff between quality and speed (limited by the size of a pixel in the image set); S is the voxel size subdivision for the graph cut (Sec 6.1) -- the value can be calculated to ensure maximum resolution; graph cut overlapping area (Sec 6.2) and building-ground ring width (Sec. 6.3), defines the region where the graph cut will be performed. As long as these values are reasonable our system is able to find a seamless transition.

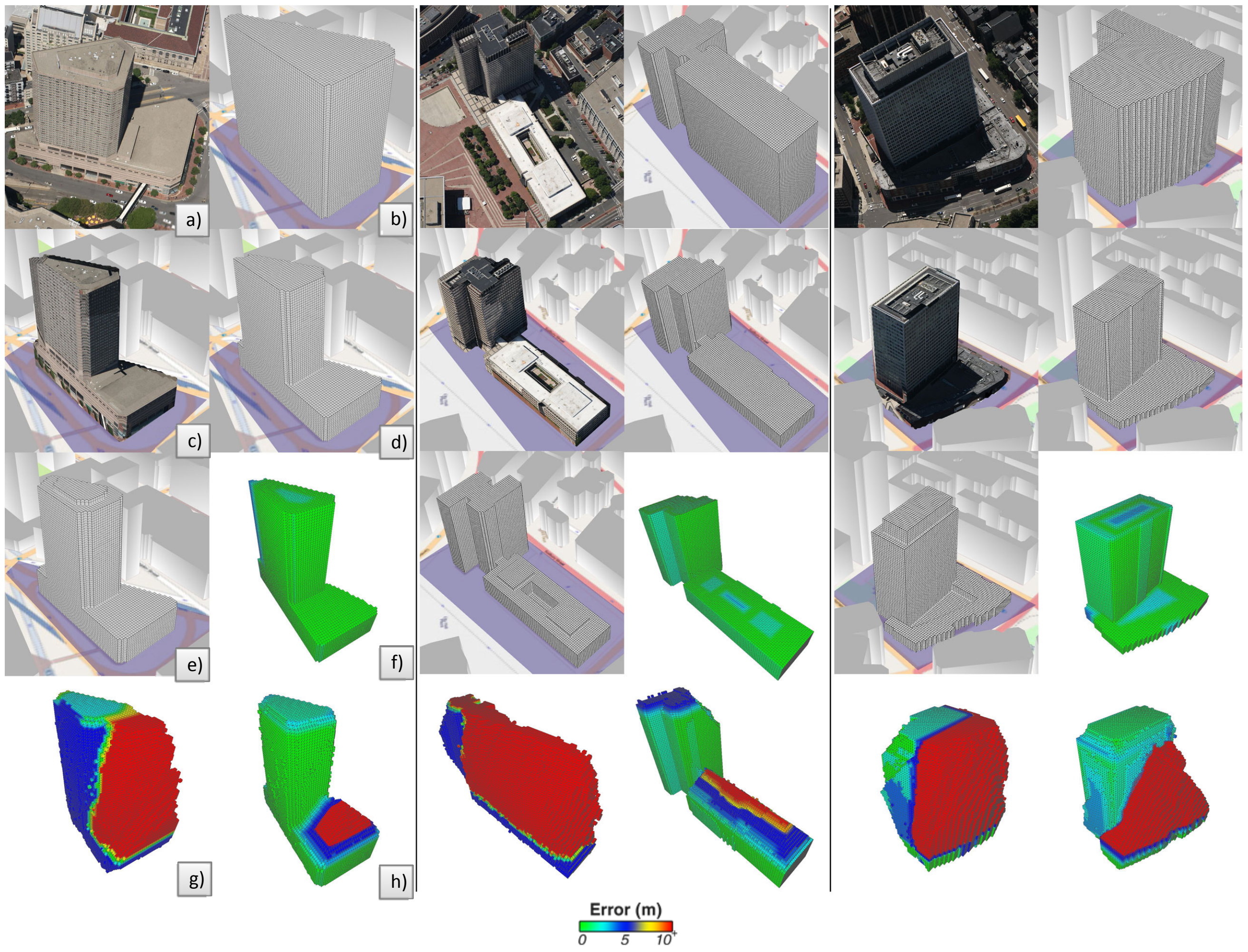


Figure 8. Building Graph Cuts & Space Carving. a-d) Aerial picture, initial voxels, our textured result, and our calculated model with no textures. e) Ground truth and f-h) show Hausdorff distance (color map: green=0m, blue=5m, red=10m or more) between ground truth and our proxy, graph-cut space carving, and manual-segmentation space carving (see text).

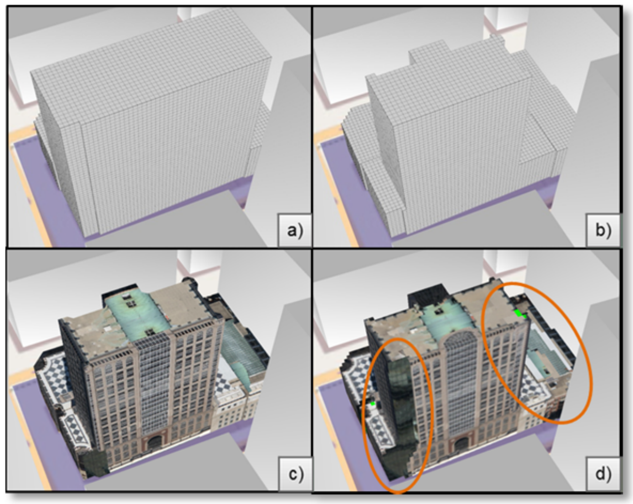


Figure 9. Texture Mapping Comparison. a) Initial model. b) Calculated proxy model. Mismatch/discontinuities occur due to geometry/calibration errors that are in general unavoidable in a dense city. Yet, c) our surface graph cuts compensate for inaccuracies and produce a continuous/coherent texturing, better than d) standard projective texture mapping

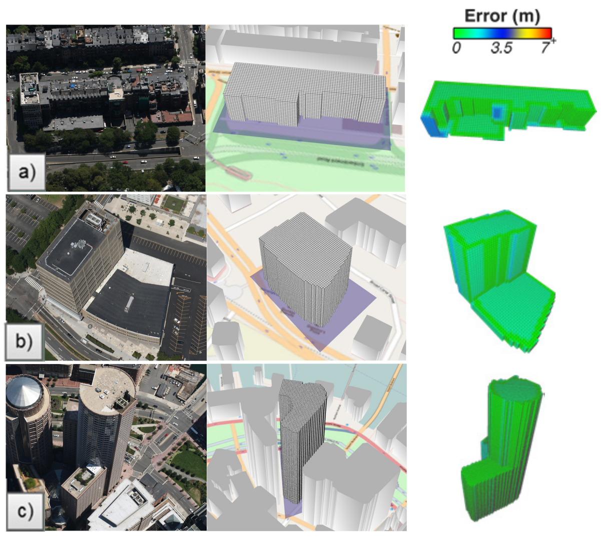


Figure 10. Building Reconstruction for Various Building Sizes/Complexities. For a) small building (20m), b) medium size building (90m) and c) large building (180m), (left) aerial images, (middle) initial voxels, and (right) reconstruction error using Hausdorff distance (green=0m, blue=3.5m, red=7m or more).

7 Results and Analysis

We have used our method and system for several large urban examples. Figures 7-18, supplementary figure page, and our video visually show our results and analysis. Our system is implemented in C/C++ and uses Qt/Boost/OpenCV. It runs on a Windows PC with Intel Xeon 2.53 GHz and NVIDIA GTX 580 graphics card. Our example dataset consists of a grid of about 58 by 19 aerial viewpoints over central Boston, MA (USA). At each viewpoint, a camera cluster takes 5616 x 3744 resolution images in five directions: one direction straight-down, and 4 diagonally downward facing directions at about 90-degrees from each other when projected on the ground plane (note: our method makes no assumption about the spatial and angular distribution of the camera views). This totals 4667 images from pre-calibrated viewpoints. The area has 1785 buildings assumed to lie on a flat ground plane. We set the default initial building height to 35 meters (assumed residential zone height). Medium-height high-rise zones are set to an initial building height of 125 meters and tall high-rise zones are set to initial overestimated building height of 250 meters.

There are two user parameters, voxel size r and texture size per voxel S. As described before r defines the voxels size and we found empirically $r = 2$ or $r = 4$ is a good balance in time and reconstruction accuracy. The parameter S can be calculated from r to use the maximum resolution of the images (user can decide to decrease it to speed up the process).

There are two building height clustering parameters: the threshold to discard the column variability and c_e which defines when to stop the clustering process. In our examples, the first parameter is set to two times the standard deviation and the latter to 0.3.

Finally, there are two more parameters regarding the surface graph cuts that depend on how much the images overlap. In our case, the amount of overlap between patches and the overlap region between building and ground textures are both set to 4m.

Reconstruction time depends mostly on the voxel size r and subdivision factor S. For our dataset, a "half resolution" reconstruction (e.g., $r = 4$ and $S = 4$) takes 22 seconds per building on average (10 hours total time). A "full resolution" reconstruction (e.g., $r = 2$ and $S = 4$) consumes 109 seconds per building (51 hours total time). The timing includes local file I/O. A typical building has from 15 to 130 contiguous patches (of the same

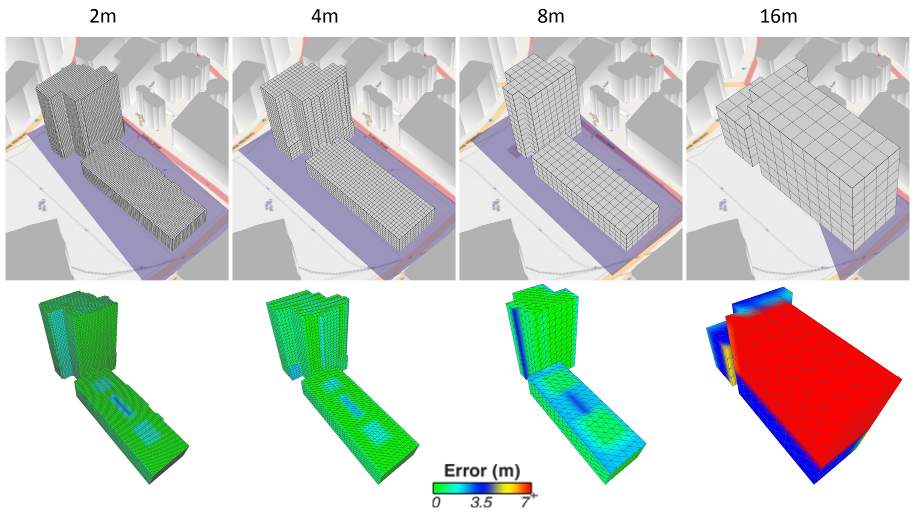


Figure 11. Result Comparison of Different Voxel Sizes. From left to right, we increase the voxel size. When the size is too large, reconstruction fails. When the size is small, the reconstruction presents similar results but excessive processing might occur. Hausdorff distance error: green=0m, blue=3.5m, red=7m or more.

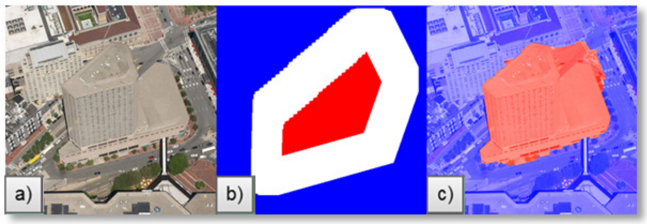


Figure 12. Graph-cut Space Carving. To perform space carving, as in Figure 8g, we use a) an initial image, b) perform automatic labeling (using the initial voxels as masks), and c) calculate a graph-cut segmentation.

image id) before graph cut application and 74 patches on average. A representative building graph has about 150k vertices, 300k edges, 80k triangles before grouping voxels for rendering and 5k triangles after grouping voxels. The ground graph is at pixel resolution and the integrated ground graph cut solution is stored in a grid of 4x4 12MP images (so that the 16 tiles can fit in texture memory and leave space for building texture atlases).

Memory requirements depend on the stage in the pipeline. Building geometric reconstruction requires about 100MB and can be reduced to less than 1MB per building after processing. Per building graph cut processing requires less than 200MB and the atlas creation requires less than 850MB (the requirement is higher since the images are loaded at maximum resolution).

7.1 Building Reconstruction

We show in Figures 7-10 several examples and comparisons for individual building modeling. Figure 7 contains intermediate results from the volumetric reconstruction process of an example building. Figure 7a has a close-up of the OSM street map used as input. Using an image processing algorithm, we find the building outline and choose a default medium high-rise height in this zone. In Figure 7b, we show the initial volumetric approximation subdivided into voxels. Figure 7c shows the calculated per-voxel variance – it is computed for all voxels but only the exterior voxels are visible. Nevertheless, the photo-consistency of the upper roof structure is evident. Figure 7d

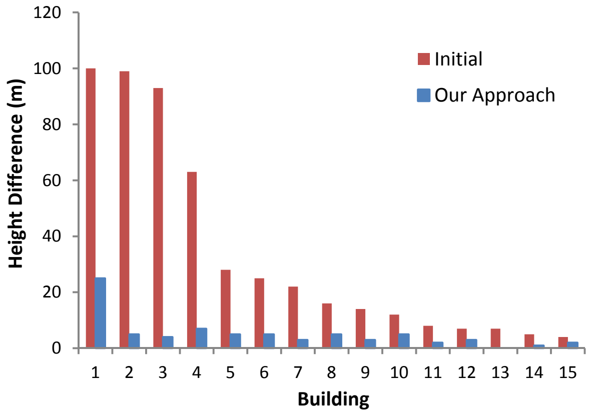


Figure 13. Reconstructed Building Height vs. Ground Truth. For 15 buildings, red bars represent the difference between the initial model height and ground truth. The blue bars indicate the difference between our refined model and ground truth.

shows the voxels with minimum variance per voxel column, which begins to reveal the building structure. In Figure 7e, we also draw all the voxels beneath each selected minimum variance voxel. Finally, Figure 7f shows the proxy model after clustering and filtering. This same process is repeated for all buildings.

In Figures 8a-d and 9a-c, we show the initial volumetric approximation, the computed proxy model, and the textured result after surface graph cut processing. Our approach is able to produce reasonable proxies for this variety of building shapes. For comparison, we show in Figure 8e the ground truth (obtained by manual modeling) and in Figures 8f-h the accuracy of several reconstructions is compared to ground truth using Hausdorff distance: we show the reconstruction error of our proxy (8f) and two versions of space carving (8g-h). As one can observe, the reconstruction error for our proxy is small. To create the first version of space carving, we use Graph Cut Segmentation [31] (as explained Figure 12) to automatically segment the foreground (i.e., the building in view) from the background (i.e., everything else). For the second version, we manually perform

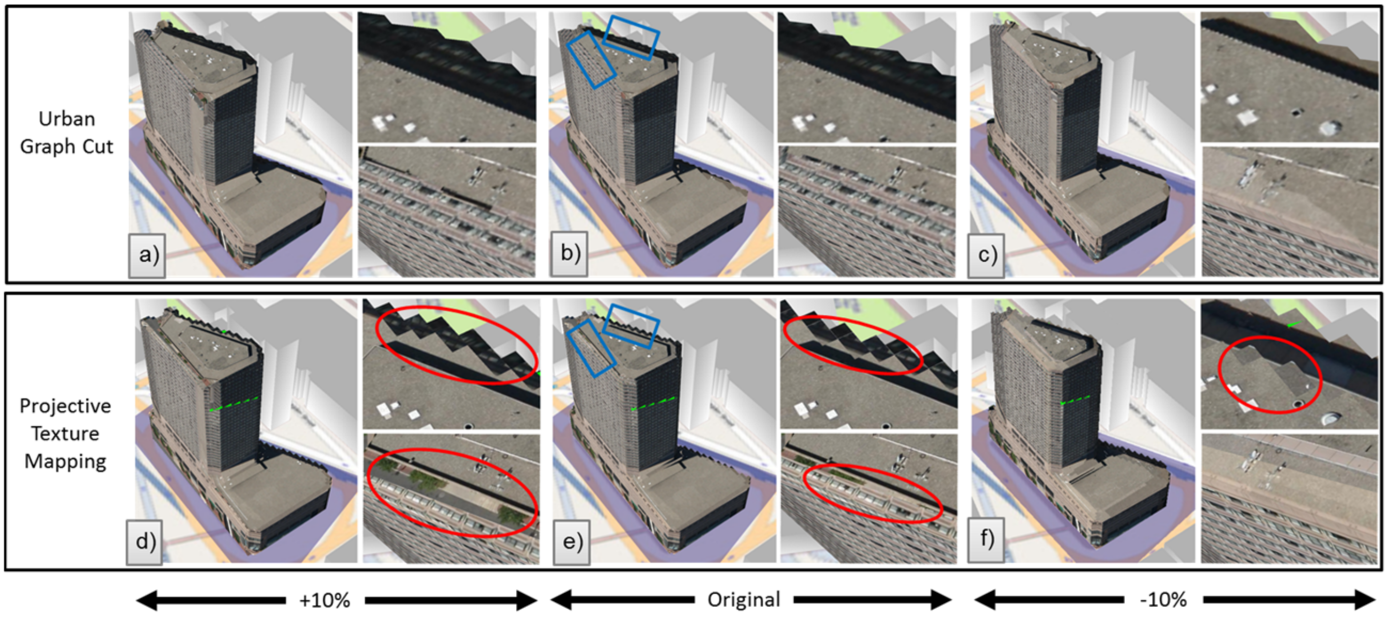


Figure 14. Graph-Cut vs. Projective Texture Mapping. Comparison of our graph cut algorithm with projective texture mapping for the original building and two altered proxies: building expanded +10% in all directions with a random noise in the height map of ±5m (left) and collapsed -10% in all directions with a random noise in the height map of ±5m (right). Our approach creates a seamless texture transition from facade to roof. In fact, as compared to projective texture mapping, it reduces the ill visual artifacts in all cases as can be seen by our results in the top row.

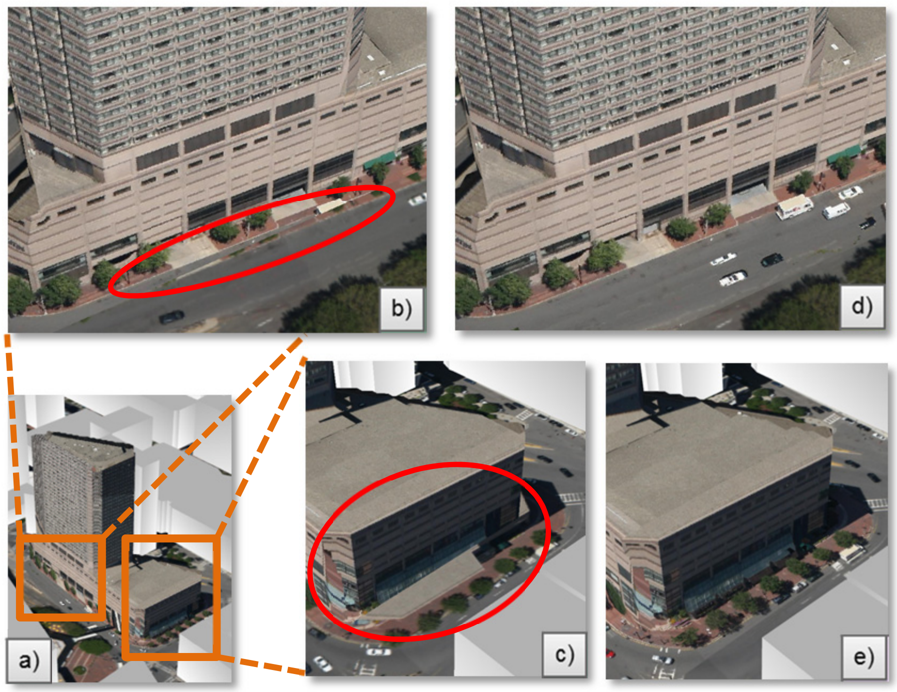


Figure 15. Building-Ground Surface Graph Cuts. a) We show two close-ups of this building. b-c) With projective texture mapping, there are discontinuities, missing content, and building projections at the boundary between the building and the street. d-e) Our building-ground surface graph cuts are able to find a smooth transition between the two structures and produce a coherent and visually plausible appearance.

the segmentation using a painting tool -- a task that it is impractical for large-scale urban reconstruction (e.g., it took between one hour to two hours to create the 25-50 masks of each building). Nevertheless, despite perfect segmentation we found that in general the obtained building reconstruction was inferior to ours. This is due primarily to the relatively sparse image sampling of each building and to the camera viewpoints being "above" the city (e.g., a distant camera would theoretically see the building more from the side but the view is most likely be occluded by another building).

In Figure 10, we present the reconstruction for buildings of different sizes and complexities. Figure 10a is a small building of 20m height, 10b is a medium size building of 90m height, and 10c is a large building of 180m height. For each building, we show its picture, the initial proxy, and the Hausdorff distance between refined proxy and ground truth. The absolute reconstruction error is approximately constant regardless of the size of the building although the defects are more visible in the small buildings. The error would, of course, be larger if there are not enough images that capture the building.

Figure 11 shows the impact of voxel size r in the reconstruction process. When the voxel size is too big our method is not able to reconstruct the building. When the voxel size is small, the vertical sampling is dense enough to find low variance points and the reconstruction can be performed. However, if the value is too small, excessive processing might occur.

Figure 13 summarizes the error in reconstructed building height as compared to ground truth (gathered from Wikipedia) for 15 well-known buildings. The average initial height error is 72%. Our system reduced the building height error to an average error of 1%-3% with a 95% confidence interval.

7.2 Surface Graph Cuts

The impact of our surface graph cuts is observed in Figures 9c-d, Figure 14, Figure 15, and Figure 16. Figure 9d contains the result of a naïve projective texture mapping. The imprecision in the proxy model, camera calibration, and the high-level of occlusion with neighboring buildings makes it challenging to obtain a perfect texture-mapping. Our additional use of (multiple) building surface graph cuts is able to compensate for these imprecisions and produce a visually-plausible approximation to the building's appearance (Figure 9c).

Figure 14 contains a comparison of our graph cut algorithm with projective texture mapping over the proxy. We compare the original building (middle) with two altered proxies to see how the proxy error affects the texture step. To create the altered

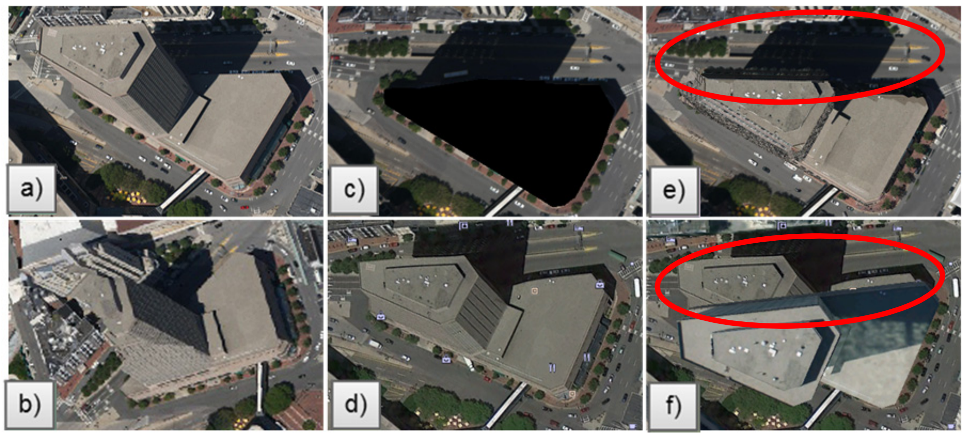


Figure 16. Ground Surface Graph Cuts. a) A downward looking original aerial image in our dataset (note occluded roads). b) Visual artifacts of using a naïve graph cut due to ignored inter-building occlusions. c) The result when using our ground surface graph cut method -- our graph cut strategy found content from other images to fill-in road pixels with building projections. d) An image of the ground surface from Google Earth with no building proxies. e) Our method using building proxies and the ground from 'c'. f) Using Google Earth imagery in projective texture mapping with buildings yields similar bad artifacts as in 'b'.

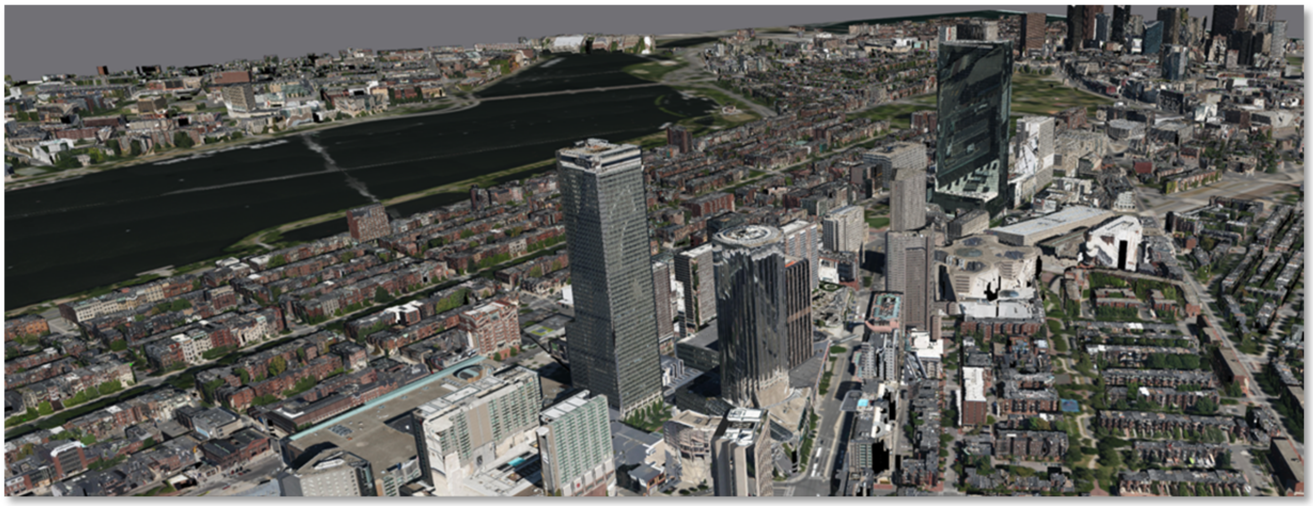


Figure 17. Full Dataset View. We show a bird's eye view of the textured 3D model produced by our system.

proxies we expanded the original building in all directions of the building +10% (left) and we collapsed in all directions of the building -10% (right); in both cases we added a random noise of ±5m in the height map. As observed in the top row, our approach manages to make less visible the error in the transition in the top images. Moreover, our approach compensates for the incorrect proxy and is able to eliminate the unwanted appearance of content (e.g., sidewalk, bushes, and side walls). In this example, it is accomplished by automatically extending the wall texture to meet the roof texture, thus producing a transition with reduced visual artifacts – however, the solution while smooth might not be physically correct. Our technique cannot always produce an improvement (i.e., compare bottom right picture of 14c with the bottom right picture of 14f). However, the smoothness of the image transition is never worse than the original.

Figure 15 contains a comparison of building-ground surface graph cuts. For the building in Figure 15a, Figures 15b and 15c show the result using our proxies and standard projective texture mapping. By enabling the computation of building-ground surface graph cuts, we are able to improve the coherence at the interface of the building and ground surfaces, as is seen in Figures 15d and 15e. In particular, notice the discontinuity of the roads and cars in Figures 15b and the projection of the extra roof surface in Figure 15c – both of which are eliminated in our result.

Figure 16 contains an example of the benefit of our ground surface graph cuts. Figure 16a contains the initial top-down view of an example area (we choose a camera with a view direction that is closest to the vertical axis). Observe how the building in the middle occludes some of the nearby roads and buildings. Figure 16b contains the result of a naïve graph cut without taking into account the buildings proxies – notice the disturbing visual artifacts despite the attempt of minimizing neighboring pixel differences with the graph cut. Figure 16c shows the result of our ground surface graph cut: buildings are not rendered on purpose and the occluded road pixels are automatically filled-in using content from other images. Figure 16d contains an image of the ground surface from Google Earth. Figure 16e shows the visual quality of our method using proxies and the ground surface from 'c'. In contrast, using Google's ground images (Figure 16f) yields similar disturbing artifacts as in 'b'.

7.3 Urban-Scale Reconstruction

We show in Figures 1, 17, 18, and supplemental figure page several bird's eye views of urban-scale examples (i.e., a fragment or portion of a city). Figures 1 and 17 show views of Boston reconstructed using our method. Figure 18 shows some close-ups of several city areas and the views using Google Earth, including its crowd-sourced buildings. It is important to note that Google Earth is using a *different image set* than ours though qualitatively similar and its models are all manually created. Our method is able to automatically produce good geometric proxies and to use surface graph cuts to stitch together the aerial imagery yielding visually effective texture mapping.

7.4 Limitations

Our approach is not, however, without limitations. First, our 2.5D assumption is applicable to most urban structures but not all (e.g., bridges or very modern building structures). Our 2.5D reconstruction currently only produces flat roofs – thus our method can process a building with a slanted roof but it would be simplified to a flat roof. Second, our method cannot robustly resolve uncertainties introduced by shadows and/or by dark building albedos. Our reconstruction method functions well only if the chroma channel of the area is sufficiently strong. Third, the sparse image sampling may prohibit certain geometric structures from being accurately reconstructed using a volumetric approach. While Google Earth renderings may be superior in some cases, our results are automatic and thus can be viewed as the final product or could be a first step for later refinement. Fourth, we have assumed for our results a flat ground plane (that mostly holds for Boston downtown). Fifth, our method focuses on the reconstruction of "tall building areas" (e.g., financial districts) with high density.

8 Conclusions and Future Work

We have presented an automatic urban-scale modeling approach using volumetric reconstruction from aerial calibrated images with surface graph-cut based texture generation. Our method generates building proxies using voxel and color consistency, exploits surface graph-cuts for recovering occluded facades and ground imagery and for assembling a seamless plausible texture mapping, and outputs 3D urban models comparable to other public systems.

We list several future work items. First, an improvement is to close the loop between graph cut calculation and proxy computation; e.g., an iterative process going between refinement of the proxy and re-computing graph-cuts. Second, to handle slanted roof buildings we plan to refine the clustering step to differentiate the case where the k-means clustering presents a high error value. In that case we plan to find the

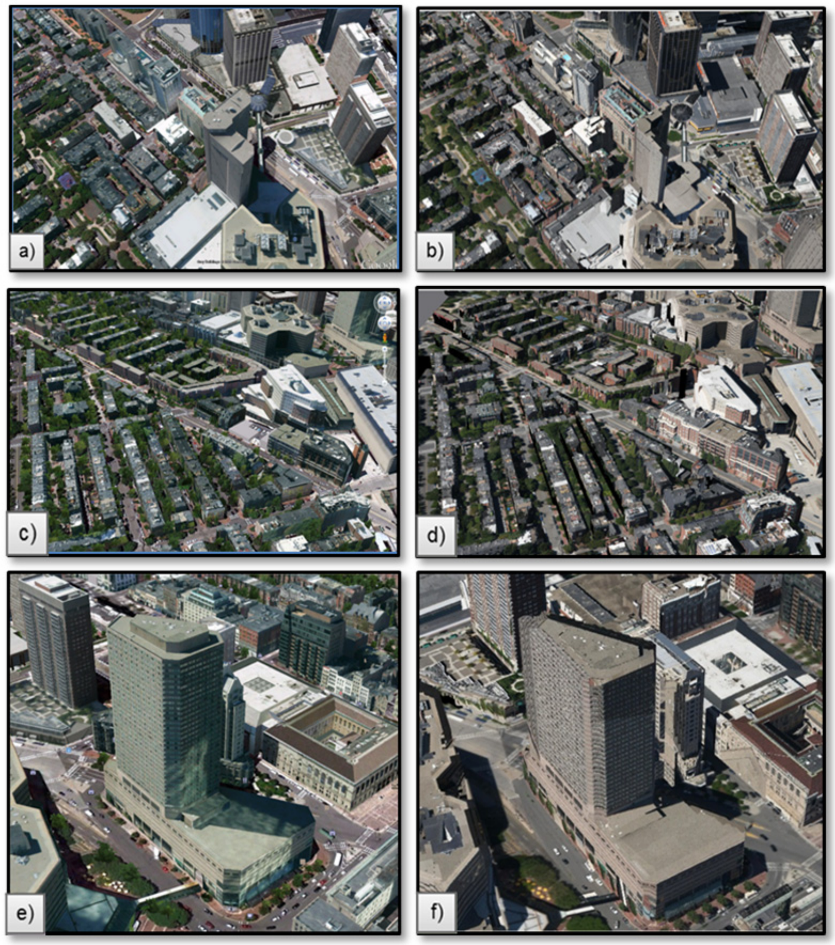


Figure 18. Google Earth Comparisons. We show several comparisons between Google Earth snapshots (a,c,e) and our result (b,d,f). Our method yields similar quality results in most cases and thus opens up the door for the rapid creation of city-scale 3D models.

planes that best fit the distribution of points instead of applying our current heuristic. Third, we have observed additional information is present in the luminance channel of the images; in particular, sharp building edges may appear distinctly. We plan to exploit those edges to improve the proxy model. Fourth, our clustering method finds the macro-structure of a building. However, our reconstruction process captures additional structural detail (e.g., roof in Figure 7e), as could a secondary wide-baseline stereo method using our proxies. Fifth, additional street-level imagery could be used to improve the facade reconstruction. We intend to incorporate these tools to further refine building shapes.

REFERENCES

[1] Agarwal S., Snavely S., Simon I., Seitz S., Szeliski R., 2009. Building Rome in a Day, IEEE ICCV, 72-79.

[2] Allene C., Pons J.P., Keriven R., 2008. Seamless Image-Based Texture Atlases using Multi-band Blending, ICPR, 1-4.

[3] Bokeloh M., Wand M., Seidel H.-P.; 2010. A connection between partial symmetry and inverse procedural modeling, ACM Trans. on Graphics, 29(4).

[4] Boykov Y, Kolmogorov V., 2003. Computing geodesics and minimal surfaces via graph cuts, IEEE ICCV, 26-33.

[5] Ding M., Lyngbaek K., Zakhor A., 2008. Automatic registration of aerial imagery with untextured 3D LiDAR models. IEEE CVPR, 1-8.

[6] Frahm J.M., Georgel P., Gallup D., Johnson T., Raguram R., Wu C., et al., 2010. Building Rome on a Cloudless Day, ECCV, 368-381.

[7] Frueh C., Sammon R., Zakhor a., 2004. Automated texture mapping of 3D city models with oblique aerial imagery, 3DPVT, 396-403.

[8] Gallup D., Pollefeys M., Frahm J.M., 2010. 3D Reconstruction Using an n-Layer Heightmap, DAGM, 1-10.

[9] Gao Z., Nocera L., Neumann U., 2012. Fusing Oblique Imagery with Augmented Aerial LiDAR, ACM SIGSPATIAL, 426-429.

[10] Goesele M., Snavely N., Curless B., Hoppe H., Seitz S., 2007. Multi-View Stereo for Community Photo Collections, IEEE ICCV, 1-8.

[11] Golovinski A., Kim V. G., Funkhouser T., 2009. Shape-based Recognition of 3D Point Clouds in Urban Environments, IEEE ICCV, 2154-2161.

[12] Grzeszczuk R., Kosecka J., Vedantham R., Hile H., 2009. Creating Compact Architectural Models by Georegistering Image Collections, 3DIM, 1718-1725.

[13] Kutulakos K.N., Seitz S.M., 2000. A Theory of Shape by Space Carving, IJCV, 38(3), 199-218.

[14] Kwatra V., Schodl A., Essa I., Turk G., Bobick A., 2003. Graphcut Textures: Image and Video Synthesis Using Graph Cuts, ACM Trans. on Graphics, 22(3), 277-286.

[15] Lafarge F., Keriven R., Bredif M., Vu H.H., 2010. Hybrid multi-view Reconstruction by Jump-Diffusion, IEEE CVPR, 350-357.

[16] Lafarge F., Mallet C., 2011. Building large urban environments from unstructured point data, IEEE ICCV, 1068-1075.

[17] Lazebnik S., Furukawa Y., Ponce J., 2007. Projective Visual Hulls, IJCV, 74(2), 137-165.

[18] Lefebvre S., Hornus S., Lasram A., 2010. By-example Synthesis of Architectural Textures, ACM Trans. on Graphics, 29(4).

[19] Lempitsky V., Ivanov D., 2007. Seamless Mosaicing of Image-Based Texture Maps, IEEE CVPR, 1-6.

[20] Liao H.H., Lin Y., Medioni G., 2011. Aerial 3D Reconstruction with Line-Constrained Dynamic Programming, IEEE ICCV, 1855-1862.

[21] Mathias, M., Martinovic A., Weissenberg J., Gool L.V., 2011. Procedural 3D Building Reconstruction using Shape Grammars and Detectors, 3DIMPVT, 304-311.

[22] Matusik W., Buehler C., Raskar R., Gortler S., McMillan L., 2000. Image-based Visual Hulls, ACM SIGGRAPH, 369-374.

[23] Montenegro A., Carvalho P., Gattass M., Velho L., 2004. Adaptive Space Carving, 3DPVT, 199-206.

[24] Müller, P., Zeng, G., Wonka, P., and Van Gool, L.; 2007. Image-based procedural modeling of facades, ACM Trans. on Graphics, 26(3), 85.

[25] Musiaski P., Wonka P., Aliaga D., Wimmer M., Gool L., Purgathofer W., 2012. A Survey of Urban Reconstruction, Eurographics STARs, 28 pages.

[26] Nan L., Sharf A., Zhang H., Cohen-Or D., Chen B., 2010. Smart-Boxes for Interactive Urban Reconstruction, ACM Trans. on Graphics, 29(4), 93.

[27] Park, J. P., Lee, K. H., Lee, J. 2011. Finding Syntactic Structures from Human Motion Data, Comp. Graphics Forum, 30(8), 2183-2193.

[28] Pierre S., Martino P., Georgios O., 1994. Mathematical Morpholog and Its Applications to Image and Signal Processing, ISMM, 484p.

[29] Pollard T., Mundy J.L., 2007. Change Detection in a 3D World, IEEE CVPR, 1-6.

[30] Poullis C., You S., 2009. Photorealistic Large-scale Urban City Model Reconstruction, IEEE TVCG, 654-669.

[31] Schmidt F.R., Toppe E., Cremers D., 2009. Efficient Planar Graph Cuts with Applications in Computer Vision, IEEE CVPR, 351-356.

[32] Shalom S., Shamir A., Zhang H., Cohen-Or D., 2010. Cone carving for surface reconstruction. ACM Trans. on Graphics, 29(6), 150.

[33] Shen C., Huang S., Fu H., Hu S., Zhou Q., 2011, Adaptive partitioning of urban facades. ACM SIGGRAPH, 184-194.

[34] Sinha S.N., Pollefeys M., 2005. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation, IEEE ICCV, 349–356.

[35] Stava O., Benes B., Mech R., Aliaga D., and Kristof P., 2010. Inverse procedural modeling by automatic generation of L-systems, Comp. Graphics Forum, 29(2), 665-674.

[36] Taillander, F., 2005. Automatic building reconstruction from cadastral maps and aerial images, CMRT, 105-110.

[37] Teboul O., Kokkinos I., Simon L., Koutsourakis P., Paragios N., 2011. Shape Grammar Parsing via Reinforcement Learning, IEEE CVPR, 2273-2280.

[38] Toshev A., Mordohai P., Taskar B., 2010. Detecting and parsing architecture at city scale from range data, IEEE CVPR, 398-405.

[39] Tran S., Davis L., 2006. 3D surface reconstruction using graph cuts with surface constraints, ECCV, 219-231.

[40] Vanegas C., Aliaga D., and Benes B., 2010. Building reconstruction using Manhattan-world Grammars, IEEE CVPR, 358-365.

[41] Vogiatzis G., Torr P., Cipolla R., 2005. Multi-view Stereo via Volumetric Graph-cuts, IEEE CVPR, 391-398.

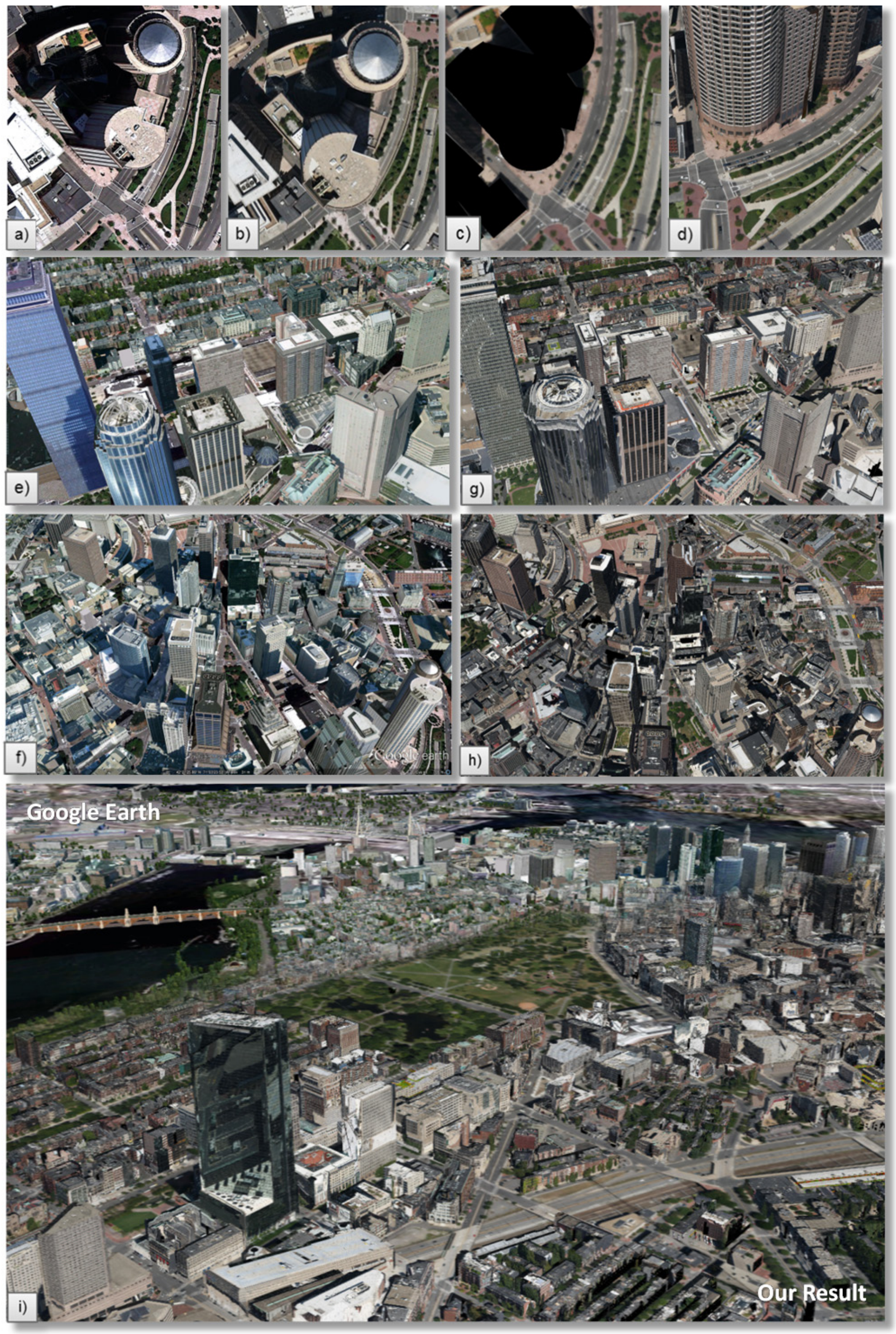
[42] Wang L., Neumann U., 2009. A Robust Approach for Automatic Registration of Aerial Images with Untextured Aerial LiDAR Data, IEEE CVPR, 2623-2630.

[43] Xiao J., Fang T., Zhao P., Lhuillier M., Quan L., 2009. Image-based Street-side City Modeling, ACM Trans. on Graphics, 28(5), 114-126.

[44] Zheng Q., Sharf A., Wan G., Li Y., Mitra N., Cohen-Or D., Chen B., 2010. Non-local Scan Consolidation for 3D Urban Scenes, ACM Trans. on Graphics, 29(4), 94-103.

[45] Zhou Q., Neumann U., 2010. 2.5D Dual Contouring: A Robust Approach to Creating Building Models from Aerial LiDAR Point Clouds, ECCV, 115-128.

Automatic Urban Modeling using Volumetric Reconstruction with Surface Graph Cuts: Supplementary Images



Supplementary Figure: Ground Surface: a) Google Earth image, b) our similar input image, c) our ground image after graph cut (building shown in black) – notice road is un-occluded, and d) 3D view of our result. Comparisons: e-f) Google Earth views in Boston and g-h) similar views using our results. i) Another view of the city where the upper left image triangle is from Google Earth and the bottom right image triangle is our result – a smooth blended transition is done along the diagonal (going from bottom left to upper right).