

# Automatic Alignment and Multi-View Segmentation of Street View Data using 3D Shape Priors

Timo Pylvänäinen  
Nokia Research Center

timo.pylvanainen@nokia.com

Joonas Itäranta  
Nokia Research Center

joonas.itaranta@nokia.com

Kimmo Roimela  
Nokia Research Center

kimmo.roimela@nokia.com

Ruisheng Wang  
NAVTEQ

ruisheng.wang@chi.navteq.com

Ramakrishna Vedantham  
Nokia Research Center

ramakrishna.vedantham@nokia.com

Radek Grzeszczuk  
Nokia Research Center

radek.grzeszczuk@nokia.com

## Abstract

We propose an automatic method for fast and accurate alignment and multi-view segmentation of city-scale street view data. We use simple 3D building outlines to bootstrap and automate all three steps of the algorithm: alignment of the LIDAR data, image segmentation using graph cuts and its multi-view refinement. The method can process city-scale datasets in approximately a day on a single server.

## 1. Introduction

Mobile Augmented Reality (MAR) has emerged recently as a promising technology for extending the Web to the physical world. Key AR enablers, such as visual tracking [19] and recognition [18], are now supported on handheld devices. Yet a broader adoption of MAR is limited by the lack of high-quality content—an accurately aligned 3D geometry of urban centers with links to information that can be overlaid on a user’s camera viewfinder.

We are working with street view data since that guarantees broad and uniform coverage needed to provide reliable location-based services. We focus first on generating building masks accurately aligned to the building contours in the panoramic images, as seen in Fig. 1(D). These are currently used in a prototype MAR system to highlight parts of the image containing links to information.

Generation of accurate masks faces two challenges. One, the street view data and the building outlines are often misaligned as in Fig. 1(A) where the building outlines are projected into an unaligned panoramic image. We correct this error by aligning the LIDAR point clouds to the building outlines as seen in Fig. 1(B) and then applying the same transformation to the camera poses as seen in Fig. 1(C). Two, the simple building outlines often cannot model accurately the complex architecture of some buildings. This is noticeable for the building on the right in Fig. 1(C). We

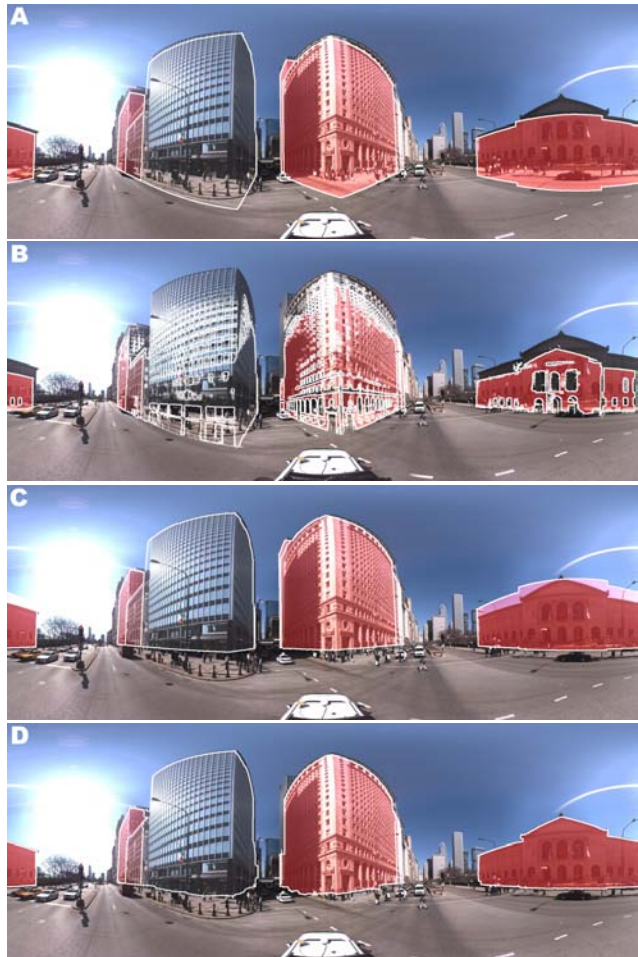


Figure 1. A panoramic image is overlaid with (A) unaligned building outlines, (B) LIDAR point clouds aligned to building outlines, (C) aligned building outlines, (D) result of our multi-view segmentation algorithm.

refine the masks obtained from projecting the building outlines by applying multi-view segmentation of the panoramas. Both stages of the algorithm use simple building outlines as 3D shape priors to bootstrap the problem.

The method is fully automatic and fast. We can process city-scale datasets in a day on a single server. The efficiency is achieved in part by implementing the computationally costly steps of the method in graphics hardware.

### 1.1. Related Work

Similar to Lothe *et al.* [12], we correct drift in large-scale reconstructions by aligning them to a coarse 3D model of the environment using non-rigid Iterative Closest Point (ICP) algorithm [4, 16], except we work with the LIDAR data while they use point clouds obtained using structure from motion.

A related problem of aligning image data to point cloud data is addressed in [22] as well as in [17]. In our case, the LIDAR data is collected simultaneously with the image data both mounted on the same fixed rig so very little registration error exists between data collected simultaneously. Methods such as these could, however, be used to improve the internal consistency of the data by aligning images to LIDAR data collected on a different pass of the same location.

Image segmentation is done with graph cuts [3, 1] using an efficient implementation by Boykov and Kolmogorov [2]. There has been much work done on segmentation using shape priors. The most pertinent references include [11, 6, 10]. Our approach differs from this earlier work in that we obtain the shape priors automatically by projecting the 3D building outlines into the panoramic images. The probability distribution for pixel intensities belonging to the objects is also initialized automatically from the segmented point clouds.

Our approach is related to bi-layer segmentation of stereo [8] and multi-view segmentation [21, 15]. While this earlier work focuses on simultaneous segmentation of registered 2D images and 3D points, we make individual image segmentations consistent across views by re-projecting segmentations from the neighboring views using available 3D information and use voting to find and correct areas with conflicting results.

While currently our work focuses on obtaining an accurate and stable multi-view segmentation, in the future, we plan to use these results for robust 3D geometry reconstruction and image-based modeling of urban landscapes [7, 20, 13, 5].

### 1.2. Our Approach

This paper makes the following unique contributions:

- Fast and reliable ICP-based 3D alignment of street

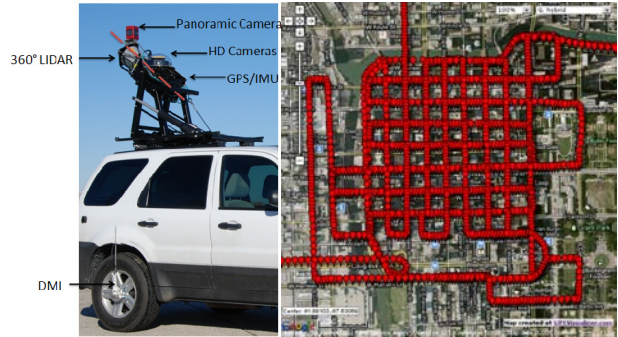


Figure 2. Figure shows NAVTEQ collection vehicle (left) and geographic area covered by street view data (right).

view data to the building outlines that corrects for the errors due to drift in sensor measurements and misalignment between the two sources of data. The alignment is used to segment from the LIDAR data the points that are likely to belong to the building façades.

- Fast and automatic graph-cuts-based segmentation of panoramic images that uses the building outlines and the segmented point clouds to initialize and constrain the algorithm. The inclusion of prior shape knowledge ensures robust performance with noisy and challenging data.
- 3D labeling of data that accumulates the segmentation evidence from multiple views and combines it with available 3D information to produce consistent and accurate results.

The rest of the paper is organized as follows. The preliminaries are presented in Sec. 2. Sec. 3 describes the alignment of street view data to the building outlines. Sec. 4 formulates our approach to image segmentation using graph cuts. Our approach to multi-view segmentation is presented in Sec. 5. The results are presented in Sec. 6 and the conclusion in Sec. 7.

## 2. Preliminaries

Data is collected by NAVTEQ using a data collection vehicle as shown in Fig. 2 (left). This mobile mapping system is composed of a 360 degree LIDAR sensor (Velodyne HDL-64E), panoramic camera (Ladybug 3), high definition cameras (Prosilica), GPS, Inertial Measurement Unit (IMU) and Distance Measurement Instrument (DMI). The Velodyne LIDAR sensor consists of 64 lasers mounted on upper and lower blocks of 32 lasers each and the entire unit spins. This design allows for 64 separate lasers to each fire thousands of times per second, generating over one million points per second. The Ladybug 3 covers more than 80 percent of a full sphere, with six high quality 1600x1200 Sony

CCD sensors, and provides up to 12 MP images at 15 Frames Per Second (FPS). We select 8MP output since the gain from having more pixels is insignificant. All of these sensors are geo-referenced through a Global Positioning System (GPS) and Inertial Measurement Unit (IMU).

We are working with a dataset collected for downtown Chicago that contains approximately 1000 building outlines, 21000 panoramic images captured at 4-meter intervals, and 4 billion LIDAR points covering a distance of 8 km and the geographic area shown in Fig. 2 (right). We also have a dataset for San Francisco with similar characteristics from which we use portions corresponding to specific landmarks.

The building outline database comes from Sanborne Inc. It consists of 2D polygons that represent the building footprint as seen from an aerial view. For each 2D polygon, base and roof elevation are also provided. If a building has complex geometry, it is very coarsely approximated as a set of sub-buildings that have separate elevation data for each.

### 3. Data Alignment

The depth measurements collected by the LIDAR are initially merged in a world coordinate system using sensor pose data from the IMU. The IMU uses GPS and inertial dead reckoning to keep track of the LIDAR sensor pose in the world. This process is susceptible to drift and GPS measurement errors that need to be corrected. The panoramic image collection system is physically fixed to the LIDAR and will have exactly the same error.

The GPS error and IMU drift depend on a number of external factors, which vary in time and place. The LIDAR scan therefore suffers from non-rigid deformations. Locally, however, the error can be approximated by a similarity transformation. We therefore split the scan data into manageable segments, estimate the similarity transformation at the segment boundaries and interpolate to make the corrections continuous and smooth.

We choose to split the scan at points where the vehicle has returned to a place it has already scanned earlier. Typically, in a city this will happen at every street block corner. There are two advantages to choosing these points: the misalignment from the scans at the same location from different times is most noticeable at these points, and secondly, street corners typically provide clear building corners (both in the building outline data and in the LIDAR scan) ideal for estimating similarity transformation using the iterative closest point (ICP) algorithm.

At each intersection, we consider a point cloud spanning about three blocks. The data used to estimate the transformations therefore overlap ensuring continuity in the alignment.

### 3.1. Preprocessing

The ICP algorithm iterates between assigning points to the closest planes and estimating the similarity transformation that minimizes the distance between the points and the planes they are assigned to. To obtain good results, it is critical to remove outliers. Since the building models in the GIS database are fairly crude, with usually just one rectangular polygon per building wall, only planar structures can be aligned. So we consider only the dominant plane of each wall in alignment, and prune all the other points in preprocessing.

To find the dominant planes, we first threshold all points that are too far from any wall in the building model. A very large threshold can be used, we chose 20 meters. This will get rid of some large planar structures, such as the ground plane, which might otherwise be problematic in the following step. The remaining points are next clustered based on which wall they are closest to. Finally, Hill Climbing RANSAC [14] is used to fit a plane to each cluster of points. For each cluster, only points within 30cm threshold of the estimated final plane are retained.

The building models only contain vertical façades and the LIDAR scans do not reliably capture the extent of the walls. For these reasons, it is tricky to reliably estimate the altitude correction using the ICP alone. We assume that the building footprints are at the street level and we know how high from the ground the sensor is mounted on the vehicle. For initial altitude correction, we simply find the closest building wall to each IMU location and use the difference between the base and IMU altitude as the correction. The altitude corrections for all IMU locations are averaged for final result. The altitude is later refined, as explained in the next section.

### 3.2. Estimating Similarity Transformation

After preprocessing, the ICP alternates between two steps. First, each point is simply assigned to the closest wall after applying the current estimate of the similarity transformation. Then a new similarity transformation is estimated which minimizes the cost function:

$$E = \sum_{i=1}^N (n_{p(i)}^T (s \cdot \text{rot}_z(\mathbf{v}_i, \alpha) + \mathbf{t}) - d_{p(i)})^2, \quad (1)$$

where  $p(i)$  gives the index of the wall to which point  $i$  is currently assigned,  $n_k$  the normal of the wall  $k$ , and  $d_k$  the distance of the wall from the origin. The terms of the sum are simply squared distances from the walls.  $\text{rot}_z(\mathbf{v}, \alpha)$  denotes rotation of the point  $\mathbf{v}$  about the  $z$ -axis (i.e. gravity) by  $\alpha$  radians.

While it is easy to optimize for full rotation by using unit quaternions, we do not recommend this. There are often indentations in walls, for example many buildings have exits

which extrude slightly from the building. If these are not removed by the preprocessing, their best least squares fit is obtained by tilting the building slightly. At the same time, the vertical direction is the most reliable measurement provided by accelerometers in the IMU. We therefore only optimize the rotation around the gravity vector. The parameters to optimize then are  $\alpha$ ,  $s$ ,  $t$ , the rotation angle about the z-axis, scalar scale factor, and vector offset, respectively. The cost function is optimized using the Levenberg-Marquardt algorithm.

Altitude correction refinement is done before and after the main ICP loop using the ICP point to plane associations. A cost for each candidate altitude is given which consists of the number of points below their corresponding walls, and the number of points above their corresponding walls divided by ten. The reason for division by ten is that we found that it is more likely for the buildings to extend higher than the building models indicate, than for the base of the model to be too high. We do a brute force search for the lowest cost altitude correction around the initial altitude estimation.

### 3.3. Interpolation

The timestamps associated with the 3D points and the images are used to interpolate the similarity transformations. For each point in the LIDAR scan, its final position is given by a linear combination of the locations obtained from the individual transformations applied to it. In equations

$$\mathbf{x}' = \frac{\sum_{i=1}^K g(t_i - t_x) T_i(\mathbf{x})}{\sum_{i=1}^K g(t_i - t_x)} \quad (2)$$

where  $g$  is an exponential kernel with suitable width, we use  $e^{-\frac{t^2}{100}}$ , which works well for the drift rate in our data (units are in seconds). In practice transformations for which  $g(t_i - t_x)$  is negligible are not considered.

For the panoramic image locations, the same interpolation is used. For the panoramic image rotation, we simply replace  $\mathbf{x}$  with a rotation matrix. A linear combination of rotation matrices may not, of course, be orthonormal. The SVD is used to obtain the closest orthonormal matrix in the Frobenius norm.

We found that this interpolation gave better results than quaternion interpolation, but there is also a good justification. The transformations were estimated to essentially move points to their correct positions. Considering the axes of each rotation matrix as points along the principal axes in the camera coordinate system, each term in the linear combination “votes” where those points should go. The linear interpolation of these points works exactly the same way as for the location. The SVD in the end simply finds the best rotation matrix to do this.

## 4. Single-View Segmentation

### 4.1. Masks Generation

Once we have aligned the LIDAR data and the panoramic images to the building outlines, we project them into the panoramic images for use in the segmentation algorithm. For this purpose, we have implemented a hardware-accelerated renderer using OpenGL 2.0 and the GL Shading Language, allowing us to quickly project geometry into the panoramic image space and examine the results interactively.

Each panoramic image is registered in an azimuth-elevation coordinate system relative to the orientation of the scanning vehicle, whereas the aligned building outlines and LIDAR data have global UTM coordinates. In order to render these data accurately using single-precision arithmetic, we first center the building and point cloud data on the sensor location. The resulting vertex data can be sent to our renderer and rasterized into a cube map, which is then rotated and warped in a fragment shader to project the geometry into the coordinate system of the original panoramic image.

The initial set of buildings associated with each panoramic image is based on distance from the panorama origin. These building outlines are first rendered as a mask image with unique RGB colors. The resulting image is scanned to identify buildings that are actually visible in the panoramic image.

The point clouds associated with each visible building are then rendered into a second mask, using the same RGB coding as for the building outlines. The depth buffer is first initialized with slightly offset depth values from the building outlines, and the points depth-tested against the depth buffer. This generates a mask image containing the points corresponding to visible walls of buildings.

Finally, a depth image of the (non-offset) building outlines combined with the point clouds is generated. This image encodes 24-bit depth values as RGB colors, giving the world-space distance from the panorama sensor to the building at each pixel location.

### 4.2. Segmentation with Graph Cuts

For each label (building ID or background), the projected building masks indicate our prior understanding where the buildings (or background) should be. For each label, we dilate the projected outline prior to get a candidate area for that label. An infinite cost is set for assigning the building label to a point outside the dilated prior. For 2048x1024 pixel panoramic images, we dilate by 50 pixels.

The foreground masks used to seed the segmentation are generated by projecting the points corresponding to visible walls of buildings into the panoramas. Because of slight variation in depth which pushes the points behind the build-



Figure 3. Point cloud is first projected to the image, holes are filled and finally the result is eroded slightly to avoid false labeling background as buildings. Top half shows the original projection, and lower half the final result.

ing model wall, and because the LIDAR scan does not get a return laser from windows, the masks will have holes in them. These holes are removed by a simple vertical and horizontal filling, where any background pixel between two points with the same label along vertical or horizontal lines is filled. Due to misalignment, the point clouds do not always fall exactly inside the buildings, and for this reason the filled projection is eroded slightly. Fig. 3 illustrates this process.

Next we generate a binary mask of pixels which are part of the optimization problem. A point is included in this mask if all of the following are true:

1. According to the dilated label masks, there are multiple possible labels for the point
2. The point is not assigned a building label in the filled and eroded foreground masks.

This mask is then dilated by one pixel. After dilation, the boundary of the optimization area contains points which can have only one label. These labels are forced by assigning a zero cost for the label and infinite cost for any other label. We use the graph cut algorithm [3, 9, 2] to propagate these known labels to the rest of the data. The labels are initialized based on the projected building outlines. Fig. 4 illustrates the final setup for the graph cut.

The unary cost for the graph cut is computed using a simple feature. For each label, 20 sample features are extracted at random positions. For each label the sample positions are chosen such that they are indicated to belong to that label by the filled and eroded foreground masks. In the case of the background, which is never indicated by the foreground mask, we use points which cannot be labeled anything but background based on the dilated label candidate areas.

The 14 dimensional feature we use is based on a 20x20 pixel patch, and consists of the average color channel values, the variance in each of the three color channels, and

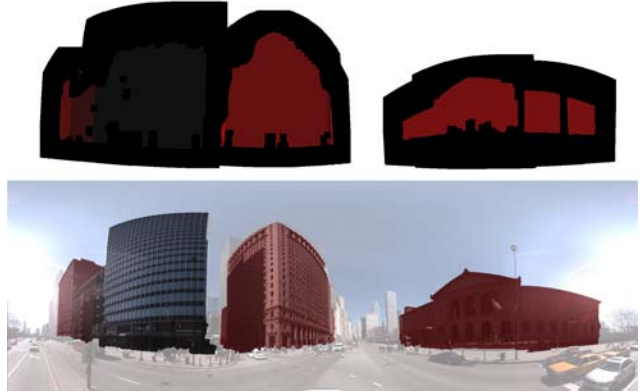


Figure 4. Top image shows a set up for the graph cut problem. Black pixels are subject to labeling by the graph cut. The background (white) and the buildings (colors) are fixed. Nodes are added to the graph only for the fixed pixels neighboring the black pixels. Bottom image shows the final segmentation result overlaid on the original panorama.

finally an 8 bin orientation histogram. The orientation histogram is accumulated proportional to the histogram magnitudes and the final feature vector is normalized to unit length. Because of this normalization, for very smooth areas the gradient histogram is negligible compared to the mean color, in which case the feature mostly represents the color of the patch. For high gradient areas, the feature mostly represents the dominant gradient orientations.

For each pixel part of the optimization, the feature is compared against the features for the possible labels. The unary cost for assigning the pixel a specific label is set to the shortest distance between the computed feature at that location, and the feature samples for that label.

## 5. Multi-View Segmentation Refinement

The segmentation is initially performed for each panoramic view independently. Due to challenging lighting conditions present in the panoramic images and complex, dynamically changing occlusions occurring when driving a vehicle in an urban environment, it is difficult to get consistent segmentations in every panoramic image. A much better result can be obtained by aggregating the segmentation results over multiple views.

To this end, for image  $i$ , we combine the segmentation results for images  $i - 5$  to  $i + 5$ . We use the building models to transfer the labels across the views. For each pixel in image  $i$ , we estimate its depth by tracing the corresponding ray in space. If the ray intersects with the building models, the depth is unambiguously resolved. If the ray does not intersect with any of the building models, we choose the wall closest to the ray and compute the depth by intersecting the ray with that wall plane.

Once the depth is resolved, the ray translates to a 3D

point which can be projected to images  $i - 5$  through  $i + 5$  based on the IMU camera pose measurements. Since the depth estimation is prone to errors, we generate a sequence of 3D points at multiple depths around the estimated depth and collect the labels from these multiple projections. For each projection, we check for possible occlusions using the depth masks generated using our hardware accelerated panorama viewer (as described in Sec. 4). If the projection is occluded, no information about the labels is collected.

The labels from the segmentation results projected into image  $i$  are accumulated into a probability distribution. Each transferred label gets a weighted vote. The weight depends on how far from the estimated depth the current projection depth is. Votes based on projection depth far from the estimated depth get a smaller weight. Similarly, the weight depends on how far the projection image is from the central image, i.e. labels transferred from image  $i - 1$  have a higher weight than labels from image  $i - 5$ .

After normalization, the voting gives us a discrete probability distribution, where the elements are the possible labels. This distribution fuses observations from multiple images integrated over multiple possible depth estimates. We default to the labeling indicated by the projected building models, and only change it if there is sufficient evidence for an alternate label.

## 6. Results

For the LIDAR data alignment stage, on the downtown Chicago dataset, the system detects a total of 530 locations where there are multiple panoramas from the same location. The majority of these results originate from road segments which were driven twice. These cases could be removed, but estimating this number of transformations is not a problem for our system. Fig. 5 shows the locations where transformations were estimated, as well as the resulting corrections to the camera location. If this data is compared to the map, it is clear how the tall buildings introduce GPS error. Fig. 7 shows statistics about the estimated GPS error.

Accurate altitude measurements are very difficult for GPS. The building footprints may not be absolutely correct either, but since the objective is to render buildings to panoramic images it is sufficient that the buildings are consistent across the dataset. The simple alignment we propose produces very pleasing results in this respect, as can be seen in Fig. 1. Fig. 6 shows the altitude corrections for the entire drive.

We have created a dataset for testing the segmentation algorithm consisting of a number of landmarks with interesting architecture. The dataset contains panoramic images, visible building outlines, and segmentation masks for the areas around the landmarks and is available for download <sup>1</sup>.

<sup>1</sup><http://www.anonymous.com>

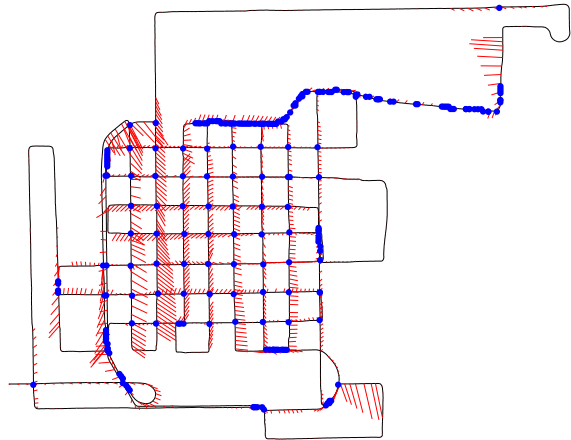


Figure 5. The alignment interpolated between intersections. The black line shows the original IMU track. The red lines indicate the correction vectors at fixed intervals, amplified 10 times. The blue dots indicate locations of the estimated transformations.

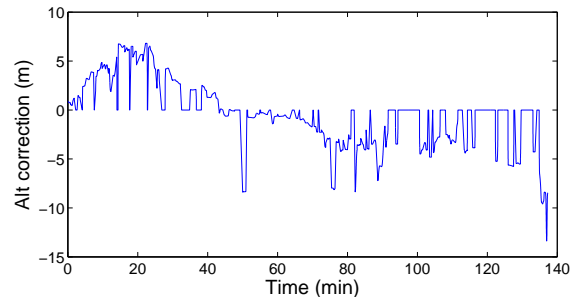


Figure 6. The figure shows the altitude corrections for the camera poses for the duration of one drive. The altitude correction is estimated only at intersections and interpolated elsewhere. The slight discontinuities are due to locations where we do not have suitable building models, in which case the altitude correction defaults to zero.

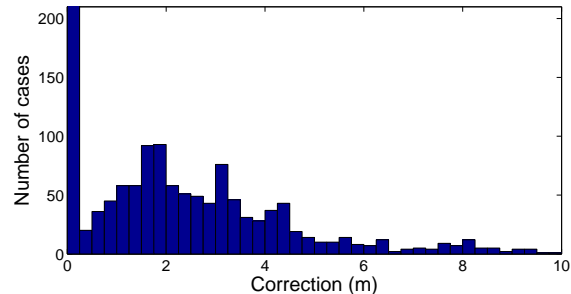


Figure 7. Histogram of the computed correction vector magnitudes. This is only the map correction indicated by the ICP, i.e. the altitude correction is not included. There is a peak at zero, but this is due to the cases where alignment defaulted to zero because no buildings were available. The results indicate that the GPS and the inertial-based tracking are already fairly accurate in the system.

We use 2MP ( $2048 \times 1024$ ) images for segmentation and upscale the masks to 8MP if necessary.

The initial segmentation already produces good results, but often occluding objects make the segmentation very challenging. For example, in Fig. 8(B) the street light in front of the Art Institute of Chicago (left) is labeled as the building. However, the multi-view segmentation, shown in Fig. 8(C), successfully removes the street light from the mask. Fig. 9 shows additional results of single-view segmentation improves the building contours and multi-view segmentation smoothes out the segmentation results across multiple views.

We ask the reviewers to peruse the supplemental material downloadable from a Program Chairs’ web site at: [http://homepages.inf.ed.ac.uk/rbf/3DPVT/DOWNLOADS/paper33\\_supp.zip](http://homepages.inf.ed.ac.uk/rbf/3DPVT/DOWNLOADS/paper33_supp.zip).

As can be seen from the video, the masks generated using multi-view segmentation are generally more accurate than the masks obtained by projecting the building outlines. At the same time, it can be seen from the videos that the multi-view segmentation still occasionally produces incorrect segmentations. We feel that this is the limit of how good the results can be with purely image-based segmentation and that further breakthroughs will come from performing simultaneous 3D and 2D segmentation [21, 15].

### 6.1. Computational complexity

Processing a single drive has five phases: (1) find intersections, (2) estimate transformations, (3) generate masks, (4) segment images and (5) fuse multi-view segmentations. All except for the first step are easily parallelizable. Except for step (2), the computation is done on a server with four 4-core Intel(R) Xeon(R) CPUs running at 2.5GHz. Step (2) is done on a single Nvidia(R) Quadro(R) FX 1700 GPU. The following table summarizes approximate computational times for steps (1)–(2) and (4)–(5) running on the downtown Chicago dataset.

| step | unit time | total CPU time | real time  |
|------|-----------|----------------|------------|
| 1.   | 15 min    | 15 min         | 15 m       |
| 2.   | 1.7 min   | 15 h           | 1 h        |
| 4.   | 1 min     | 350 h          | 22 h       |
| 5.   | 3 s       | 17 h           | 1 h        |
|      |           |                | 24h 15 min |

Unit times are per image for steps (4)–(5), per intersection for step (2) and for the entire drive for step (1). “Real time” is the time spent by the server to complete the step.

Since mask rendering takes almost no time, the runtime is determined by reading of the point clouds. Currently those are reloaded for each view separately. By caching them in memory we should be able to complete this step for the whole dataset in about 3 hours on a single GPU.

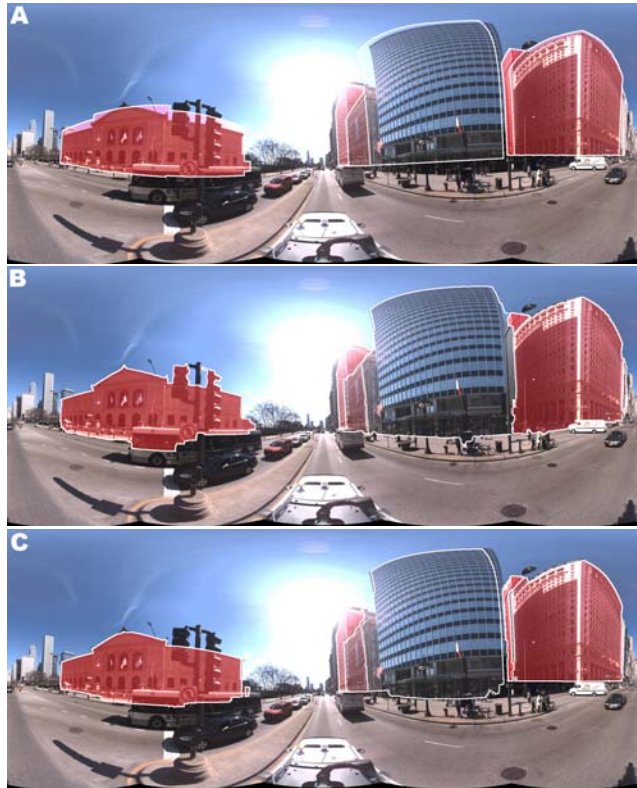


Figure 8. The Art Institute of Chicago. Masks projected into the panorama were generated from (A) aligned building models, (B) single-view segmentation and (C) multi-view segmentation.

## 7. Conclusion and Future Work

We have demonstrated a method for automatic alignment and multi-view segmentation of large-scale street view data. The method uses simple 3D building outlines obtained from an existing GIS database to initialize and constrain the solution.

We have shown that using graph cuts for image segmentation lets us precisely label pixels belonging to the buildings despite inaccuracies of the 3D models and misalignments in the data. The multi-view segmentation refinement corrects mistakes of the single-view segmentation due to difficult lighting conditions and challenging occlusions present in urban environments. We have shown that the proposed method is robust and efficient. We can process city-scale datasets in approximately a day on a single server.

In the future, we plan to perform joint 2D and 3D segmentation using building outlines as shape priors and introduce a set of simple yet flexible rules for refining building geometry based on common architectural constructs. The resulting 3D models would form the backbone of a full augmented reality system, where the buildings in a live camera view become active UI components.

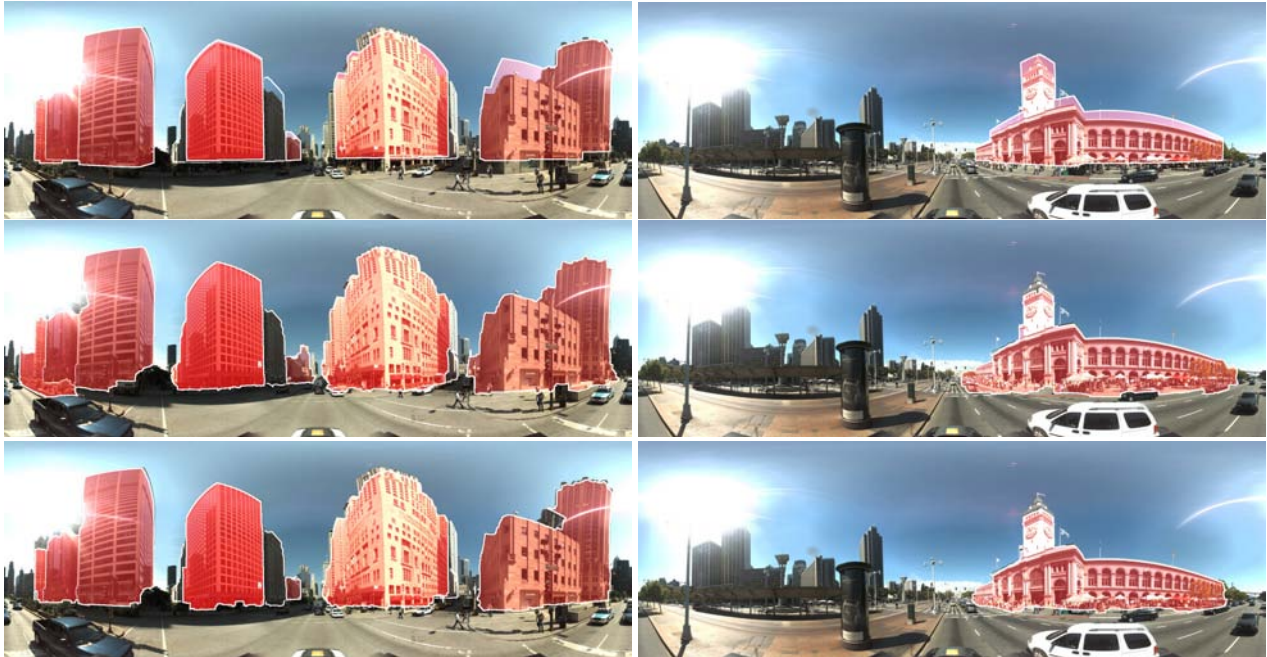


Figure 9. Segmentation results for the Chicago Tribune building (left) and the San Francisco Ferry Building (right). Top row: aligned building outlines projected to a panoramic image, middle row: single-view segmentation, bottom row: multi-view segmentation.

## References

- [1] Y. Boykov and G. Funka-Lea. Graph Cuts and Efficient N-D Image Segmentation. *Int. J. Comput. Vision*, 70(2), 2006. 2
- [2] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9), 2004. 2, 5
- [3] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), 2001. 2, 5
- [4] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. In *IEEE ICRA*, 1991. 2
- [5] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool. 3D Urban Scene Modeling Integrating Recognition and Reconstruction. *Int. J. Comput. Vision*, 78(2-3), 2008. 2
- [6] D. Freedman and T. Zhang. Interactive Graph Cut Based Segmentation with Shape Priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, 2005. 2
- [7] C. Frueh, S. Jain, and A. Zakhor. Data Processing Algorithms for Generating Textured 3D Building Facade Meshes from Laser Scans and Camera Images. *Int. J. Comput. Vision*, 61(2), 2005. 2
- [8] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-Layer Segmentation of Binocular Stereo Video. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, 2005. 2
- [9] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, 2002. 5
- [10] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image Segmentation with A Bounding Box Prior. *IEEE International Conference on Computer Vision*, 0, 2009. 2
- [11] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy Snapping. *ACM Trans. Graph.*, 23(3), 2004. 2
- [12] P. Lothe, S. Bourgeois, F. Dekeyser, E. Royer, and M. Dhome. Towards Geographical Referencing of Monocular SLAM Reconstruction Using 3D City Models: Application to Real-Time Accurate Vision-Based Localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09)*, volume 0, 2009. 2
- [13] M. Pollefeys, D. Nistér, et al. Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision*, 78, 2008. 2
- [14] T. Pylvänäinen and L. Fan. Hill Climbing Method for Random Sample Consensus Methods. In *International Symposium on Visual Computing*, 2007. 3
- [15] L. Quan, J. Wang, P. Tan, and L. Yuan. Image-Based Modeling by Joint Segmentation. *Int. J. Comput. Vision*, 75(1), 2007. 2, 7
- [16] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001. 2
- [17] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai. Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes. *International Journal of Computer Vision*, 78(2):237–260, 2008. 2
- [18] G. Takacs, V. Chandrasekhar, et al. Outdoors Augmented Reality on Mobile Phone using Loxel-Based Visual Feature Organization. In *MIR '08: Proceeding of the 1st ACM International Conference on Multimedia Information Retrieval*, 2008. 1
- [19] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose Tracking from Natural Features on Mobile Phones. In *ISMAR '08: Proc. of International Symposium on Mixed and Augmented Reality*, 2008. 1
- [20] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based Façade Modeling. *ACM Trans. Graph.*, 27(5), 2008. 2
- [21] J. Xiao, J. Wang, P. Tan, and L. Quan. Joint Affinity Propagation for Multiple View Segmentation. *IEEE International Conference on Computer Vision*, 0, 2007. 2, 7
- [22] W. Zhao, D. Nister, and S. Hsu. Alignment of Continuous Video onto 3D Point Clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1305–1318, 2005. 2