

# Combining Maps and Street Level Images for Building Height and Facade Estimation \*

Jiangye Yuan

Computational Sciences & Engineering Division  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831  
yuanj@ornl.gov

Anil M. Cheriyyadat

Computational Sciences & Engineering Division  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831  
cheriyadatam@ornl.gov

## ABSTRACT

We propose a method that integrates two widely available data sources, building footprints from 2D maps and street level images, to derive valuable information that is generally difficult to acquire – building heights and building facade masks in images. Building footprints are elevated in world coordinates and projected onto images. Building heights are estimated by scoring projected footprints based on their alignment with building features in images. Building footprints with estimated heights can be converted to simple 3D building models, which are projected back to images to identify buildings. In this procedure, accurate camera projections are critical. However, camera position errors inherited from external sensors commonly exist, which adversely affect results. We derive a solution to precisely locate cameras on maps using correspondence between image features and building footprints. Experiments on real-world datasets show the promise of our method.

## Keywords

data fusion; GIS map; building model

## 1. INTRODUCTION

The rapid development of data collection capabilities gives rise to multi-modal datasets. Fusing the available data leads to new solutions to problems that are challenging, even impossible, with single-modal datasets. In particular, integrating geographic information with image data has been

\*This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

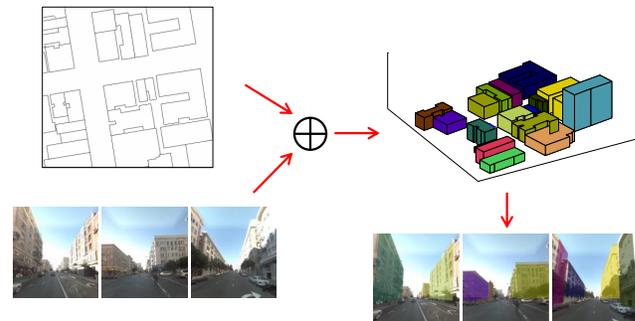
Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

*UrbanGIS 16, October 31-November 03 2016, Burlingame, CA, USA*

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 123-4567-24-567/08/06...\$15.00

DOI: 10.475/123\_4



**Figure 1: Building height estimation and building facade identification using building footprints and street level images.**

increasingly exploited [20, 17, 18], which significantly enhances the information extraction capability. In this paper, we utilize building footprint data from GIS maps and street level images, and develop a fully automatic method to estimate building heights and generate building facade masks in images. The concept is illustrated in Fig 1.

Building heights are important information for many applications related to urban modeling. Stereo pairs of high resolution aerial images and aerial LiDAR data are typically used to acquire height information [22, 15, 5]. Although highly accurate results can be obtained, those types of data are not easily accessible due to expensive collection process. We derive height information in a novel way. Through camera projections, we map edges of building footprints with different heights to images. The building height is determined by matching the projected edges with building rooflines in street view images. We design an indicator that incorporates color and texture information and reliably find the projected edges aligned with rooflines. This method does not require aerial sensing data. In addition, by extending building footprints vertically with estimated heights, we can generate 3D building models. Despite being significantly simplified, such building models are useful in applications requiring low storage and fast rendering.

Identifying buildings in images is an important yet challenging task for scene understanding. A main strategy is to learn a classifier from labeled data [19, 16]. Deep neural networks trained with massive labeled data have shown excellent performance on segmenting semantic objects in im-

ages, including buildings [9, 13]. In contrast, we generate building masks by projecting the 3D building models onto images. This approach provides an alternative solution that has the following advantages. First, it is capable of dealing with large appearance variations without complicated training and expensive labeled data collection. Second, each individual building in images is associated with a building on maps. Such a link allows attribute information from map data to be easily transferred to the images. For example, we can identify the buildings in an image corresponding to a specific restaurant if the information is given on maps. Note that while extracting builds from images has been addressed through a similar strategy of projecting georeferenced models [6, 17], those studies use existing building models instead of 2D maps.

Our approach involves matching building features between maps and images. However, even when camera projection parameters are available from sensors, cross alignment between maps and images still poses a major challenge. A main reason is that the measurement of camera positions mostly relies on Global Positioning Systems (GPS), which tend to be affected by measurement conditions, especially in high density urban areas. Despite various new techniques combined with additional sensory information [14, 8, 11], GPS errors remain at a noticeable level. For example, a median position error of 5-8.5 meters is reported for current generation smartphones [21]. To overcome inaccurate spatial alignments, we propose an effective approach based on correspondence between map and image features to refine camera positions on maps. The method is fully automated and works reliably on real-world data.

In this work, we use building footprint layers from OpenStreetMap (OSM)<sup>1</sup> – the largest crowd sourced maps produced by contributors using aerial imagery and low-tech field maps. The OSM platform has millions of contributors and is able to generate map data with high efficiency. We use Google Street View images, which capture street scenes with a world-wide coverage. Images can be downloaded through an publicly available API<sup>2</sup> with all camera parameters provided.

## 2. RELATED WORK

A number of previous studies have explored the idea of creating building models based on 2D maps, where a critical step is to obtain height information. In [10], building heights and roof shapes are set to a few fixed values based on known building usage. In more recent work, stereo pairs of high resolution aerial images and aerial LiDAR data are used to estimate height information [22, 15, 5]. However, since those types of data are not available for most areas in the world, the methods do not scale well. In this work, a new method is proposed to acquire building heights, which utilizes map data and street view images. Since input data are widely available, the method can scale up to very large areas.

In order to correctly project map data onto images, images need to be registered with maps. This is a difficult task because it requires matching between abstract shapes on a ground plane and images from very different views. This has been pursued in a few recent studies. In [4] omnidirec-



**Figure 2: Building height estimation.** (a) A map containing building footprints. The green marker represents the camera location. Blue edges indicate the part within the field of view. (b) A street level image. All blue lines are the projected edges with different elevation values.

tional images are used to extract building corner edges and plane normals of neighboring walls, which are then compared with structures on maps to identify camera positions. The method in [7] follows the same framework and aims to refine camera positions initially from GPS. It works on a single view image, but it involves manual segmentation of buildings in images to obtain highly accurate building corner lines. Instead of 2D maps, digital elevation models (DEM) has also been utilized, where building roof corners are extracted and matched with those in images [2]. However, DEM is much less accessible than maps. It should be noted that existing techniques for camera pose estimation given 2D-to-3D point correspondences (the  $PnP$  problem) are not applicable here because point correspondences are not available. We propose a method that registers a single view image with maps through a voting scheme. The method is fully automated and works reliably on real-world data.

The rest of the paper is organized as follows. Section 3 presents the method to estimate building heights and identify buildings in images. The method for estimating accurate camera position on maps is discussed in Section 4. In Section 5 we conduct experiments on large datasets and provide quantitative evaluation. We conclude in Section 6.

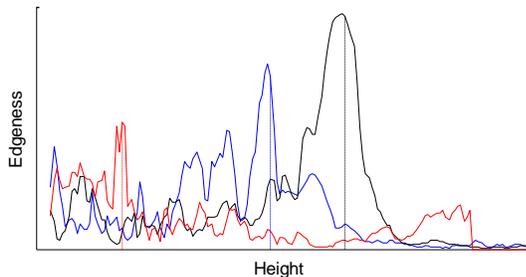
## 3. BUILDING HEIGHT AND FACADE ESTIMATION

For a building footprint in a georeferenced map, we have access to 2D geographic coordinates (e.g., latitude and longitude) of the building footprint. If we set the elevation of the building footprint to the ground elevation, we have 3D world coordinates. Then we project the edges of the building footprint onto the image through camera projections. The projected edges should outline the building extent at its bottom. As we increase the elevation of the building footprint, the projected edges move toward the building roof. When the projected edges are aligned with building rooflines, the corresponding elevation is the roof elevation of the building. The building height is simply the increased elevation. This procedure is illustrated in Fig. 2, where we project footprint edges within the field of view.

We need to determine whether projected footprint edges are aligned with building rooflines. A straightforward way is

<sup>1</sup><http://www.openstreetmap.org/>

<sup>2</sup><https://developers.google.com/maps/documentation/streetview/>



**Figure 3: Height estimation based on edgeness scores. Each color represents one building. Dashed lines indicate actual heights of buildings.**

to compute image gradients and select the projected edges with the maximum gradient magnitude as the roofline. However, because lighting conditions change significantly and there exist many straight edges other than rooflines, gradient magnitude often fails to represent the existence of rooflines.

Here we use an edgeness indicator that incorporates both color and texture information [12]. It is fast to compute and performs reliably. We first compute spectral histograms at a local window around each pixel location (the window size is set to  $17 \times 17$  in our experiments). A spectral histogram of an image window is a feature vector consisting of histograms of different filters responses. It has been shown that spectral histograms are powerful to characterize image appearances [12]. Here we use a spectral histogram concatenating histograms of RGB bands and two Laplacian of Gaussian (LoG) filter responses. Two LoG filters are applied to the grayscale image converted from RGB bands ( $\sigma$  is set to 0.5 and 1, respectively). The histogram of each band has 11 equally spaced bins. Local histograms can be computed efficiently using the integral histogram approach. Since histograms of RGB bands are sensitive to lighting conditions, we weight those histograms by 0.5. The edgeness indicator value at  $(x, y)$  is then defined as the sum of two feature distances between pixel locations  $\langle (x + h, y), (x - h, y) \rangle$  and  $\langle (x, y + h), (x, y - h) \rangle$ , where  $h$  is half of the window side length.  $\chi^2$  difference is used as the distance metric. This edgeness indicator reflects the appearance change between neighboring windows and thus better captures rooflines, which are essentially boundaries separating two distinctive regions. Fig. 3 shows a plot of edgeness scores versus heights for three buildings, where actual building heights are well captured by maximum edgeness.

The main steps of the proposed method are described as follows.

1. *Identify visible building edges on a map.* Based on camera parameters, we determine the field of view on the map. We select the edges of building footprints that are not occluded by other buildings. Buildings far away from the camera may not show clear rooflines in an image. Therefore, we only consider buildings within a distance of 60 meters.
2. *Project selected edges onto the image.* 2D geographic coordinates of the edges are known. We assign ground

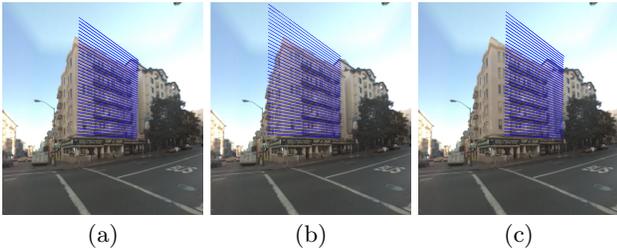
elevations to the edges and map them onto the image through a projective camera projection, which results in polylines in the image. Highly accurate ground elevations can be obtained from public geographic databases.

3. *Determine building heights.* For each visible building, we gradually increase the elevation of selected edges and project them onto the image. The projected edges scan the image through a series of polylines. When the sum of the edgeness scores on a polyline reaches the maximum, the corresponding elevation gives the building height. There are cases where taller buildings behind the target building are also visible, but they usually have rooflines with different length and shape, which do not give maximum edgeness values.
4. *Process a set of images.* The height of a building can be estimated sufficiently well with one image as long as building rooflines are visible in the image. If a collection of images are available, where a building appears in multiple images, we can utilize the redundancy to improve accuracy. For each building, we create a one-dimensional array with each element representing a height value and initialize all elements to zeros. After scanning an image for a building, we add one to the element corresponding to the estimated height value. Once all images containing the building are processed, we choose the element with the maximum value to obtain the building height. In some images where only lower part of a building can be seen, estimated heights are incorrect. We use a simple technique to deal with this issue. Because in such a case the projected edges are within building facades, the edgeness scores are generally small. We ignore the estimated height if the corresponding indicator value is smaller than a threshold, and thus incorrect estimates are not recorded in the array.
5. *Generate building facade masks in images.* Simple 3D buildings models can be generated by extruding boxes from building footprints with estimated heights. Buildings in the field of view are projected back onto images, which results in labeling facades of each building. To address the visibility problem (building models partially occluded by others), we use the painter's algorithm, where the farthest facade is projected first.

Our method treats building roofs as flat planes. For buildings with other roof shapes, the resulting height is generally between the top and the base of roofs. Since most non-flat roofs have a small height, our method can still give an estimate close to the mean height. The method may not work well when building rooflines in an image are completely occluded by other objects (e.g., trees and other buildings). However, with multiple views available, the method can exploit the rooflines of the same building visible in other images taken from different angles. If a building is completely obstructed on the map but visible in images because it is taller than obstructing buildings, the method will skip the building.

## 4. CAMERA LOCALIZATION ON MAPS

As discussed earlier, we need to register images with maps for accurate projections. In this paper, we focus on camera



**Figure 4: Impact of camera position errors.** (a) A building footprint edge projected onto an image with a correct camera position. A series of elevation values are used. (b) and (c) The projections with camera positions moved 3 meters away from the correct one.

positions, the measurement of which is much more noisy than orientation measurements. To illustrate the effect of camera position errors on projection results, Fig. 4 shows the building footprint edges projected onto images with shifted camera positions. As can be seen, there are clear misalignments between projected footprint edges and buildings in images even though camera positions are shifted by only 3 meters. Such misalignments may cause projected building footprints to fail to match the corresponding rooflines and hence incorrect height estimation. They also lead to incorrect building facade masks. In the following we propose a method that takes an initial position that may be noisy and estimate the accurate position on maps.

#### 4.1 Camera position from point correspondence

A 3D point  $\mathbf{P} = (X, Y, Z)^T$  in world coordinates can be projected onto a pixel location  $p = (u, v, 1)^T$  in an image plane through the camera projection equation:

$$\lambda \mathbf{p} = [\mathbf{K} | \mathbf{0}_3] \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} = \mathbf{KRP} - \mathbf{KRC}. \quad (1)$$

$\lambda$  is the scaling factor,  $\mathbf{K}$  the camera intrinsic matrix,  $\mathbf{R}$  the camera rotation matrix, and  $\mathbf{C}$  the location of the camera center in world coordinates. Note that  $-\mathbf{RC}$  equals to the camera translation. Given a pair of  $p$  and  $P$ , we have

$$\mathbf{C}' = \mathbf{P} + \lambda \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}. \quad (2)$$

That is, given correspondence between a pixel location and a 3D point in world coordinates, the possible camera positions  $\mathbf{C}'$  lie on a 3D line defined by (2).

In this work, we assume that camera position errors are mainly in a horizontal plane, because vertical camera positions can be reliably obtained with existing ground elevation data. Let  $\mathbf{C}' = (X_C, Y_C, Z_C)^T$ . We can discard Z dimension and simplify (2) to

$$\begin{bmatrix} X_C \\ Y_C \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} + \lambda \begin{bmatrix} \Delta X \\ \Delta Y \end{bmatrix}, \quad (3)$$

where  $\Delta X$  and  $\Delta Y$  are truncated from  $(\Delta X, \Delta Y, \Delta Z)^T = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}$ . This defines a line on a 2D plane. If correspondence of two pairs of points is given, the camera position can be uniquely determined, which is the intersection of two lines.

#### 4.2 Camera position from image and map correspondence

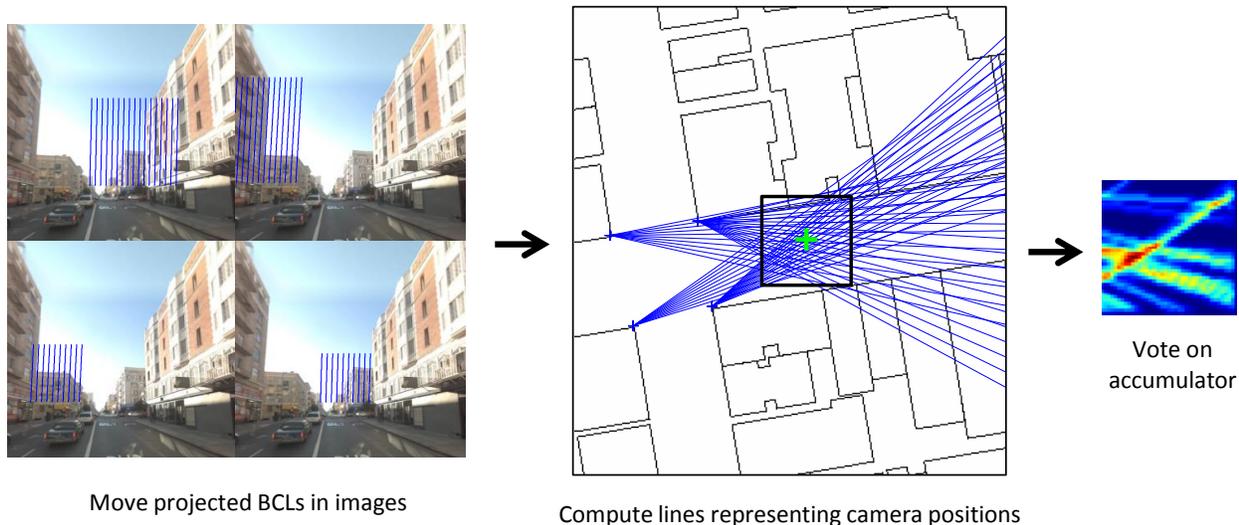
Based on the above analysis, we can determine camera positions based on point correspondence between images and maps. We use corners on building footprints, which can be easily identified from map data. However, it is difficult to find the corresponding points in an image. To provide better features for matching, we place a vertical line segment on each building footprint corner (two end points have the same 2D coordinate but different elevation values). We will refer to such a line as a building corner line (BCL). The line length is fixed. Note that at this stage building height is not available. When projecting a BCL onto an image with an accurate camera position, it should be well aligned with building edges in the image.

Under the assumption that position errors are within a horizontal plane, for a BCL projected onto an image with an inaccurate camera position, the displacement is along image columns. We first project a BCL onto an image using an initial camera position. Next, we horizontally move the projected BCL toward both directions within a certain range. The range is set to be inversely proportional to the distance from the BCL to the camera position. At each moving step, we have a pair of  $\mathbf{P}$ , the building footprint corner, and  $\mathbf{p}$ , the projected point moved along columns. By using (2), we can compute a line on the map that represents potential camera positions. We create an accumulator, which is a 2D array centered at the initial camera position. For each line, we increment the value of the bins the line passes through. The increased value is determined based on how well the moved BCL is aligned with building edges. At the end, the bin with the highest value gives the most likely camera position, which results in the best match between BCLs and building edges in images. The procedure is illustrated in Fig. 5.

When moving a projected BCL, we need to know whether it is aligned with a building edge so that proper values can be assigned to the accumulator. Detecting building edge is another challenging task. In an image of a street scene, contrast of building edges varies significantly due to changes of lighting conditions. Building edges can be fully or partially occluded by other objects. Moreover, there exist a large number of edges from other objects. Although the voting-based method is able to tolerate a reasonable amount of errors in detection results, a large number of false positives and false negatives can still confuse the method.

We exploit the idea of line support regions [3], which is widely used for straight line extraction. A line support region consists of pixels with similar gradient orientations, where a line segment can be extracted. A common practice for building edge extraction in previous work [4, 7] is to assume that building edges have relatively large contrast. However, we observe that in many cases there are only slight color changes around building edges. Line support regions are formed based on consistency of gradient orientations regardless of magnitudes and hence better suited in this task.

Since street level images are captured by a camera with its Y-axis orthogonal to the ground, building edges in images should be close to vertical lines. We select pixels with gradient orientations within  $22.5^\circ$  around the vertical direction and find connected regions with large vertical extents and small horizontal extents (in the experiments two thresholds are set to 50 and 20 pixels, respectively). When a projected



**Figure 5: Illustration of camera position estimation.** By moving projected BCLs (blue lines) in images, we compute potential camera positions (blue lines) on the map, which add values to an accumulator based on alignment measure between projected BCLs and building edges. The green marker indicates the initial camera position, and the black window shows the extend of the accumulator. In the accumulator, red pixels represent large values.

BCL hits a region center, the corresponding accumulator bins increment by one. A Gaussian density kernel is placed on each region center, and a projected BCL close to a center point casts a value equal to the maximum density value it reaches.

### 4.3 Integration with height estimation

Given an image and a map, the procedure includes the following two steps. The first is to apply the voting based method to determine the camera position, and the second is to estimate the building height by examining projected building footprints. In certain scenarios, a set of line support regions that do not correspond to building edges happen to form a strong peak, resulting in an incorrect camera location. To address this issue, we integrate height estimation with camera position estimation, which utilizes the information of building footprint edges in addition to corners to obtain accurate positions and simultaneously produces building heights.

From a resulting accumulator, we select the top  $k$  local maxima as candidate camera positions. For each of them, we project building footprint edges to match building rooflines in an image, as described in Section 3. We calculate the sum of edgeness scores for all projected building footprint edges that matches rooflines. We select the candidate with the largest sum, where rooflines in the image should be best aligned with building footprints. This strategy integrates two separate steps and reduces ambiguities in correspondence between BCLs and building edges. Algorithm 1 summarizes the algorithm for processing a street view image.

## 5. EXPERIMENTS

To evaluate the proposed method, we compile a dataset consisting of Google Street View images and OSM building

---

### Algorithm 1 Building height estimation with camera position refinement

---

- 1: Extract line support regions
  - 2: Compute an accumulator using the voting method. Select the locations of top  $k$  peaks  $pos(i), i = 1, \dots, k$
  - 3: **for**  $i:=1$  **to**  $k$  **do**
  - 4:   Set camera position to  $pos(i)$
  - 5:   Identify visible buildings on map
  - 6:   **for** each visible building **do**
  - 7:     Project building footprints onto image and estimate height based on edgeness values
  - 8:   **end for**
  - 9:   Compute  $S(i)$ : the sum of edgeness values of projected edges matching rooflines
  - 10: **end for**
  - 11: Output building heights with  $\max(S)$
- 

footprints. Images are collected by a camera mounted on a moving vehicle. The camera faces the front of the vehicle so that more building rooflines are visible. The image size is  $905 \times 640$ . The intrinsic and extrinsic camera parameters are provided, where extrinsic parameters are derived from GPS, wheel encoder, and inertial navigation sensor data [1]. We use 400 images covering an area in San Francisco, CA. Map data of the same area is downloaded from OSM, which are in the vector form. We convert the map data into an image plane with a spatial resolution of 0.3 meter. Geo-location information of map data and camera positions is converted into the UTM coordinate system for easy distance computation. Although camera positions in the dataset are more accurate than those solely based on GPS, we observe clear misalignments when projecting map data onto images.

We apply the proposed method to the dataset with the fol-



**Figure 7: Images causing incorrect height estimation. Blue lines represent detected rooflines.**

Error tolerance	2 m	3 m	4 m
Accuracy (refined positions)	72.2%	83.6%	90.1%
Accuracy (initial positions)	67.3%	78.6%	87.7%

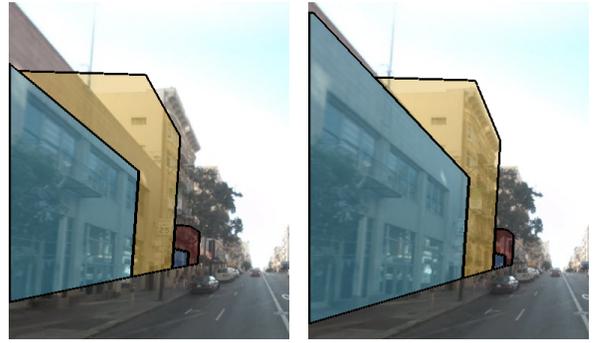
**Table 1: Accuracy of height estimation over different error tolerance.**

lowing parameter setting. For camera position estimation, we use an accumulator corresponding to a local window on the image plane converted from the map. The local window is centered at the initial camera position and of size  $40 \times 40$  pixels, which is to correct position errors up to 6 meters. When there are very few detected line support regions corresponding to building edges, estimated camera positions are not reliable. To address this issue, we set an accumulator threshold (set as 1.5) and use estimated camera positions only when the detected peak is above the threshold. We select the largest 5 peaks in an accumulator as candidates of camera positions. To scan an image with projected building footprints for height estimation, we use discrete elevation values from 3 to 100 meters with a step size of 0.2 meter, which cover the range of building heights in the dataset.

Fig. 6 shows a subset of resulting building models around a city block. We use building height information derived from LiDAR data as ground truth. Building models using ground truth heights are also displayed in Fig. 6. As we can see, they are very close to each other. There is one building where the estimated height is significantly different from ground truth, which is pointed by a red arrow. We examine the images that the building height is derived from and find that the error is caused by low image quality combined with unusual lighting conditions. Two images are shown in Fig. 7, where the proposed method confuses shadow borders as rooflines because the actual rooflines have a extremely low contrast.

To provide quantitative measurements, we calculate errors by comparing the height values from our method with ground truth. We set different error tolerance values (the maximum allowable deviation from ground truth) and compute the percentage of the buildings that have correct height estimation, which is reported in Table 1. The results agree with the observation in Fig. 6. We also evaluate the heights obtained without applying camera position estimation and show accuracy measurements in the table. As can be seen, refined camera positions lead to a clear improvement over initial ones.

With images aligned with maps, we label buildings by projecting visible parts of building models onto images. Fig. 8



**Figure 9: Building facade masks by projecting building models with raw camera position (left) and refined position (right).**

presents example results, where the facade masks obtained by projecting building models are well aligned with building facades in images. To assess the quality of segmented buildings, we select 100 images and manually generate building masks. Each building has a unique label corresponding to the footprint in the map. We calculate an accuracy rate as correctly labeled pixels divided by overall building pixels, where labels of a building that match other buildings are also penalized. Our results reach an accuracy rate of 85.3%. Camera position accuracy is particularly important for the facade mask quality. We find using raw positions the accuracy drops by 9.2%. Fig. 9 shows facade masks from projections using raw and refined camera positions. It can be seen that using the raw position projected buildings severely deviate from the corresponding buildings in images.

We implement the method using MATLAB on a 3.2-GHz Intel processor. The current version of the code takes on average 3 seconds to process one image, including camera position estimation and height estimation. The efficiency can be further enhanced by using parallel computing resources because each image is processed independently.

## 6. CONCLUSIONS

We have presented a new approach that fuses 2D maps and street level images, both of which are easily accessible, to perform challenging tasks including building height estimation and building facade identification in images. The method makes effective use of complementary information in data generated from two distinct platforms. Due to the wide availability of input data, the method is highly scalable. Our experiments show that the method performs reliably on real world data.

The proposed method to estimate camera position can work as an add-on for enhancing GPS accuracy. Although in this work we do not measure exact improvements on location accuracy because of the lack of ground truth, we indeed find that even one meter shift of an estimated camera position causes noticeable misalignments between projected building footprints and buildings in images. Worth mentioning is that this method is particularly suited for dense urban neighborhoods, which is often the most challenging situation for acquiring accurate GPS measurements.

We find that more accurate roofline detection and building edge detection can further improve results. In future work,

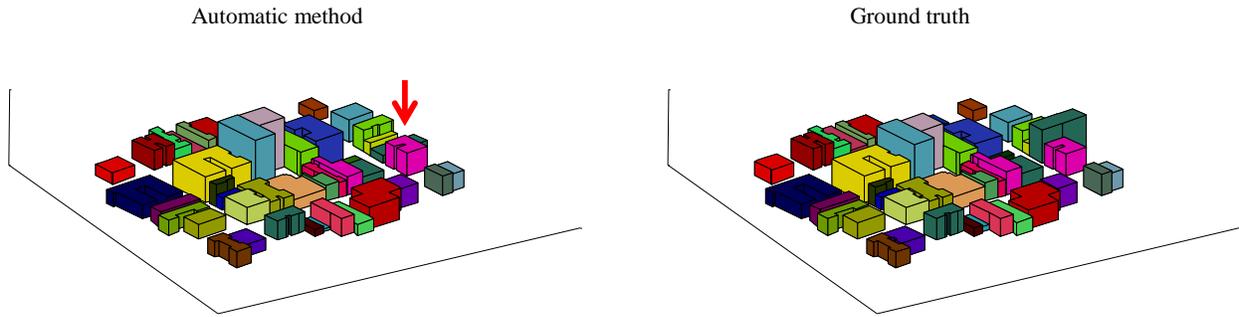


Figure 6: 3D building models. Left: building models generated by our method. Right: building models using LiDAR derived height information. The red arrow indicates one building with incorrect height estimation.



Figure 8: Example results of projecting building models onto images. Different buildings are represented by distinct colors.

we will investigate supervised learning based approaches, where a building boundary detector learned from labeled data is expected to provide more meaningful boundary maps.

## 7. REFERENCES

- [1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.
- [2] M. Bansal and K. Daniilidis. Geometric urban geo-localization. In *CVPR*, 2014.
- [3] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *TPAMI*, 8(4):425–455, 1986.
- [4] T.-J. Cham, A. Ciptadi, W.-C. Tan, M.-T. Pham, and L.-T. Chia. Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map. In *CVPR*, 2010.
- [5] L.-C. Chen, T.-A. Teo, C.-Y. Kuo, and J.-Y. Rau. Shaping polyhedral buildings by the fusion of vector maps and LiDAR point clouds. *Photogrammetric Engineering & Remote Sensing*, 74(9):1147–1157, 2008.
- [6] P. Cho and N. Snavely. 3D exploitation of 2D imagery. *Lincoln Laboratory Journal*, 20(1):105–137, 2013.
- [7] H. Chu, A. Gallagher, and T. Chen. GPS refinement and camera orientation estimation from a single image and a 2D map. In *CVPR workshops*, 2014.
- [8] N. M. Drawil and O. Basir. Intervehicle-communication-assisted localization. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):678–691, 2010.
- [9] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.
- [10] N. Haala and K.-H. Anders. Fusion of 2D-GIS and image data for 3D building reconstruction. *International Archives of Photogrammetry and Remote Sensing*, 31:285–290, 1996.
- [11] K. Jo, K. Chu, and M. Sunwoo. Interacting multiple model filter-based sensor fusion of GPS with in-vehicle sensors for real-time vehicle positioning. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):329–343, 2012.
- [12] X. Liu and D. Wang. A spectral histogram model for texton modeling and texture discrimination. *Vision Research*, 42(23):2617–2634, 2002.
- [13] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, pages 3431–3440, 2015.
- [14] H. Qi and J. B. Moore. Direct kalman filtering approach for GPS/INS integration. *IEEE Transactions on Aerospace and Electronic Systems*, 38(2):687–693, 2002.
- [15] F. Tack, G. Buyuksalih, and R. Goossens. 3D building reconstruction based on given ground plan information and surface models extracted from spaceborne imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:285–290, 2012.
- [16] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *ECCV*, pages 352–365. 2010.
- [17] C.-P. Wang, K. Wilson, and N. Snavely. Accurate georegistration of point clouds using geographic data. In *International Conference on 3D Vision-3DV*, pages 33–40, 2013.
- [18] S. Wang, S. Fidler, and R. Urtasun. Holistic 3D scene understanding from a single geo-tagged image. In *CVPR*, pages 3964–3972, 2015.
- [19] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM Transactions on Graphics*, 28(5):114, 2009.
- [20] J. Yuan and A. M. Cheriyyadat. Road segmentation in aerial images by exploiting road vector data. In *COM. Geo*, pages 16–23, 2013.
- [21] P. A. Zandbergen and S. J. Barbeau. Positional accuracy of assisted GPS data from high-sensitivity GPS-enabled mobile phones. *Journal of Navigation*, 64(03):381–399, 2011.
- [22] L. Zebedin, J. Bauer, K. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *ECCV*. 2008.