3D City Modeling from Street-Level Data for Augmented Reality Applications

Timo Pylvänäinen[1] Jérôme Berclaz[1] Thommen Korah[1] Varsha Hedau[1]
Mridul Aanjaneya[2] Radek Grzeszczuk[1]

[1]Nokia Research Center, Palo Alto, CA [2]Stanford University, Palo Alto, CA

Abstract

We present a method for automatically creating compact and accurate 3D city models needed for enhanced Augmented Reality applications. The input data are panorama images and LIDAR scans collected at street level and positioned using an IMU and a GPS. Our method corrects for the GPS error and the IMU drift to produce a globally consistent and well registered dataset for the whole city. We use structure from motion and skyline detection to complement the limited range of LIDAR data. Additionally, we propose a novel reconstruction technique that exploits architectural properties of urban environments to create an accurate 3D city model from incomplete data. Our method is able to process an entire city, or several terabytes of data, in a matter of days. We show that our reconstruction achieves higher accuracy than a commercial solution.

1. Introduction

The development of compelling outdoors Augmented Reality (AR) applications assumes the existence of a large-scale city model that is well-registered to the physical world. Abstract models with simplified geometries are desirable to create natural-looking augmentation of the physical world with virtual information content. Moreover, to ensure scalability, it should be possible to generate the model automatically from data that are easy to acquire.

In this paper, we introduce an approach for automatic generation of compact and accurate 3D city models from street-level LIDAR and panoramic imagery. This method can produce results such as the one shown on Fig. 1a. By intelligently fusing diverse sources of information, reasoning about the visibility of the scene, and taking into account the architectural properties of buildings, we are able to produce a 3D city model that is architecturally sound, looks correct where we have data available, and looks plausible where the data are missing. The resulting model is precisely registered to the physical world and the buildings are represented by simple shapes that compactly describe their geometry.

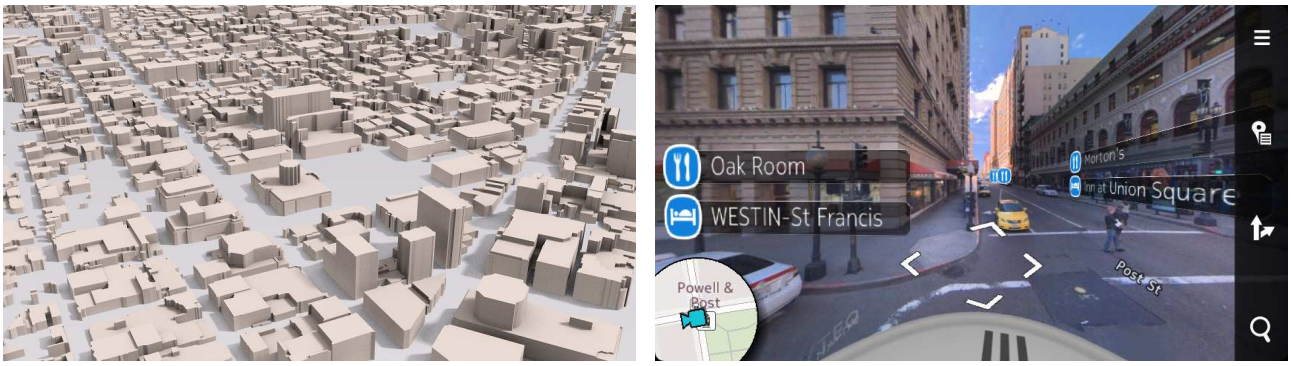
Fig. 1b illustrates how the 3D city model can be used to enhance a mobile augmented reality application. The visual realism of the scene is achieved by rendering an omnidirectional panorama of the location. The abstracted city model is used to create a natural-looking augmentation of the scene with virtual content. Using simple geometric primitives such as planes facilitates the alignment of the annotations with the building façades. The model can also serve as a geometric proxy for image-based rendering [14, 5] of the city from panoramic imagery.

1.1. Related Work

Large-scale 3D reconstruction of cities is an active area of research in which people have worked with a variety of data sources, including street-level imagery and LIDAR [6, 20, 3, 25], community photo collections [23, 13], and aerial imagery and LIDAR [24, 26]. We focus on using street-level data. Such data are easier and cheaper to acquire than aerial data, have higher resolution and precision at the street-level, and offer a more uniform city coverage than community photo collections.

Models with different levels of abstraction have been explored for city reconstruction, e.g. building façades as textured-mapped vertical planes [25], façades as ruled surfaces [3] and a dense height map model [10, 12]. While piecewise planar models result in visually pleasing reconstructions of individual façades, they cannot explain full 3D volume of the scene. On the other hand, dense models may capture the details of the surfaces but are unsuited for efficient rendering, storage, or transmission.

In this work, we focus on recovering simplified abstractions of geometry that can be used as a city-scale 3D canvas on which virtual objects can be painted. We recover full 3D models of the building geometry, while specifically handling problems due to missing data. Our method uses a similar height-map representation to [10], however, the reasoning we do on this representation is more global in nature. Instead of independently determining a height at every point on the grid, we find a segmentation yielding extruded footprint abstractions for different architectural blocks in the area. We do this by formulating an efficient multi-



(a) (b)

Figure 1: (a) A 3D city model automatically created using the proposed method. (b) A screenshot from Nokia City Scene, a mobile augmented reality application that uses the 3D city model to create a natural-looking augmentation of an urban scene.

label graph-cut problem that combines evidence from LIDAR and image data. Our graph-cut formulation is similar to the one of [8] and achieves similar space carving, however, instead of a binary label 3D grid we use multiple labels on a 2D grid, which makes our approach more scalable.

The Manhattan-world assumption has been extensively used by recent multi-view approaches. In [7, 8] and [11] it is used to extract planar structures for textureless indoor and outdoor surfaces, respectively, which are challenging for traditional multi view stereo. We employ Manhattan priors to guide our models where we have missing data. For instance, the LIDAR data often have holes, or lack full visibility of tall buildings due to limited range. Our graph-cut framework allows for propagation of height labels to estimate the heights in the missing parts of the data. Like [10], we orient our height map with dominant directions of the world. Similarly, we also assume that every observed data point is supported below by a solid column extending to the ground and encode this via the unary costs of our graph-cut framework.

Recovering 3D geometry models that align perfectly with the real world is challenging due to the limitations of outdoor positioning systems such as GPS and IMU. Visual odometry and loop closure techniques have been studied extensively in the literature [19, 16, 18, 4]. Coarse external reference models have been used to correct for drift in large scale problems [17, 21]. In our case, since no external model is available for alignment, we use portions of data that capture the same geographic location to register different instances of data with each other.

We propose a two stage registration framework. The city is first partitioned into blocks and local registration is performed to align the data within each block independently. The second stage performs an efficient global optimization that stitches the blocks together into a consistent solution.

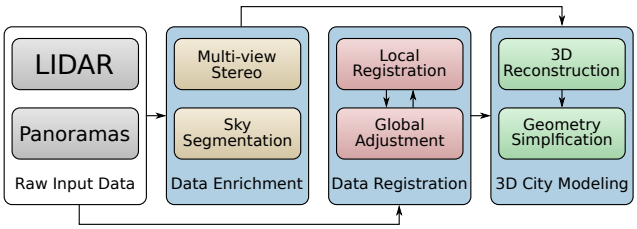


Figure 2: Phases of the city modeling pipeline. The incomplete LIDAR data are enriched through multi-view stereo and sky segmentation. Next, the raw input data are registered to a globally consistent positioning through iterative local registration and global optimization. Lastly, the 3D city model is created using a novel graph-cut based reconstruction and segmentation algorithm. The result is simplified and converted into a final 3D mesh.

2. 3D City Modeling Framework

The main components of our 3D modeling framework are highlighted in Fig. 2 and we outline them briefly here.

Raw Input Data consist of dense LIDAR point cloud and omnidirectional panoramic images acquired using a vehicle equipped with a high quality inertial measurement unit (IMU) and GPS system.

Data Enrichment deals with incomplete data resulting from the limited sensing range of LIDAR (∼100m horizontally and ∼20m vertically). We complement the LIDAR data with structure-from-motion (SfM) point clouds computed from panoramas[1]. In our system, we use the multi-view stereo implementation *PMVS* [9]. The extrinsic calibration from the IMU is accurate enough that no bundle adjustment step is necessary. While SfM clouds do not have the range limitations of LIDAR, their density does fall

[1]Note that for lower-cost acquisition systems, LIDAR sensor could be completely replaced with SfM data.

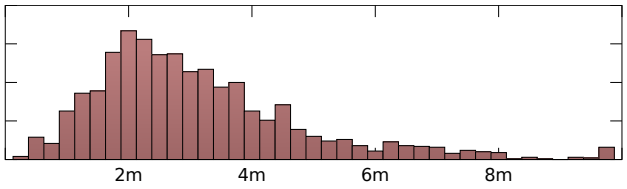


Figure 3: Histogram of positional corrections after registration.

off with height. To deal with very tall buildings and to be able to precisely delimit building roof lines, we developed a robust sky segmentation algorithm based on Grabcut [22]. The output is a sky mask for each panorama.

Data Registration ensures that the collected data are self-consistent and properly registered in 3D. To this end, we first partition the LIDAR data into sets of segments corresponding to city blocks. We then perform local registration of LIDAR segments per partition with ICP. Individual, locally-registered point clouds for each partition are then adjusted in a global optimization step that attempts to unify local results. Global analysis is also used to detect outliers and estimate a better initialization for the ICP in local registration, forming a feedback loop between local and global optimization.

3D City Modeling analyzes available data and exploits architectural properties of urban environments to deal with missing data. We use the facts that building walls have dominant directions, are vertical, and typically extend from the ground up. All this information is encoded into an efficient multi-label graph-cut formulation that produces an initial 3D city model that is further refined through a footprint simplification step.

3. Data Registration

Location estimation of the vehicle collecting the data has substantial inaccuracies that need to be corrected. In our experiments, the magnitude of the corrections averages to 3.1m and is distributed according to the histogram in Fig. 3.

Figure 4 shows the main steps of the registration pipeline. *City partitioning* partitions the problem into independent *local registration* tasks, the results of which are then merged into a consistent global solution during *partition consolidation*. *Spline interpolation* creates a smooth, continuous function from the local registration samples and uses the result to detect registration outliers. Final results are refined by re-initializing the outliers and re-running the registration steps.

3.1. City Partitioning

For scalability, we first partition the city-scale data into smaller units resembling city blocks, which can be processed independently. The data are collected by several vehicles that drive on different streets of the city, most likely visiting each location more than once. Each path taken by a vehicle is referred to as a *drive*, and a continuous time segment on a particular drive as a *drive segment*. We build a graph by binning the IMU position measurements of the vehicles across different drives on a regular grid. Minimum loops of this graph give us the desired partitions of the city, and can be easily obtained by constructing a doubly connected edge list and traversing its edges. We call a *partition* the set of drive segments corresponding to minimum loops, that are processed independently in the local registration step.

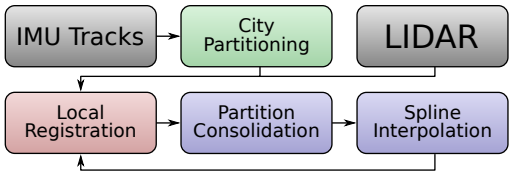


Figure 4: Flow chart of the registration pipeline. The preprocessing step shown in green partitions the problem into independent local registration tasks. The subsequent registration steps are run iteratively until convergence. The blue boxes constitute the global adjustment component in Fig. 2.

3.2. Local Registration

Each location of a city is typically visited more than once during different drives. We therefore use the overlapping drive segments to register the LIDAR data within each partition. Point clouds corresponding to each pair of drive segments are registered using the standard ICP algorithm. We estimate the point normals in a local neighborhood and minimize a squared sum of distances along the normals, allowing the planar structures to slide over each other. We found that estimating only translation was sufficient to register the input point clouds.

We maintain a set P of point clouds for which the relative translation is unknown. Initially, this is the set of all drive segments in a partition under consideration. We iteratively register the two point clouds in this set that have the largest overlap when projected along the vertical direction. When successful, the point clouds are merged reducing the size of P by one. If the registration fails, we move on to the next best candidate pair in P. This is continued until no further merge is possible. If the final result contains more than one set of registered drive segments, we treat them as belonging to separate partitions for the remaining steps of the pipeline.

Input point clouds to the registration do not necessarily represent the same structure. For instance, in tunnels or access ramps we do not want to merge segments that cross each other at different heights. To this end, we check that

the vertical span of points agree in areas where the registered sets share a significant number of observations. We also perform a global statistical analysis of the results to detect incorrect registrations.

3.3. Partition Consolidation

Drive segments are typically shared across multiple partitions; each drive segment usually belongs to at least two partitions – one on each side of the street. Since local registrations are done independently, a drive segment may have multiple offsets that align it to each of its parent partitions.

To unify the offsets for any drive segment that appears in multiple partitions, we formulate a global optimization problem where the partitions are moved with respect to each other to minimize the offset inconsistencies. Given N independently registered partitions, we estimate translations $\mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_N]^T$ that minimize the following function

$$E_m = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \sum_{k \in F_{ij}} \|(\mathbf{t}_i + \mathbf{o}_{ik}) - (\mathbf{t}_j + \mathbf{o}_{jk})\|, \quad (1)$$

where F_{ij} is the set of shared drive segments between partitions i and j, and $\mathbf{o}_{ik}$ is the offset computed for segment k during local registration of partition i. This can be formulated as a linear least squares problem $\mathbf{AT} = \mathbf{B}$ with a single global optimum. $\mathbf{A}$ is an $M \times N$ matrix where row k has 1 and -1 at columns i and j, respectively, and $\mathbf{B}$ is an $M \times 3$ matrix where row k contains $\mathbf{o}_{jk} - \mathbf{o}_{ik}$. M is the total number of pair-wise constraints from the local registration.

Care has to be taken to guarantee that the problem is well posed. It is easy to see that Eq. 1 is not changed if some global $\hat{\mathbf{t}}$ is added to all $\mathbf{t}_i$. To avoid this, we anchor an arbitrary partition p by introducing an extra row to A and B thereby fixing offset $\mathbf{t}_p = \mathbf{0}$. Once a solution is obtained, we normalize the result by applying a global $\hat{\mathbf{t}}$ such that $\sum_{i=1}^{N} \hat{\mathbf{t}} + \mathbf{t}_i = 0$. This results in a solution that is closest to the original GPS measurements and makes the final solution independent of the choice of the anchor partition.

3.4. Spline Interpolation

Registration yields discrete correction offsets along every drive. For practical usage, those offsets still need to be properly interpolated. We do this by fitting a cubic Hermite spline to the offsets applied to the drive segments. The offset corrections are interpolated using the distance traveled along the drive.

Even after consolidation, consecutive offsets might still conflict, as evidenced in Fig. 5a. The figure shows offset corrections obtained by the registration for different segments of a drive, plotted along the vertical axis. The offset for each drive segment is shown by a horizontal bar spanning across the spatial range of the segment and terminated with vertical lines. The lines represent the variance of offsets provided by partitions that share the segment. We detect outliers by fitting a spline to all but one registration sample, and verifying that the remaining sample is explained well by the interpolation. Finally, we fit a spline only to samples that were not flagged as outliers. The resulting curves are shown as solid lines in Fig. 5a.

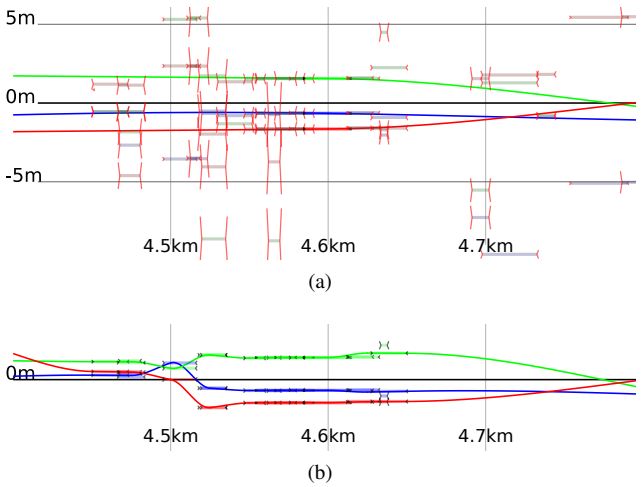


Figure 5: Cubic Hermite spline fitting is used to find the final correction function of the drive. The red, green and blue lines represent the altitude, east and north offsets, respectively. The continuous curves represent the spline. The horizontal bars are the offsets produced by point cloud registration and partition consolidation. (a) Before re-estimation the local registration results are still showing conflicting and noisy offsets. (b) After re-estimation the local registration results are in agreement and form a smooth curve. Note that the graphs in (a) and (b) have the same scale.

We then repeat the registration process for partitions containing outlier segments using the spline to initialize the offsets. The improved initialization allows us to reduce the ICP search radius and helps make correct point associations. The repeated local registration step either discards or refines the result. Fig. 5b illustrates the result after re-estimation. The majority of the drive segments are successfully registered after re-estimation, e.g. between 4.5km to 4.6km. Occasionally, a few samples fail to register and are discarded, e.g. near 4.7km.

4. 3D City Modeling

3D city modeling entails converting noisy and incomplete sensor data to a set of simple, abstract geometric shapes that approximate the true underlying buildings in a visually pleasing and meaningful way. All available input data are first converted into rays. Both LIDAR and SfM data are stored as rays with an origin and an end point. We convert each sky segmentation panoramic mask into a 360

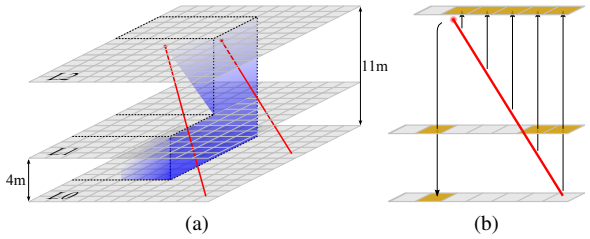


Figure 6: (a) A hypothetical point cloud in a blue shading. The cloud density decreases with height. The building shape determines a set of admissible building heights: $\mathbf{h} = [0m, 4m, 11m]$. (b) The costs accumulated by the the right ray shown in red in (a). Orange shading indicates increased costs.

degree fan of *sky rays* delineating the highest observed non-sky points in all directions. The sky rays extend to infinity.

We rely on architectural priors to help us deal with incomplete data, sensor noise and occluders such as trees and cars. Building walls extend along a few principal directions, are vertical, and typically extend from the ground up. Deviations from these assumptions are usually due to small structures that are undesirable in the abstract models we want to create. These architectural priors are implicitly encoded in our graph-cut problem formulation with a 4-connected neighborhood. The input data are rotated about the vertical axis to maximize the peaks in their east and north coordinate histograms. Since the majority of data points originate from building walls, this is a reliable way of guaranteeing that the walls are axis-aligned.

Graph-cut based optimization has been successfully used for different segmentation tasks to quickly find an approximate solution for the following minimization problem

$$E(\mathbf{h}) = \sum_{p \in V} d_p(h_p) + \sum_{(p,q) \in A} s_{pq}(h_p, h_q), \qquad (2)$$

where $\mathbf{h}$ is a labeling over a set of nodes V and A is a set of neighbor pairs [15, 1]. Together (V, A) form a graph. The term d_p can be interpreted as the prior likelihood of a given label and s_{pq} as a smoothness cost favoring connected components. We use this formulation to perform height based segmentation of the available data into a minimal set of building structures.

We define our problem on a 2D grid spanning east and north directions with cells of size 25 cm $\times$ 25 cm. Each cell adds a node to V and four edges to A corresponding to the north, east, south, and west neighbors of the cell. For each cell, the street level and the roof height are obtained by analyzing the lowest and the highest data points in that cell. For efficiency, we perform a mean-shift clustering on the building heights to obtain a small set of admissible values, $\mathbf{h}$. For each cell, there is a cost $d_p(h)$ associated with each possible height $h \in \mathbf{h}$ that is represented as a cost matrix.

We compute costs in such a way that labels corresponding to the height of building walls have low cost. Since LIDAR points have an associated origin, they can be represented as rays. The origins of SfM and sky points are taken to be the camera positions of the source panoramas. We assume that rays pass through empty space and stop on building structures. Assumptions based on architectural priors imply that a ray passing through cell p at some height e_p indicates that the height at cell p must be less than e_p. We therefore increase the costs $d_p(h)$ for all $h > e_p$. Similarly, we infer that a ray ending in cell p at height s_p indicates that the height at p should be at least s_p and we can therefore increase the costs $d_p(h)$ for all $h < s_p$. The process is illustrated in Fig. 6. We trace all LIDAR, SfM and sky rays to accumulate the cost matrices.

For each cell, we accumulate the point-height distribution function, which can be quickly queried to determine how many points are above a certain height in that cell. When tracing rays, each cell is checked to see if too many points are above, and if so, they are discarded. These rays typically originate from noisy LIDAR and SfM measurements or from a building arcade that violates our assumption of no overhanging structures.

To further improve the reconstruction, we use the IMU path to interpolate a terrain elevation map for the area. We form a *street mask* from the LIDAR points near the terrain elevation and then filter it to fill in small gaps. Because LIDAR often sees the street under the tree branches, this process is able to mask out trees from the reconstruction. For rays with end point on the street mask, the second step of accumulating costs below the end point is skipped.

We add a small bias cost to $d_p(h_i)$ that is monotonically decreasing with increasing height h_i. In the presence of no other evidence, higher solutions are favored. In practice this behaves exactly like the maximum volume bias in [9], and we refer to that paper for more detailed explanation on how this results in rectilinear structures. Fig. 7c illustrates how aligning observed walls with coordinate axes, combined with the maximum volume biasing and smoothness cost, enables the algorithm to complete missing data in a plausible way.

More precisely, we define

$$\begin{aligned} d_p(h) = &\sum_{r \in R_l} \alpha_1 H(h - h(r,p)) + \sum_{r \in R_p} \alpha_2 H(h - h(r,p)) \\ &+ \sum_{r \in R_s} \alpha_3 H(h - h(r,p)) + \sum_{\mathbf{x} \in P_l} \alpha_4 S(\mathbf{x}) H(x_z - h) \\ &+ \sum_{\mathbf{x} \in P_p} \alpha_5 S(\mathbf{x}) H(x_z - h) + \beta(h), \end{aligned} \qquad (3)$$

where R_l, R_p, R_s are LIDAR, SfM and sky rays, respectively, and P_l, P_p are the LIDAR and SfM points at point p, H the Heaviside step function, $S(\mathbf{x})$ the indicator function for the street mask, and $\alpha_{[1:5]}$ different weights for the input components. $h(r, p)$ is the height of a ray at point p, or ∞ if the ray does not pass through p or travels under a significant column of points before reaching p. The maximum volume biasing term $\beta(h)$ decreases for higher values of h.

To generate the smoothness cost s_{pq}, we analyze the point cloud for strong vertical planar structures. The smoothness costs is set to favor label transitions along such planar structures. In the absence of planar structures, high smoothness cost together with the maximum volume bias in the unary costs results in rectilinear completion of the segments.

The sky segmentation plays an important role in providing an upper limit for the height of a building. The sky rays shoot over the building roofs, providing a strong evidence about the building height. Together the LIDAR+SfM and the sky bound the correct label between high costs above and below. In the absence of reliable point measurements towards the top of the building, there will be a span of low cost layers between sky and point data, but the maximum volume bias will push the building height up to the level indicated by the sky rays.

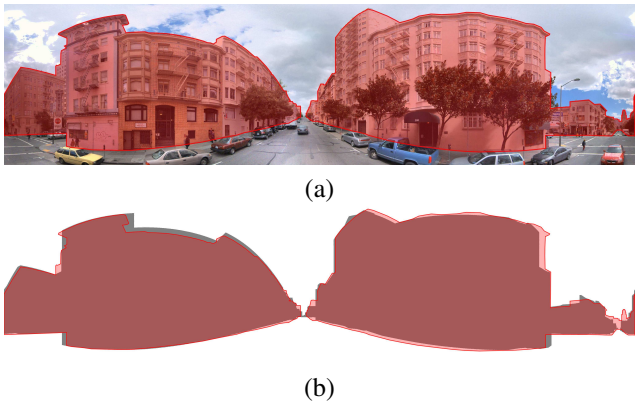
Once all the costs are determined, the graph-cut solver [2, 15, 1] is used to obtain optimal label for each cell.

4.1. Geometry Simplification

The footprint labels generated using the graph-cut algorithm can still be noisy and walls that are not axis aligned will show rasterization artifacts. Small structures on roofs of buildings, e.g., flag poles, antennas, etc. can introduce tiny shards into the reconstruction. Similar artifacts can occur if the input point clouds are not perfectly registered.

The result is cleaned up by iteratively joining tiny components with the neighboring components that share a long boundary and are close in height. The boundaries of the final segments are converted into polygons and each edge curve between two segments is simplified using recursive subdivision that approximates the curve by a piecewise linear curve formed by a subset of the original vertices. We stop the subdivision when the maximum deviation of the optimized boundary is below a specified threshold from the original. We generate the 3D meshes by extruding the footprint to create walls at appropriate height and by adding flat roofs.

A terrain model interpolated from the IMU track of the vehicle is used to drop components that would appear too low to be a building. This terrain is also used to clip the bottoms of the buildings to conform to street level. Figures 1a and 7 showcase the scale and accuracy of the reconstruction.



(a)

(b)

Figure 8: Ground truth, in red, overlaid on top of (a) original panorama and (b) a 2D projection of our reconstruction, in gray.

5. Results

In this section, we evaluate our algorithm both quantitatively and qualitatively on panoramic imagery and LIDAR data acquired from street level. We do this by gauging the accuracy of 2D projection of our results against a manually annotated ground truth. This allows us to assess how precise our reconstructed model looks from street level, which represents the objective of our method. Furthermore, we show that our automatic reconstruction compares favorably to a hand crafted commercial solution.

5.1. Data Collection

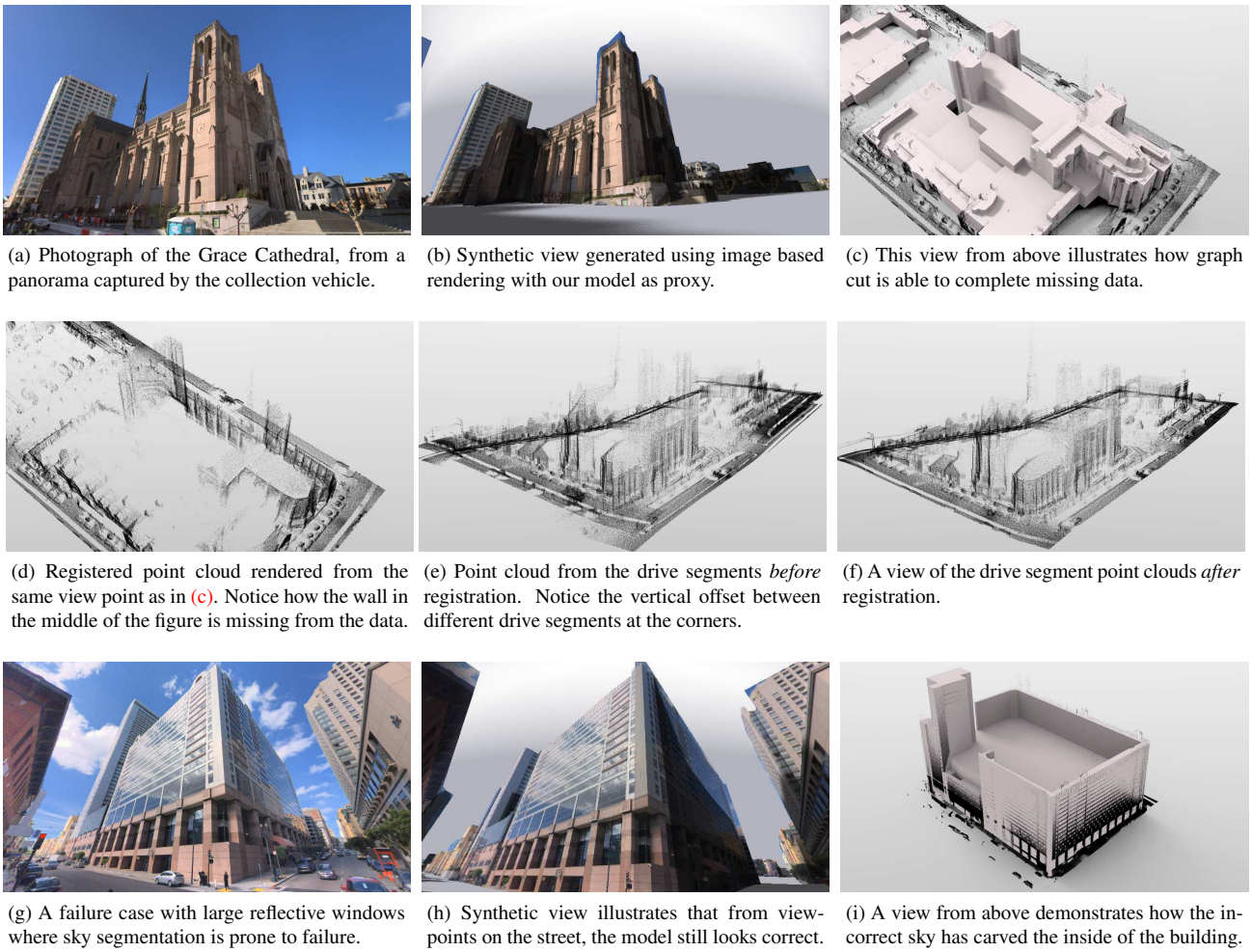
Our test dataset consists of panoramic images and LIDAR point clouds captured in the streets of San Francisco. In total, more than 180,000 panoramas captured every 4 meters cover most of the downtown area. LIDAR from 23 drive missions represents more than 6 TB of data. This dataset spans diverse urban neighborhoods, such as downtown areas with skyscrapers, and residential areas with small buildings and vegetation. San Francisco is a city with plenty of steep hills, further complicating the reconstruction task.

5.2. Ground Truth and Baseline

We created a ground truth by manually labeling building vs. non-building on street-level panoramic images. A total of 120 such images were chosen at random across the whole city. An example is illustrated on Fig. 8.

We compare the quality of our building reconstruction pipeline to a building footprint database from Sanborn Inc[2]. The database consists of 2D polygons that represent the building footprint as seen from an aerial view. For each 2D polygon, the base and the roof elevation are also provided.

[2] http://www.sanborn.com/services/3dcities



(a) Photograph of the Grace Cathedral, from a panorama captured by the collection vehicle.

(b) Synthetic view generated using image based rendering with our model as proxy.

(c) This view from above illustrates how graph cut is able to complete missing data.

(d) Registered point cloud rendered from the same view point as in (c). Notice how the wall in the middle of the figure is missing from the data.

(e) Point cloud from the drive segments *before* registration. Notice the vertical offset between different drive segments at the corners.

(f) A view of the drive segment point clouds *after* registration.

(g) A failure case with large reflective windows where sky segmentation is prone to failure.

(h) Synthetic view illustrates that from viewpoints on the street, the model still looks correct.

(i) A view from above demonstrates how the incorrect sky has carved the inside of the building.

Figure 7: Sample results of geometry reconstruction and registration.

If a building has complex geometry, it is approximated as a set of sub-buildings that have separate elevation data for each. The models represent the city at a similar level of abstraction that we would like to achieve using our automated pipeline.

5.3. Evaluation

The quality of our models is evaluated by comparing their 2D projection to the same view point as the ground truth panoramas with the manual annotation. We use the normalized symmetric difference between the 2D surfaces as a metric:

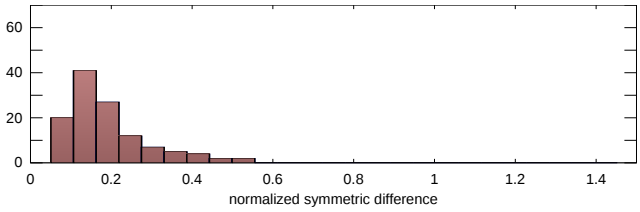
$$e(P) = \frac{|P \cup L| - |P \cap L|}{|L|} \quad , \tag{4}$$

where P is the 2D projection of our buildings, L is the manually labeled ground truth and $|.|$ represents the area of a polygon. The gray and red areas in Fig. 8 correspond to P and L, respectively.

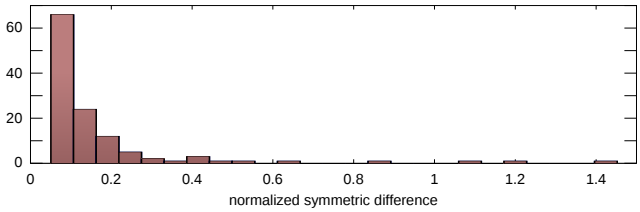
The result of the evaluation is plotted in Fig. 9 and shows that our reconstruction is on average more accurate than the high quality Sanborn model. A few outliers with larger difference are mostly due to trees interfering with building reconstruction.

Qualitative results highlighting the effects of the different components of our pipeline are shown in Fig. 7. The last row of Fig. 7 shows a typical failure case, where the sky is reflected from windows and sky segmentation masks part of the façade as sky. While we attempt to detect incorrect sky rays by not allowing points to pass under large columns of points, in this case there is just not enough evidence to do so. This is because structure from motion also fails on reflective surfaces and the infrared LIDAR sees through the windows.

Our method efficiently produces models that are both highly compact and accurate at city scale. The data registration and 3D City Modeling phases for the entire city of San



(a) Sanborn



(b) Our approach

Figure 9: Histograms of normalized symmetric difference (Eq. 4) with respect to the ground truth. Lower values indicate a more accurate reconstruction.

Francisco took only a day on 32 cores. The city partitioning step makes our approach highly parallelizable.

6. Conclusion and Future Work

We have demonstrated that the proposed method is able to accurately register huge sets of input data very quickly and consequently reconstruct accurate geometry. When projected to view-points where the acquisition vehicle visited, the resulting geometry matches well with manually annotated ground truth, outperforming commercial models.

The structure from motion process is very computationally expensive and we plan to seek faster alternatives to complete the depth information where it is needed and not provided by LIDAR. Further robustness measures are required to improve sky segmentation and deal with noise such as trees and big cars.

References

[1] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *TPAMI*, 2004.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *TPAMI*, 2001.

[3] N. Cornelis, B. Leibe, K. Cornelis, and L. Gool. 3D Urban Scene Modeling Integrating Recognition and Reconstruction. *IJCV*, July 2008.

[4] M. Cummins and P. Newman. Probabilistic Appearance Based Navigation and Loop Closing. In *ICRA*, 2007.

[5] P. Debevec, Y. Yu, and G. Borshukov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *EGWS*, 1998.

[6] C. Frueh, S. Jain, and A. Zakhor. Data Processing Algorithms for Generating Textured 3D Building Facade Meshes from Laser Scans and Camera Images. *IJCV*, 2005.

[7] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Manhattan-World Stereo. In *CVPR*, 2009.

[8] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing Building Interiors from Images. In CVPR, 2009.

[9] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 2010.

[10] D. Gallup, J.-M. Frahm, and M. Pollefeys. A heightmap model for efficient 3d reconstruction from street-level video. In *3DPVT*, 2010.

[11] D. Gallup, J.-M. Frahm, and M. Pollefeys. Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction. In *CVPR*, June 2010.

[12] D. Gallup, M. Pollefeys, and J.-M. Frahm. 3d reconstruction using an n-layer heightmap. In *Proceedings of the 32nd DAGM conference on Pattern recognition*, 2010.

[13] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-View Stereo for Community Photo Collections. In *ICCV*, 2007.

[14] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *SIGGRAPH*, 1996.

[15] V. Kolmogorov and R. Zabin. What Energy Functions can be Minimized via Graph Cuts? *TPAMI*, 2004.

[16] A. Levin and R. Szeliski. Visual Odometry and Map Correlation. *CVPR*, 2004.

[17] 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 *CVPR*, 2009.

[18] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real Time Localization and 3D Reconstruction. In *CVPR*, 2006.

[19] D. Nister, O. Naroditsky, and J. Bergen. Visual Odometry. In *CVPR*, 2004.

[20] M. Pollefeys and *et al.* Detailed Real-Time Urban 3D Reconstruction from Video. *IJCV*, 2008.

[21] T. Pylvänäinen, K. Roimela, R. Vedantham, J. Itaranta, R. Wang, and R. Grzeszczuk. Automatic Alignment and Multi-View Segmentation of Street View Data using 3D Shape Priors. In *3DPVT*, 2010.

[22] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Trans. Graph.*, 2004.

[23] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. *ACM Trans. Graph.*, 2006.

[24] V. Verma, R. Kumar, and S. Hsu. 3D Building Detection and Modeling from Aerial LIDAR Data. In *CVPR*, 2006.

[25] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based Street-Side City Modeling. *ACM Trans. Graph.*, 2009.

[26] L. Zebedin, J. Bauer, K. F. Karner, and H. Bischof. Fusion of Feature- and Area-Based Information for Urban Buildings Modeling from Aerial Imagery. In *ECCV*, 2008.