Watertight Floor Plans Generated from Laser Range Data

Eric Turner

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-69
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-69.html

May 15, 2013

Acknowledgement

I would like to thank my advisor and the reader.

Watertight Floor Plans Generated from Laser Range Data

Eric Turner

Advisor: Avideh Zakhor

U. C. Berkeley, Department of Electrical Engineering and Computer Science

elturner@eecs.berkeley.edu

Abstract

We describe an approach to automatically generate a floor plan for the interior environment of a building using laser scan data. This floor plan is meant to accurately indicate the positions of walls within an area of interest in the building. The proposed algorithm separates the floors of a building scan, selects a representative sampling of wall scans for each floor, and triangulates these samples to develop a watertight representation of the walls for each of the scanned areas. Curves and straight line segments are fit to these walls, in order to mitigate any registration errors from the original scans. This method is not dependent on the scanning system and can successfully process noisy scans with non-zero registration error. Most of the processing is performed after a dramatic dimensionality reduction, yielding a scalable approach. We demonstrate the effectiveness of our approach on a three-story point cloud from a commercial building as well as on the lobby and hallways of a hotel.

1. Introduction

The use of laser scanning technology is becoming a vital component of building construction and maintenance. During construction, laser scanning can be used to record the as-built locations of HVAC and plumbing systems before drywall is installed. In existing buildings, blueprints are often outdated or missing, especially after several remodelings.

The point clouds generated from laser scans typically represent huge amounts of data that may contain noise or gaps. There may exist excess clutter such as furniture or other non-building objects. An existing and more usable format in the architectural field is a 2D floor plan for the zones of interest. The ability to separate these building elements from the rest of the point scans and to generate a coherent floor plan can potentially enhance semantic understanding of a point cloud representing an architectural environment.

Accurate floor plans can also be used to generate full 3D models of a building. The input floor plans of these modeling systems should not contain temporary objects or clutter, such as furniture. The techniques described in [10, 13] spend significant effort to discover walls and shapes in a rasterized floor plan. Having a plan that is already represented by parametric curves and lines, such as the output of the approach to be described in this thesis, can facilitate modeling algorithms that incorporate 2D information [3, 4, 10, 13]. The resulting 3D models can be used for virtual reality or in-building navigation purposes. Both these 3D models as well as 2D floor-plans can be used to generate semantic Building Information Models (BIM), which can be used for building maintenance and energy-efficiency analysis [15].

In this thesis we describe an approach to automatically generate a floor plan for the interior environment of a building using laser scan data. This floor plan is meant to accurately indicate the positions of walls within an area of interest in the building.

2. Background

Point cloud scans of building interiors can be captured in a variety of ways. With static scanning, a stationary scanner is manually placed at a finite number of locations in the building to collect scans. In mobile scanning, a single scanning system collects data continuously while moving through the environment. In both cases, scans taken from different locations must be registered against each other, so that all scan points are represented in the same coordinate frame. Any errors in this registration process can cause errors in the resultant point cloud. Applications that analyze this point cloud must take these potential errors into account.

An architectural floor plan is defined to represent a subset of the interior walls and hallways of a building. It is often preferred that floor plans contain semantic information, such as the swing direction of doorways or restroom labels. These features are not a requirement and the processing required to produce them is beyond the scope of this thesis. The shape and configuration of wall placements in floor plans are not arbitrary, but adhere to common practices and patterns. Walls customarily are simple geometric shapes, such as straight line segments or smooth curves [15]. The placement of walls

with respect to one another is also methodical. Many building floor plan designs utilize patterns and regularity to reduce the complexity of the design and construction process [6]. The symmetric and repeating pattern common in many buildings can be used to reduce noise in an estimated model. Since the techniques proposed in this thesis are meant to be dispatched on subsets of a building's interior, the patterns across an entire building cannot always be utilized.

3D modeling of building environments is a well-studied field. Aerial and terrestrial laser scanning of urban environments has yielded accurate, detailed models of external architectures [14, 17]. Since exterior scans typically pass through windows and other openings in a building façade, they can be used to create a sparse representation of the interior floor plan [7].

Interior scanning has traditionally been used for robotic navigation in indoor environments. The rough locations of walls and objects are necessary to avoid collisions while moving about the environment. Generating floor plans for modeling, however, requires a much higher degree of precision in wall location estimates[12]. Given these antecedent studies, techniques have been developed using a single horizontal scanner collecting about the yaw direction [16]. Constructing full 3D scans allows for more sophisticated means of identifying walls from other obstacles in a building, which allows for highly accurate representations of only the features of the environment in which we are interested. In this scenario, one can compute a top-down 2D histogram of the full 3D point densities projected across the x-y plane [12]. Areas with high density are considered likely to be wall locations, since they tend to indicate large vertical surfaces. While clutter is mitigated by being less represented in the histogram than walls, no direct measures are taken to remove outlier samples before line-fitting. Further, each story of a building must be processed separately.

Previous approaches to floor plan modeling typically assume walls are well-fitted by straight line segments and whether these fitted models are watertight is not guaranteed [11, 12, 16]. Many architectural designs incorporate curved features, which would not be modeled accurately by these approaches. The absence of any water-tightness guarantee requires extensive post-processing to be devoted to removing disconnected and outlier segments that are interpreted as noise.

This thesis presents a means of generating a floor plan that fits both curves and straight segments with walls that are guaranteed to be watertight. In Section 3 we describe our approach. Sections 4 and 5 include results and conclusions respectively.

3. Proposed Approach

Given a point cloud representing a subset of the interior of a building that can cover multiple stories, we propose a three-step process to generate a watertight floor plan for each story in the point cloud by fitting line segments and curves to model features. First, we extract wall sample locations for each story. This process is done using a 2D projection and density analysis, similar to [12]. The output is a subset of samples that are representative of each wall on each story, as discussed in Sections 3.1 and 3.2. Second, the Delaunay Triangulation of these samples is computed and each triangle is labeled as "inside" or "outside" using a 2D version of the Eigencrust algorithm [9]. This process is discussed in Section 3.3. The edges

that border these two labels are output as wall locations, as detailed in Section 3.4. Thirdly, noise reduction is achieved by fitting these facets to line segments and curved components using Random Sample Consensus (RANSAC) [5], as discussed in Section 3.5.

3.1. Finding Wall Samples

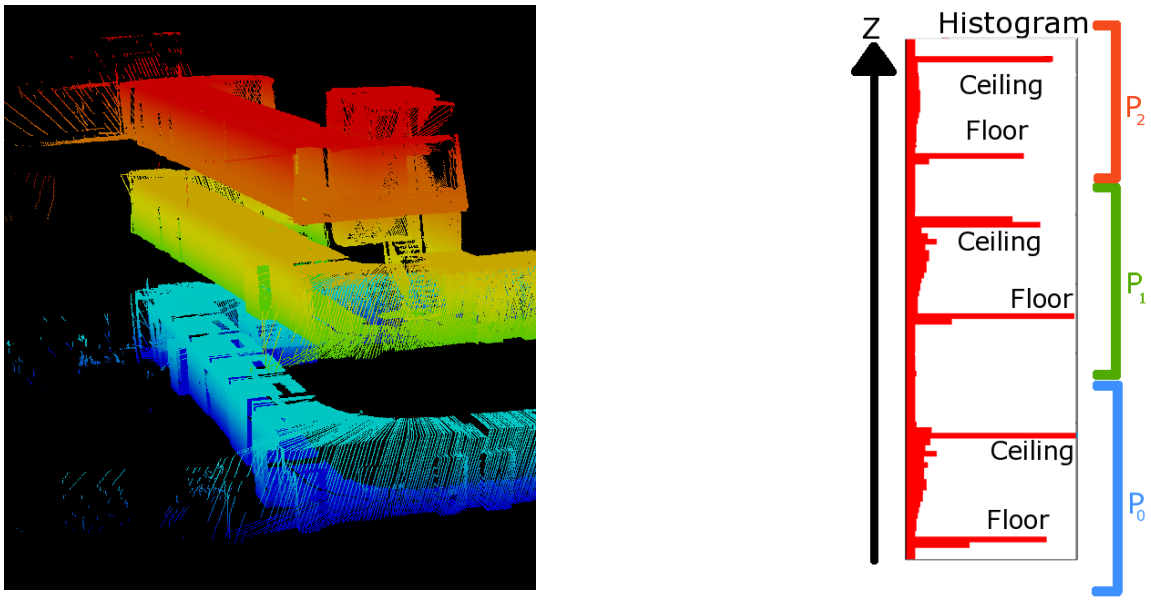
Given a 3D input point set, P, we wish to generate a floor plan for each story represented. The coordinate system is defined so that the z-component represents height. First, we compute how many stories are present in P and partition P into separate point sets P_0, P_1, ..., P_f for each level by height, as shown in Figure 1. A histogram is calculated for the distribution of the heights of all points $p \in P$. Next, we determine which bins of this histogram represent the locations of floors and ceilings.

The bins corresponding to floors and ceilings satisfy two conditions both apparent in Figure 1(b): they are local maxima and they have higher count than most other bins. Therefore, if we sort all the local maxima bins from highest count to lowest count, the floors and ceilings appear in front of this sorted order with with large counts, while all other bins appear at the end of this sorted list with lower counts. Detecting the largest discontinuity in this sorted list results in separating the potential floor/ceiling bins from the others.

The bin with the lowest z-elevation that satisfies these requirements is assumed to be the floor of the first story. The lowest floor/ceiling bin that is at least a minimum wall height threshold, typically taken to be 2 meters, above the current floor bin is considered the ceiling bin for that floor. The next highest floor/ceiling bin is taken to be the floor height of the next story. This procedure continues for all stories. A partition level is taken to be halfway between the ceiling of one story and the floor of the next story. These levels partition the point set into f separate stories $P_0 \cup P_1 \cup ... \cup P_{f-1} = P$.

For each story k, we wish to determine what subset of points in P_k represent walls. Our goal is to find these wall locations with a sparse sampling of points $S_k \subseteq P_k$. By defining a grid along the x-y plane, we can find coordinate locations that are likely to represent walls. The width of each grid cell is denoted by d_s, which should be smaller than the minimum feature size expected in a floor plan. A typical value for this parameter is 5 cm. This grid partitions P_k laterally. For a grid cell g, let the neighborhood set N_g be the subset of points of P_k that are within a horizontal distance of d_s from the center of g, as shown in Figure 2. Thus a given point can be in the neighborhood of multiple grid cells.

We perform the following checks on N_g to determine whether this neighborhood represents a portion of a wall. The first assertion is that a wall must be densely sampled, so if $|N_g|$ is less than a threshold, it is rejected. Our scanners create point-clouds with thousands of points per square meter of surface, so a threshold of $|N_g| \geq 50$ points is sufficient. The second check is to verify the height of a wall. The vertical support of N_g must be at least the minimum wall height threshold mentioned earlier. We wish to verify that the distribution of points is uniform vertically, which signifies that a flat surface was captured. For this uniformity check, we compute the histogram of the heights of N_g with 16 bins and enforce that at



(a) (b)

Figure 1. A point set P, shown in (a), is separated into three stories P_0, P_1, and P_2 based on a histogram analysis of point heights, as shown in (b).

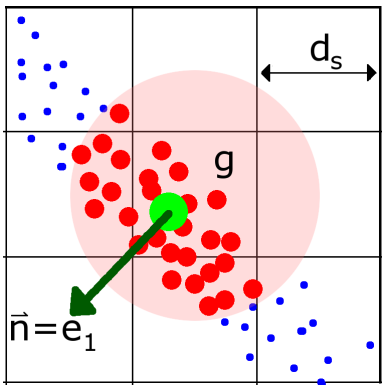


Figure 2. A point set P_k (small dots in blue) is partitioned by a grid in the x-y plane (lines in black). For each grid cell, a neighborhood of points (medium dots in red) is computed and the median position of the points (large dot in green) is taken as a wall sample. The normal of this neighborhood (vector in dark green) is computed using PCA.

least 12 of these must be non-empty. Additionally, at least three of the highest four bins must be non-empty, which ensures a neighborhood extends all the way to the ceiling. If these requirements are fulfilled, g is said to represent a wall. The median horizontal position for the elements of N_g is computed. If this median lies within the bounds of g, then it is taken as a "wall sample" for g.

The normal vector for this wall sample is calculated using Principal Components Analysis (PCA) [8]. Let C be the 2×2 covariance matrix of the x-y positions of the elements of N_g and e_1, e_2 be the eigenvectors of C with corresponding eigenvalues λ_1, λ_2. Let $\lambda_1 \leq \lambda_2$. The normal vector is chosen to be $\vec{n} = e_1$, as shown in Figure 2. The confidence of this normal estimate is computed as $c = 1 - \frac{2\lambda_1}{\lambda_1+\lambda_2}$. If c is less than 0.15, then the neighborhood points are not well-aligned and

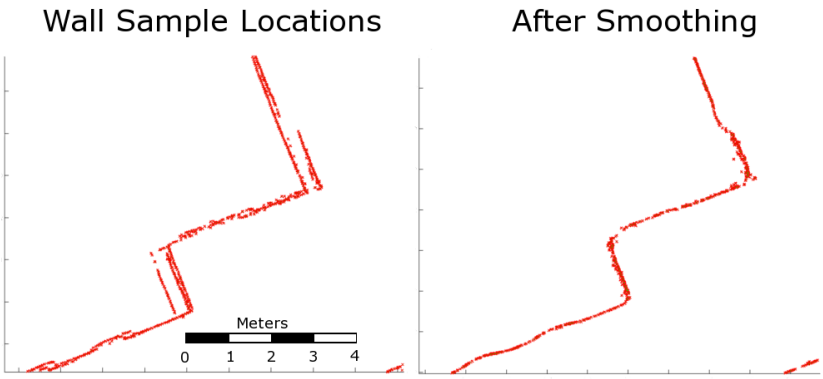


Figure 3. Average filtering reduces occurrence of registration errors exposed in wall sampling.

rejected as the location for a wall.

If pose information for the scanner is known for each point, this normal vector's direction may be flipped to guarantee that it points towards the laser scanner. Any wall samples captured far away from the scanner are thrown away as outliers. This distance is typically 5-10 meters. If a feature is not scanned within this distance, it is only partially captured and cannot be accurately represented. Performing the above operation for all grid cells results in a set of wall sample points $S_k \subseteq P_k$.

3.2. Smoothing Wall Samples

A crucial step in generating a point cloud is to register the poses of separate scan positions to a common coordinate system. Any error in this process may result in duplicate instances of a scanned wall with slight misalignments. Each instance of a wall may appear in S_k and if the registration error is significant smoothing may be required to force these wall instances to become aligned.

For a given sample $s \in S_k$ with normal vector $\vec{n}$, its smoothing set is a subset of S_k. The position of s is translated to be the mean position of this smoothing set. The smoothing set contains all samples that are (a) within some smoothing distance of s along the $\pm\vec{n}$ direction, (b) no more than $3d_s$ away from s in the direction orthogonal to $\vec{n}$, and (c) whose normals are aligned to within $\frac{\pi}{8}$ radians of $\vec{n}$. The smoothing distance must be less than the minimum allowable spacing between parallel walls, but at least as large as the expected registration error. This step is not necessary if registration error is smaller than d_s. An example of this smoothing process is shown in Figure 3.

3.3. Labeling Triangulation of Wall Samples

We wish to determine the topology of the 2D sample set S_k for each story $k \in \{0, 1, ..., f-1\}$. This task is accomplished using a 2D variant of the Eigencrust algorithm [9]. This will partition 2D space into regions labeled "inside" and "outside" and export the borders between these regions. These borders form the faces of a 2D simplicial complex, which are guaranteed to be watertight and not self-intersecting. Eigencrust has been shown to be more robust to outlier samples than similar

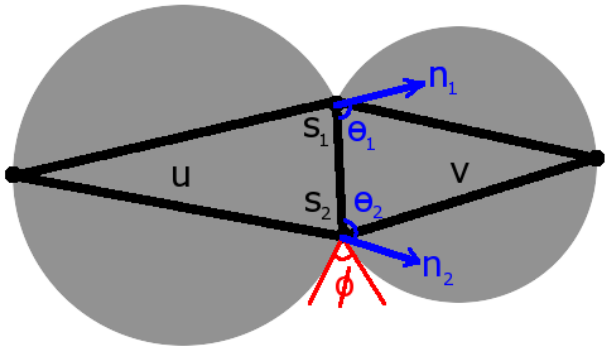


Figure 4. The angles of intersection between two triangles u and v, which share the edge $\overline{s_1 s_2}$. These values are used in computing weights between triangle pairs.

algorithms, such as Powercrust [9, 1].

The Eigencrust algorithm first computes the 2D Delaunay Triangulation, T, for the sample set S_k plus four bounding box corners. Each triangle $t \in T$ is labeled as either a pole or non-pole, where poles are defined to be the triangles whose circumcenters represent the extrema vertices of Voronoi cells. These poles are structured in a sparse graph $G = (V, E)$, where each pole is a node in the graph and edges are connected between poles with weights corresponding to the relative geometry of the triangles. Edges with positive weights indicate that triangles should have the same labeling, while negative weights indicate that the triangles connected should be labeled oppositely. Generalized eigensystems are solved in order to determine the best triangle labelings to fit these connections. For our 2D varient of Eigencrust, we keep the same criteria for placing edge weights as in [9], but modify the negative edge weight values. We have additional information of normal vectors for each sample location. If a negative edge is placed between two triangles u and v, we use the weighting:

$$w_{u,v} = -e^{4+4cos\phi+2sin\theta_1+2sin\theta_2} \tag{1}$$

where these parameters are shown in Figure 4. The value ϕ is defined as the angle at which the circumcircles of u and v intersect. The intersection of the triangles u and v is a line segment whose endpoints are samples s_1 and s_2, which have normal vectors $\vec{n}_1$ and $\vec{n}_2$ respectively. The values of θ_1, θ_2 are defined to be the angles between $\overline{s_1 s_2}$ and $\vec{n}_1, \vec{n}_2$, respectively.

Much like the edge weights used in [9], this function was not derived using any formal analysis, but rather by empirically observing the behavior of the system with different weight values. This weight is modified from [9] in order to include normal vector information for each wall sample. This weight is defined to have a large magnitude when both the normal vectors are perpendicular to the line $\overline{s_1 s_2}$, which is a strong indication of a wall in the original point cloud P_k.

Positive edges are added to the graph as follows. For each pair of samples s,$s' \in S_k$ such that (s,s') is an edge of the Delaunay Triangulation T, let u and v be the poles of s, and u' and v' be the poles of s'. Each of the pairs (u, u'), (u, v'), (v, u'), and (v, v') are edges in E. If a negative edge is not already defined, each of these pairs has positive edge weight:

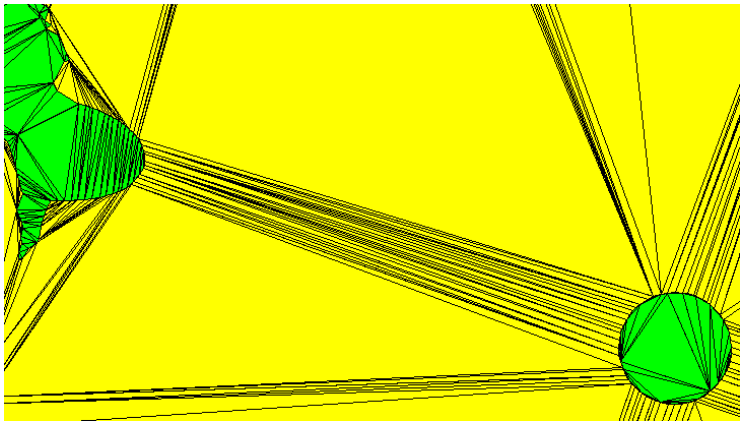


Figure 5. An example triangulation of wall samples. Each triangle is denoted as "inside" (yellow) or "outside" (green).

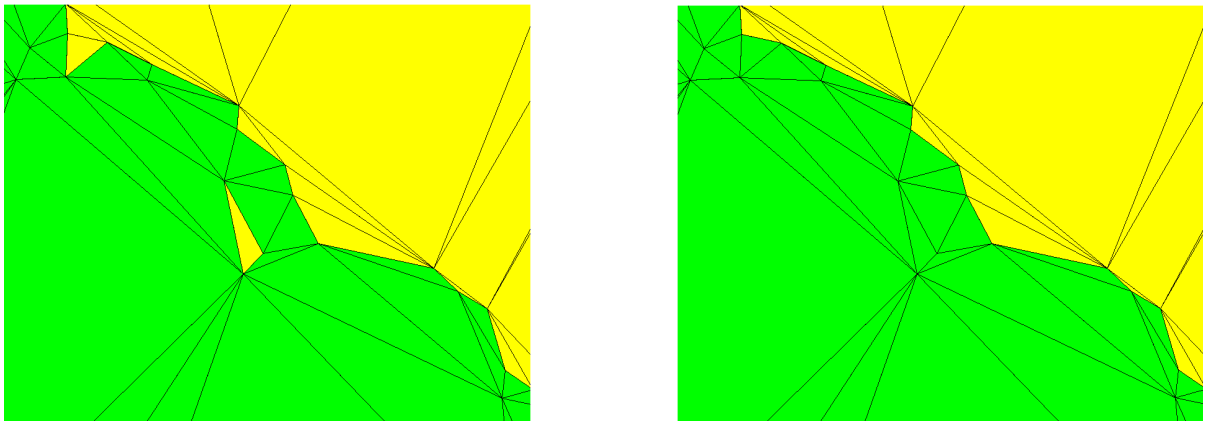
$$w_{u,u'} = e^{4-4cos\phi} \tag{2}$$

where ϕ is again defined as the angle at which the circumcircles of u and u' intersect. This weight is identical to the one used in [9], and again is based on empirical observations of how the system behaves.

We can constrain some triangles to be interior or exterior based on a priori information. The label "inside" refers to locations of open area within a building where a person can exist. The term "outside" refers to all other areas, which include both the exterior of the building as well as the areas inside walls or other solid spaces. We can immediately force the label of "outside" to any triangles connected to the bounding box corner vertices. If the pose information for the scanner is available, we can force the label of "inside" to a subset of triangles. First, we investigate whether the 2D line-of-sight from a scanner pose to a scan sample crosses triangles. If the center 50% of this laser travel distance crosses a triangle, that triangle is intersected by that scan line. If a triangle is intersected by 10 or more scan lines, it is assumed to be "inside". Second, all triangles that contain a pose position of a scanner are marked as interior. If a mobile scanning system is used, then the path traversed by that system can also be used to mark interior triangles. Each pair of adjacent pose positions represents a line segment in 2D space. If both of these poses contributed scans to the wall sample set S_k, then any triangles that are intersected by this line segment are also constrained to be interior.

The non-pole triangles are then put into a corresponding graph. Since the edge-weights of this non-pole graph in the 3D Eigencrust algorithm use the aspect ratio of bounding triangular faces, there is not a direct translation of these weight calculations to 2D. Thus the same weights are used as for the pole graph G, defined in Equations 1 and 2. A corresponding generalized eigenvalue system is solved for this non-pole graph, which provides a labeling for all triangles in T. An example output of this triangulation labeling process is shown in Figure 5.

All the "outside" nodes are collapsed into one outside super-node and all "inside" nodes are collapsed into one inside super-node, creating graph G'. The nodes and edges of G' are then converted into a matrix $L \in \mathbb{R}^{|G'|\times|G'|}$. Each element is



(a) (b)

Figure 6. Original triangulation labeling (a) is cleaned by removing small unions and sharp protrusions (b).

defined as $L_{ij} = L_{ji} = -w_{i,j}$ with the diagonal $L_{ii} = \sum_{j \neq i} |L_{ij}|$. The diagonal matrix D is the same size as L and has the same diagonal elements as L. These values are used to find the solution to the generalized eigensystem $Lx = \lambda Dx$. The eigenvector x associated with the smallest non-zero eigenvalue λ is taken as the result. Solving this system minimizes the total positive edge weights that connect oppositely labeled triangles and negative edge weights that connect triangles with the same labeling [9].

3.4. Forming Floor Plan Boundary Edges

We have found the following post-processing clean-up techniques useful for increasing the quality of the output model. First, we represent this topology of triangles as a set of unions. Any subset of triangles with the same label that forms a connected component is represented as a single union using the Union-Find algorithm [2]. Unions that are composed of a small number of triangles are most likely mislabeled, since the smallest building features should still be sufficiently sampled. The appropriate cut-off value depends on the value of d_s used to form samples, but a value of 30 triangles is typically used. All inside unions that have fewer than this many triangles are relabeled as outside. The set of unions is recomputed and then all outside unions that meet this criterion are relabeled as inside. Another post-processing smoothing process is to remove jagged edges from the boundaries between labelings. Every triangle has three neighboring triangles, each sharing one of its edges. If a triangle $t \in T$ has only one neighbor that shares the same labeling and that neighbor is connected to t's shortest edge, then t is considered to be a protrusion. It is unlikely for building geometry to be correctly represented by such a protrusion, so all protrusions labeled as inside are relabeled as outside. After this relabeling, all outside protrusions are found and relabeled as inside. An example of this processing is shown in Figure 6.

The set of edges in T that are shared by an "inside" triangle and an "outside" triangle are exported as boundary edge set, B. Each element in B is a line segment, which in total constitute a water-tight floor plan of the scanned area.

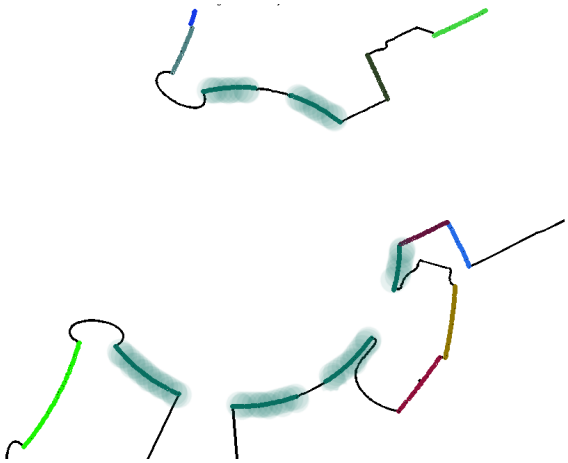


Figure 7. The walls of a circular room are fit to the same circle model (shown highlighted in teal), even though they are comprised of several connected components.

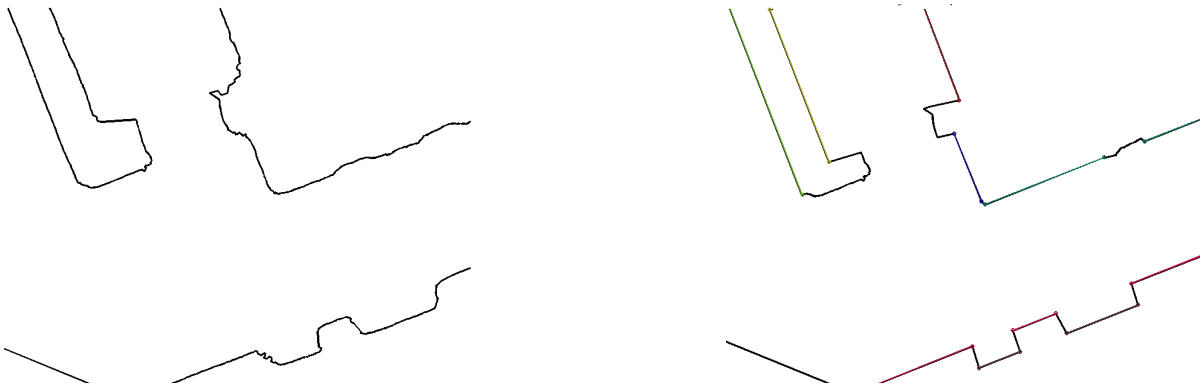
3.5. Fitting Edges With Lines and Curves

Since there may exist areas in the building geometry that are insufficiently scanned, or errors incurred during point-cloud registration, it is important to utilize common architectural patterns to reduce these artifacts. Such processing includes fitting common architectural patterns to the derived geometry of the floor plan, such as curves or lines. Additionally, many applications require a floor plan to be composed of parametric shapes.

Local model-fitting approaches such as region growing are sub-optimal because unconnected architectural features may use the same geometric model. For example, the back walls in a row of offices may lie on the same plane, even though they are in different rooms. Since we need to fit both curves and straight lines simultaneously, a reasonable technique for this situation is one that is non-local and flexible, such as Random Sample Consensus (RANSAC) [5].

We apply RANSAC to the subset of samples used as vertices in the boundary edges B. Each iteration randomly picks three samples from this set that have not yet been associated with a model. These three points uniquely define a circle. A line of best fit can also be obtained for these points. Both the circle and the line models are compared to the subset of samples still unassigned. Inlier sets for both of these models are computed. An inlier is an unassigned sample that is both within a threshold distance of a model and whose normal vector is within a threshold angle to the model's normal vector at that location. Only models that exceed a specified minimum number of inliers are considered. The line or circle model found with the smallest average error is returned as a valid model and its inlier vertices are no longer considered for subsequent models.

This process continues until no new parametric model can be found. The result is a set of models, where each model has a set of inlier samples and a parametric representation of either a line or a circle. The topology defined in B has not been altered, so its edge elements can be used to partition the inlier vertices of each model into a set of connected components, again with Union-Find [2]. Any of these connected components composed of too few samples is most likely a misclassification. Thus



(a) (b)

Figure 8. (a) Watertight wall edges; (b) curve-fitting via RANSAC reduces sample location error and sharpens corners.

the elements of any of these components with fewer than 15 samples are reset to be unassigned. Additionally, we wish to encourage models to extend along the edges defined in B. If an unassigned sample is within 15 edge hops to samples belonging to a model, that sample is associated with the closest model. These two steps encourages outlier samples to belong to models that are topologically close. These steps also grow models to be adjacent, encouraging sharp corners.

Once the revised parametric models and their respective inliers are computed, the inlier samples and edges in B are replaced by edges that conform exactly to their models' locations. This process reduces the overall number of samples and removes small perturbation errors from the floor plan. Figure 7 demonstrates the results of this process, which fits a circle to the walls of a round room with several entrances. Figure 8 shows how this process can also be applied to straight walls.

4. Results

Our approach can be applied to either static or mobile scan systems and to point sets representing one or more stories. Figure 9 shows the processing of a scan of three stories of hallways in an office building. The input point-set has 17.4 million elements and the resultant wall samplings for each story are 5,544 samples, 5,357 samples, and 3,265 samples from bottom story to top respectively. Our test code, written in unoptimized MATLAB and run on a personal laptop, processes these three stories in 371.4 seconds total. These results show how our algorithm enforces water-tightness by fitting walls to the ends of any unscanned sections, such as the hallways labeled 1 and 2 in the third column of Figure 9(c). The resultant models are compared against the current building floor plans, as shown in the fourth column of Figure 9. In areas where the wall sampling is dense, our generated floor plan is accurate with respect to these ground-truth blueprint. The level of accuracy is dependent on the input point-cloud, but typical results yield floor plans whose features are within 5 cm of ground truth.

These models are also shown in Figure 10. This compares the automatically generated models of this thesis with hand-drawn results, which were derived from the original point-cloud and wall samples. Since these hand-drawn models are created by a user with knowledge of the building environment, they represent the best-case scenario of generated floor plans. The first three rows show the models from Figure 9, while the last row shows an office building that contained mostly cubicles.

As shown, our algorithm matches well with the ground-truth. The largest disparity can be seen in the final row, where a large space labeled with "1" is removed by the automatically generated model. This space had poor sampling and highlights a short-coming of this approach. If a room is poorly scanned or unvisited, our technique will not result in an accurate model. This behavior can only be overcome with more comprehensive input.

Figure 11 shows the processing of a scan taken of a single level of a hotel. The input point-cloud has 5.8 million elements and the wall sampling produced 13,557 samples. This set demonstrates a model that is composed of both curves and straight line-segments. Figure 12 shows the lobby of this building, which has been statically scanned. This dataset has an input point-cloud of 2.7 million points from which 3,568 2D wall samples were extracted. This model demonstrates the ability of our algorithm to capture structural features such as columns. It shows how areas of high curvature can be modeled just as effectively as straight walls.

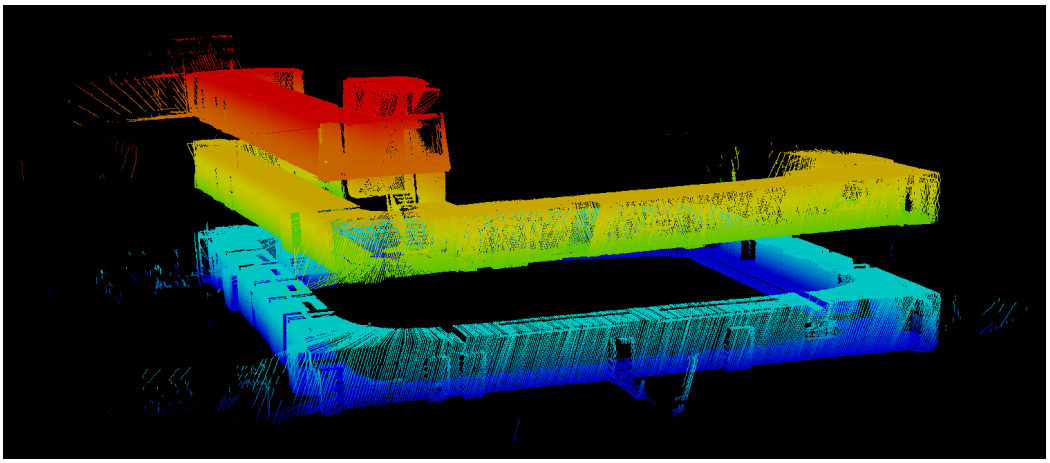
While enforcing watertight models allows for walls to be created in areas that are not scanned, it may also cause undesired homography changes. If two parallel walls are arbitrarily close, the space between them may become incorrectly labeled with neither wall existing in the final model. An example of this issue can be seen in the north-west stairway in Figure 9(d), labeled 3. This artifact details a fundamental concern with algorithms that search for a globally optimum solution: input noise in one area may cause artifacts in other areas. An approach that does not compute a global solution, but processes the input on a local scale, will confine the affects of any error to solely the region in which it occurs. Global algorithms are often still preferable, since information from neighboring parts of the input can also help the classification of a particular region.

Another limitation is that the walls placed in areas of sparse scanning may be inaccurate. The corridor in the south-east corner of the floor plan labeled 2 in Figure 9(c) is not fully scanned, so a wall is represented at the edge of the scan area, even though the hallway continues much further in reality. While the sparsity of a scan is typically due to the placement of the scanning system, it can also be caused by the partitioning of the original point-cloud into separate stories. Note that the south-west stairwell in the same model, labeled 1, is sparsely sampled and as a result the north wall is incorrectly modeled at an angle. In this case, the wall in question was thoroughly scanned, but higher in the stairway on the next story. The immediate partitioning of levels in a point-cloud causes cross-level scans to become under-represented and introduces sparsity into the wall sampling.

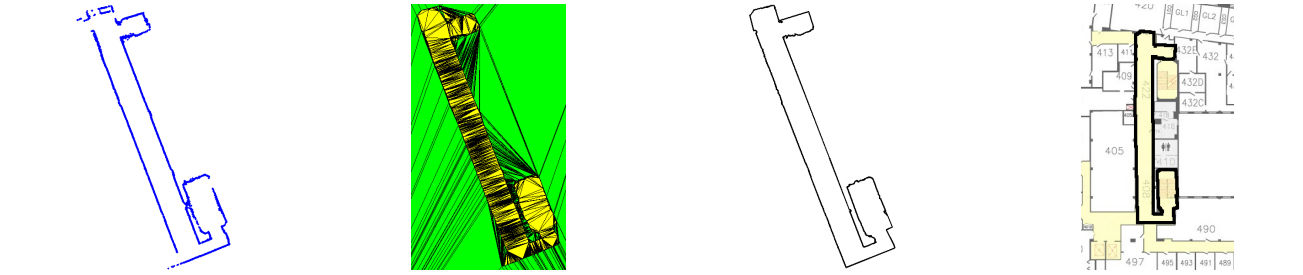
Each dataset shown has an input of millions of points, but uses the wall samples, a much sparser representation, for all topology processing. This reduction allows for many more model-fitting iterations than if fitting was performed on the original point-cloud.

5. Conclusions and Future Work

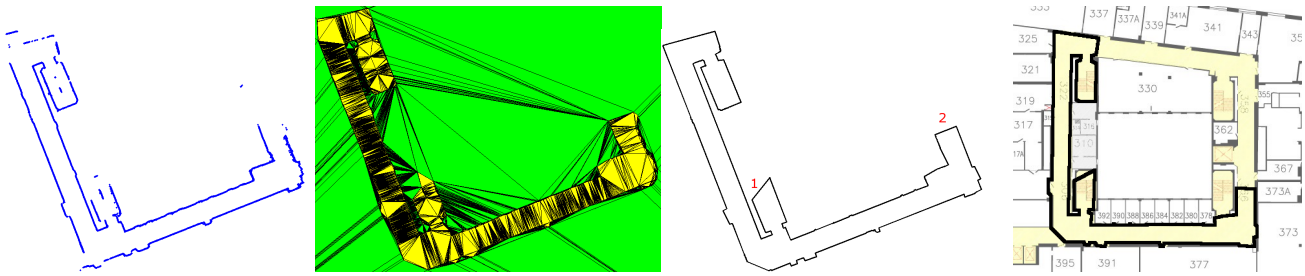
This thesis details a technique to generate 2D architectural floor plans. By projecting each story's point set to two dimensions and performing a density analysis, we can yield an increase in wall location accuracy compared to using a direct 3D



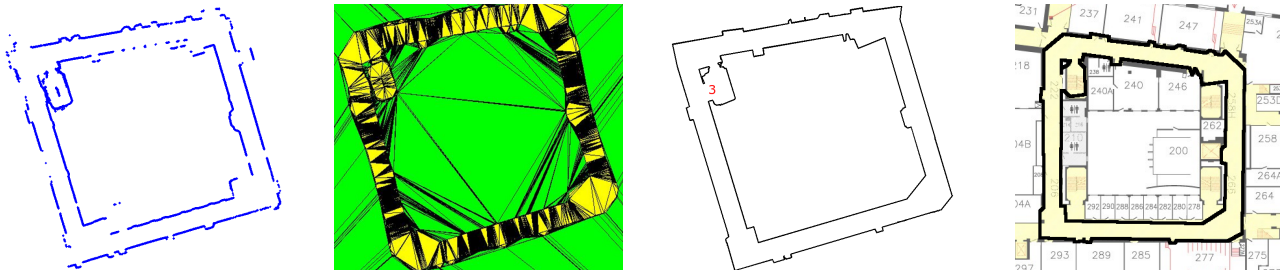
(a)



(b)



(c)



(d)

Figure 9. (a) Full point cloud for three-story model, taken with mobile scanning system; (b-d) Processing of each story, with (left to right) wall sample locations, triangulation labeling, watertight curve-fit model, and comparison against ground-truth blueprints.

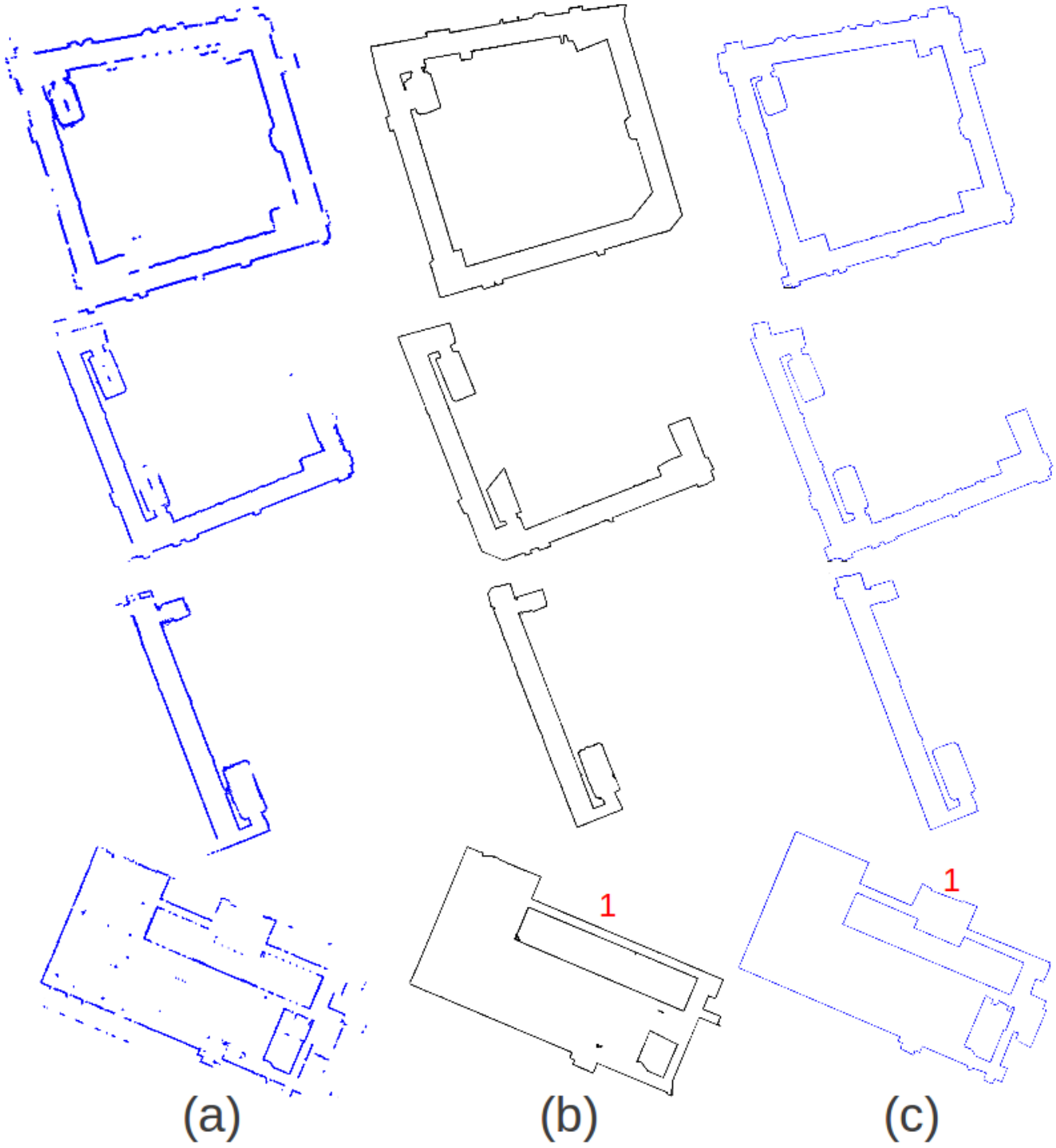


Figure 10. (a) Wall sampling of several models; (b) The final automatically generated watertight floor plan; (c) a hand-drawn representation of the same area, created by a user with familiarity of the building from the original wall samples.

approach. The Eigencrust algorithm was chosen for this problem due to its robustness to noise in the input points. While Eigencrust was designed to ignore outlier points, this application presents potential sources of input noise that are more difficult to overcome. Misregistration of the input data can result in a single wall being represented by a "double surface," as

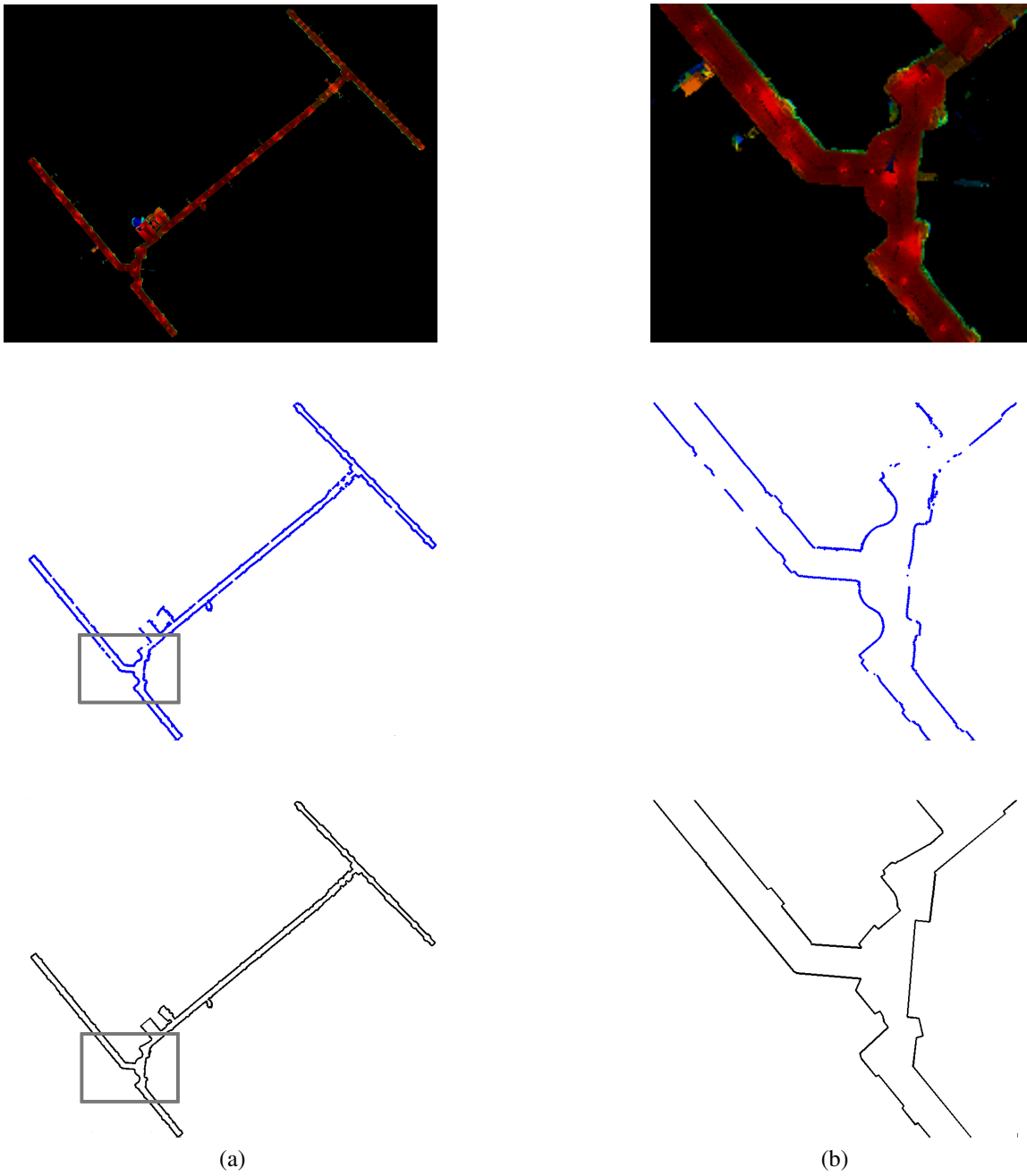
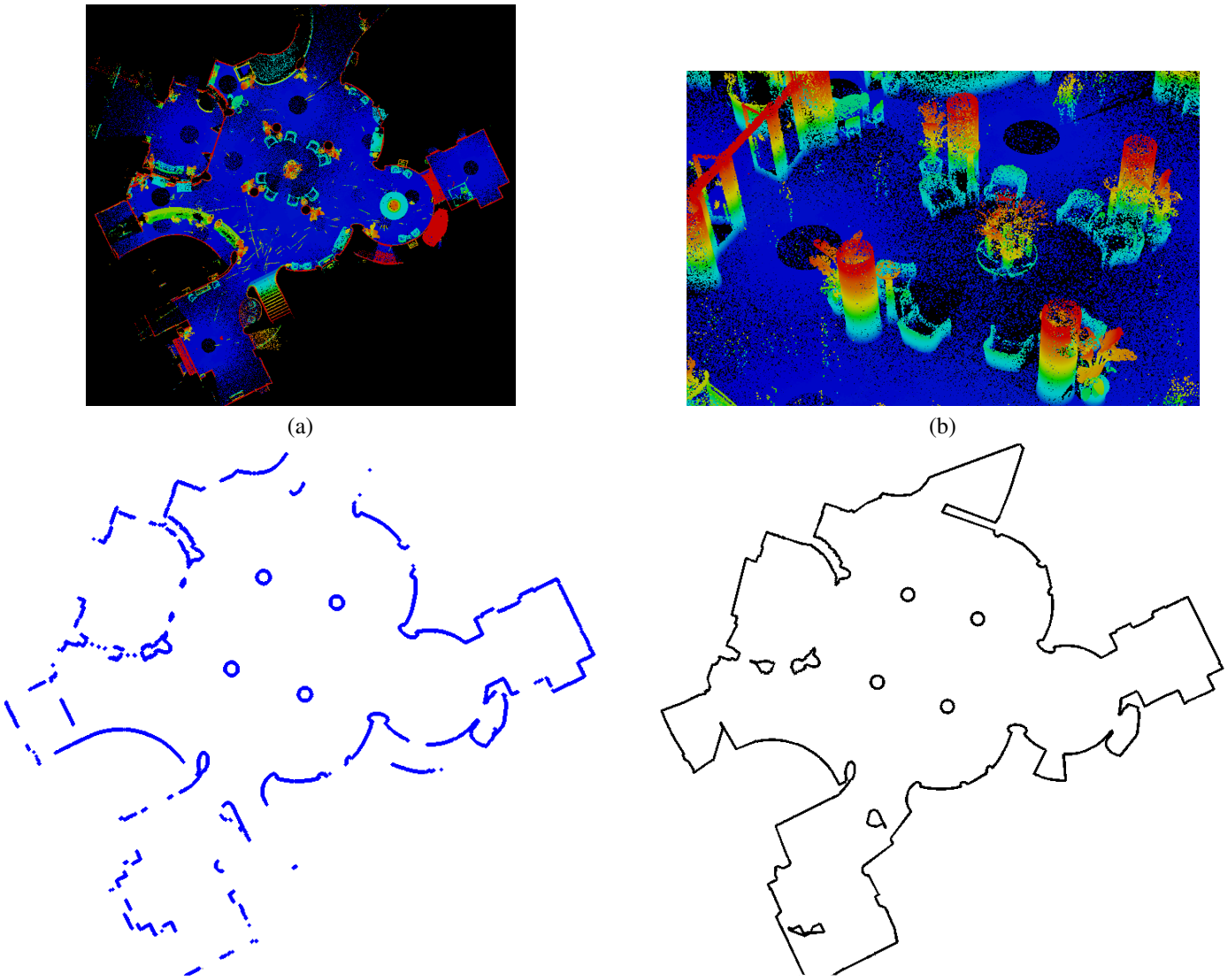


Figure 11. Hallway of hotel, captured with mobile scanner. (a) The full floor plan; (b) a close-up of a hallway intersection. The point cloud (top) was converted into wall sampling locations (middle) and boundary edges were fit to these samples (bottom).

shown in Figure 3a. Eigencrust performs poorly with such artifacts, since it often classifies the space in between the two instances of the wall as one label, and the spaces to either side of the wall as the other label. Ideally, an algorithm would classify both the in-between space as well as one side as "outside", and the other side as "inside." Performing the pre-processing steps as detailed in Section 3.2 can help mitigate these errors, but a more fundamental stategy for detecting and correcting



(a) (b) (c) (d)

Figure 12. Hotel lobby, captured with a static scanner. (a) The captured point-cloud; (b) a close-up of the center columns; (c) the wall samples extracted from this point cloud; (d) the set of wall boundary edges.

these instances of misalignment may prove fruitful as future work.

Further techniques could be used to increase the accuracy of our system. Most scanning systems capture optical imagery along with LiDAR data, but we currently do not use any camera information when forming estimates of building geometry. Accuracy could also be improved by coordinating the triangulations of each story in multi-story models. Currently the layout of each story is independent and this coordination could only be used if there is minimal registration error between building stories.

References

[1] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. *Proceedings of the Sixth Symposium on Solid Modeling*, pages 249–260, 2001. 7

[2] J. Doyle and R. L. Rivest. Linear expected time of a simple union-find algorithm. *Information Processing Letters*, 5:146–148, November 1976. 9, 10

[3] S. El-Hakim, L. Gonzo, F. Voltolini, S. Girardi, A. Rizzi, F. Remondino, and E. Whiting. Detailed 3d modeling of castles. *International Journal of Architectural Computing*, 5(2):200–220, June 2007. 2

[4] S. El-Hakim, E. Whiting, L. Gonzo, and S. Girardi. 3-d reconstruction of complex architectures from multiple data. *3D Virtual Reconstruction and Visualization of Complex Architectures*, August 2005. 2

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. 4, 10

[6] P. Galle and N. Kollegium. An algorithm for exhaustive generation of building floor plans. *Communications of the ACM*, 24(12):813–825, December 1981. 3

[7] M. Johnston and A. Zakhor. Estimating building floor-plans from exterior using laser scanners. *SPIE Electronic Imaging Conference, 3D Image Capture and Applications*, January 2008. 3

[8] I. T. Jolliffe. *Principal Components Analysis, Second Edition*. Springer, 1986. 5

[9] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien. Spectral surface reconstruction from noisy point clouds. *Symposium on Geometry Processing 2004*, pages 11–21, July 2004. 3, 6, 7, 8, 9

[10] R. Lewis and C. Sequin. Generation of 3d building models from 2d architectural plans. *Computer-Aided Design*, 30(10):765–779, September 1998. 2

[11] A. Nuchter, H. Surmann, and J. Hertzberg. Automatic model refinement for 3d reconstruction with mobile robots. *3-D Digital Imaging and Modeling*, pages 294–401, October 2003. 3

[12] B. Okorn, X. Xiong, B. Akinci, and D. Huber. Toward automated modeling of floor plans. *3DPVT*, 2009. 3

[13] S. Or, K. H. Wong, Y. Yu, and M. M. Chang. Highly automatic approach to architectural floorplan image understanding and model generation. *Pattern Recognition*, November 2005. 2

[14] F. Rottensteiner and C. Briese. A new method for building extraction in urban areas from high-resolution lidar data. *ISPRS*, 2002. 3

[15] P. Tang, D. Huber, B. Akinci, R. Lipman, and A. Lytle. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction*, 19(7):829–843, November 2010. 2

[16] C. Weiss and A. Zell. Automatic generation of indoor vr-models by a mobile robot with a laser range finder and a color camera. *Autonome Mobile Systeme*, (3):107–113, December 2005. 3

[17] A. Zakhor and C. Frueh. Automatic 3d modeling of cities with multimodal air and ground sensors. *Multimodal Surveillance; Sensors; Algorithms and Systems*, pages 339–362, 2007. 3