Interactive Acquisition of Residential Floor Plans

Young Min Kim, Jennifer Dolson, Mike Sokolsky, Vladlen Koltun, Sebastian Thrun
Stanford University
{ymkim, jldolson, mvs, vladlen, thrun}@stanford.edu

***Abstract*— We present a hand-held system for real-time, interactive acquisition of residential floor plans. The system integrates a commodity range camera, a micro-projector, and a button interface for user input and allows the user to freely move through a building to capture its important architectural elements. The system uses the Manhattan world assumption, which posits that wall layouts are rectilinear. This assumption allows generating floor plans in real time, enabling the operator to interactively guide the reconstruction process and to resolve structural ambiguities and errors during the acquisition. The interactive component aids users with no architectural training in acquiring wall layouts for their residences. We show a number of residential floor plans reconstructed with the system.**

I. Introduction

Acquiring an accurate floor plan of a home is a challenging task, yet it is a requirement in many situations that involve remodeling or selling a property. Original blueprints are often hard to find, especially for older residences. In practice, contractors and interior designers use point-to-point laser measurement devices to acquire a set of distance measurements. Based on these measurements, an expert creates a floor plan that respects the measurements and represents the layout of the residence.

In this paper, we present a hand-held system for indoor architectural reconstruction. The system eliminates the manual post-processing necessary for reconstructing the layout of walls in a residence. Instead, an operator with no architectural expertise can interactively guide the reconstruction process by moving freely through an interior until all walls have been observed by the system.

Our system is composed of a laptop connected to a commodity range sensor, a lightweight optical projector, and an input button interface (Figure 1, left). The real-time depth sensor is the main input modality. We use the Microsoft Kinect, a lightweight commodity device that outputs VGA-resolution range and color images at video rates. The data is processed in real time to create the floor plan by focusing on flat surfaces and ignoring clutter. The generated floor plan can be used directly for remodeling or real-estate applications, or to produce a 3D model of the interior for applications in virtual environments. In Section V, we demonstrate a number of residential wall layouts reconstructed with our system.

The attached projector is initially calibrated to have an overlapping field of view with the same image center as the depth sensor and projects the reconstruction status onto the surface being scanned. Under normal lighting, the projector does not provide sophisticated rendering. Rather, projection allows the user to visualize the reconstruction process. Then, the user can detect reconstruction errors that arise due to deficiencies in the data capture path, and can complete missing data in response. The user can also note which walls have been included in the model and easily resolve ambiguities with a simple input device.

II. Related Work

A number of approaches have been proposed for indoor reconstruction in computer graphics, computer vision, and robotics. Real-time indoor reconstruction has been recently explored either with a depth sensor [1] or an optical camera [2]. The key to real-time performance is the fast registration of successive frames. Similar to [1], we fuse both color and depth information to register frames. Furthermore, our approach extends real-time acquisition and reconstruction, by allowing the operator to visualize the current reconstruction status without consulting a computer screen. By making the feedback loop immediate, the operator can resolve failures and ambiguities while the acquisition session is in progress.

Previous approaches are also limited to a dense 3-D reconstruction (registration of point cloud data) with no higher-level information, which is memory intensive. A few exceptions include [3], which detects high level features (lines and planes) to reduce complexity and noise. The high level structures, however, do not necessarily correspond to actual meaningful structure. In contrast, our system identifies and focuses on significant architectural elements using the Manhattan world assumption, which is based on the observation that many indoor scenes are largely rectilinear [4]. This assumption has been widely used for indoor scene reconstruction from images to overcome the inherent limitations of image data [5][6]. The stereo method only reconstructs locations of image feature points, and the Manhattan world assumption successfully fills the area between the sparse feature points during a post-processing step. Similarly, our system differentiates between architectural features and miscellaneous objects in the space, produces a clean architectural floor plan, and simplifies the representation of the environment. Even with the Manhattan world assumption, however, the system still cannot fully resolve ambiguities introduced by large furniture items and irregular features in the space without user input. This interactive capability relies on the system's ability to integrate new input into a global map of the space in real time.

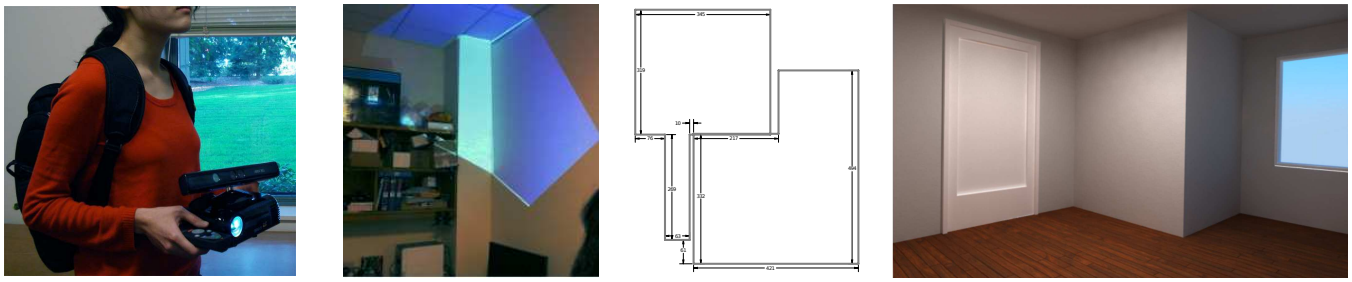

Fig. 1. Our hand-held system is composed of a projector, a Microsoft Kinect sensor, and an input button (left). The system uses augmented reality feedback (middle left) to project the status of the current model onto the environment and to enable real-time acquisition of residential wall layouts (middle right). The floor plan (middle right) and visualization (right) were generated using data captured by our system.

Simplifying the representation also reduces the computational burden of processing the map. Registration of successive point clouds results in an accumulation of errors, especially for a large environment, and requires a global optimization step in order to build a consistent map. This is similar to reconstruction tasks encountered in robotic mapping and is usually solved by bundle adjustment, a costly off-line process [7][8]. Employing the Manhattan world assumption simplifies the map construction to a one-dimensional, closed-form problem.

The augmented reality component of our system is inspired by the SixthSense project [9]. Instead of simply augmenting a user's view of the world, however, our projected output serves to guide an interactive reconstruction process. Directing the user in this way is similar to re-photography [10], where a user is guided to capture a photograph from the same viewpoint as in a previous photograph. By using a micro-projector as the output modality, our system allows the operator to focus on interacting with the environment.

III. System Overview and Usage

The data acquisition process is initiated by pointing the sensor to a corner, where three mutually orthogonal planes meet. This defines the Manhattan-world coordinate system. The attached projector will indicate successful initialization by overlaying blue-colored planes with white edges onto the scene (Figure 2 (a)). After the initialization, the user scans each room individually as he or she loops around it holding the device. If the movement is too fast or if there are not enough features, a red projection guides the user to recover the position of the device (Figure 2 (b)).

The system extracts flat surfaces that align with the Manhattan coordinate system and creates complete rectilinear polygons, even when connectivity between planes is occluded. Sometimes, the user might not want some of the extracted planes (parts of furniture or open doors) to be included in the model even if they satisfy the Manhattan-world constraint. When the user clicks the input button (left click), the extracted wall toggles between inclusion (indicated as blue) and exclusion (indicated as grey) to the model (Figure 2 (c)). As the user finishes scanning a room, he or she walks toward another room. A new rectilinear polygon is initiated by a right click. Another rectilinear polygon is similarly created by including the selected planes, and the room is correctly positioned into the global coordinate system. The model is updated in real time and stored in either a CAD format or a 3-D mesh format that can be loaded into most 3-D modeling software.

IV. Data Acquisition Process

At each time step t, the sensor produces a new frame of data, $\mathbf{F}^t = \{\mathbf{X}^t, \mathbf{I}^t, \mathbf{P}^t, \mathcal{T}^t\}$. The sensor output is composed of a range image $\mathbf{X}^t$ (a 2-D array of depth measurements) and a color image $\mathbf{I}^t$. During the acquisition process, we represent the relationship between the planes in global map $\mathbf{M}^t$ and the measurement in the current frame $\mathbf{X}^t$ as $\mathbf{P}^t$, a 2-D array of plane labels for each pixel. $\mathcal{T}^t$ represents the transformation from the frame $\mathbf{F}^t$, which is the measurement relative to the current sensor position, to the global coordinate system, which is where the map $\mathbf{M}^t$ is defined. Throughout the data capture session, the system maintains the global map $\mathbf{M}^t$, and the two most recent frames, $\mathbf{F}^{t-1}$ and $\mathbf{F}^t$. Additionally, the frame with the last observed corner $\mathbf{F}^c$ is stored to recover the sensor position when lost. Instead of storing information from all frames, we keep the total computational and memory requirements minimal by incrementally updating the global map only with components that need to be added to the final model.

The map $\mathbf{M}^t$ is composed of loops of axis-parallel planes $\mathbf{L}_r^t$. Each room has its own loop of planes. Each plane has its axis label (x, y, or z) and the offset value (e.g. $x = x_0$), as well as its left or right plane if the connectivity is observed. A plane can be selected or ignored based on user input. The selected planes are extracted from $\mathbf{L}_r^t$ as the loop of the room $\mathbf{R}_r^t$, which can be converted into the floor plan as a 2-D rectilinear polygon. Both $\mathbf{L}_r^t$ and $\mathbf{R}_r^t$ are constrained to have alternating axis labels (x and y). For the z direction (vertical direction), we are only keeping the ceiling and the floor. We also keep the sequence of observation ($\mathcal{S}^x$, $\mathcal{S}^y$, and $\mathcal{S}^z$) of offset values for each axis direction, and we store the measured distance and the uncertainty of the measurement between planes.

The overall reconstruction process is summarized in Figure 2. As mentioned in Sec. III, the process is initiated by extracting three mutually orthogonal planes, when a user points the system to one of the corners. To detect planes

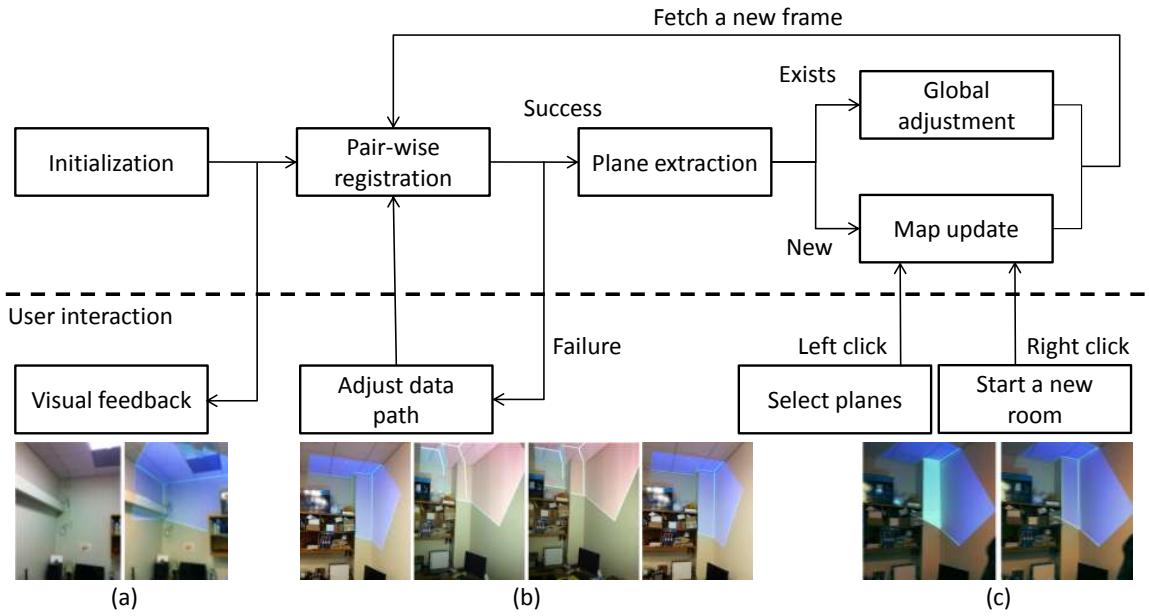

Fig. 2. System overview and usage (Section III). When an acquisition session is initiated by observing a corner, the user is notified by a blue projection (a). After the initialization, the system updates the camera pose by registering consecutive frames. If a registration failure occurs, the user is notified by a red projection and is required to adjust the data capture path (b). Otherwise, the updated camera configuration is used to detect planes that satisfy the Manhattan-world constraint in the environment and to integrate them into the global map. The user interacts with the system by selecting planes in the space (c). When the acquisition session is completed, the acquired map is used to construct a floor plan consisting of user-selected planes.

in the range data, we fit plane equations to groups of range points and their corresponding normals using the RANSAC algorithm [11]: we first randomly sample a few points, then fit a plane equation to them. Then, we test the detected plane by counting the number of points that can be explained by the plane equation. After convergence, the detected plane is classified as valid only if the detected points constitute a large, connected portion of the depth information within the frame. If there are three planes detected and they are orthogonal to each other, we assign the x, y and z axes to be the normal directions of these three planes, which form the right-handed coordinate system for our Manhattan world. Now the map $\mathbf{M}^t$ has two planes (ignoring the floor or ceiling), and the transformation $\mathcal{T}^t$ between $\mathbf{M}^t$ and $\mathbf{F}^t$ is also found.

A new measurement $\mathbf{F}^t$ is registered with the previous frame $\mathbf{F}^{t-1}$ by aligning depth and color features (Sec. IV-A). This registration is used to update $\mathcal{T}^{t-1}$ to a new transformation $\mathcal{T}^t$. Then, we extract planes that satisfy the Manhattan world from $\mathcal{T}^t(\mathbf{F}^t)$ (Sec. IV-B). If the extracted planes already exist in $\mathbf{L}_r^t$, the current measurement is compared with the global map and the registration is refined (Sec. IV-C). If there is a new plane extracted, or if there is user input to specify the map structure, the map is updated accordingly (Sec. IV-D).

A. Pair-wise registration

To propagate information from previous frames and to detect new planes in the scene, each incoming frame must be registered with respect to the global coordinate system. To start this process, we find the relative registration between

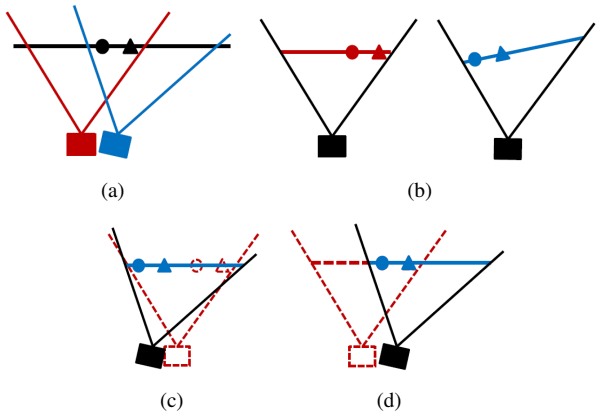

Fig. 3. (a) Flat wall features (depicted as the triangle and circle) are observed from two different locations. Diagram (b) shows both observations with respect to the camera coordinate system. Without features, using projection-based ICP can lead to registration errors in the image-plane direction (c), while utilizing the features will provide better registration (d).

the two most recent frames, $\mathbf{F}^{t-1}$ and $\mathbf{F}^t$. In using both the depth point clouds ($\mathbf{X}^{t-1}$, $\mathbf{X}^t$) and optical images ($\mathbf{I}^{t-1}$, $\mathbf{I}^t$), the frames can efficiently be registered in real time (about 20 fps).

Given two sets of point clouds, $\mathbf{X}^{t-1} = \{x_i^{t-1}\}_{i=1}^N$ and $\mathbf{X}^t = \{x_i^t\}_{i=1}^N$, and the transformation for the previous point cloud $\mathcal{T}^{t-1}$, the correct rigid transformation $\mathcal{T}^t$ will minimize the error between correspondences in the two sets:

$$\min_{y_i, \mathcal{T}^t} \sum_i \|w_i(\mathcal{T}^{t-1}(x_i^{t-1}) - \mathcal{T}^t(y_i^t))\|^2 \qquad (1)$$

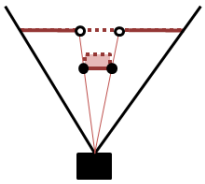

Fig. 4. Silhouette points. There are two different types of depth discontinuity: the boundaries of a shadow made on the background by a foreground object (empty circles), and the boundaries of a foreground object (filled circles). The meaningful depth features are the foreground points, which are the silhouette points used for our registration pipeline.

$y_i^t \in \mathbf{X}^t$ is the corresponding point for $x_i^{t-1} \in \mathbf{X}^{t-1}$. Once the correspondence is known, minimizing Eq. (1) becomes a closed-form solution [12]. Traditionally, the correspondence was found by searching for the closest point, which is computationally expensive. Real-time registration methods reduce the cost by projecting the 3-D points onto a 2-D image plane and assigning correspondences to points that project onto the same pixel locations [13]. However, projection will only reduce the distance in the ray direction; the offset parallel to the image plane cannot be adjusted. This phenomenon can result in the algorithm not compensating for the translation parallel to the plane and therefore shrinking the size of the room (Figure 3).

Our pair-wise registration is similar to [13], but it compensates for the displacement parallel to the image plane using image features and silhouette points. Intuitively, we use homography to compensate errors parallel to the plane if the structure can be approximated into a plane, and silhouette points are used to compensate remaining errors when the features were not planar. We first compute the optical flow between color images $\mathbf{I}^t$ and $\mathbf{I}^{t-1}$ and find a homography (a transformation found from tracked image features between them), which represents the displacement parallel to the image plane. Then, we use the homography to compute dense correspondences between the two frames. From the second iteration, the correspondence is found by projecting individual points onto the image plane.

Additionally, we modify the correspondence for silhouette points (points of depth discontinuity in the foreground, shown in Figure 4). For silhouette points in $\mathbf{X}^{t-1}$, we find the closest silhouette points in $\mathbf{X}^t$ within a small search window from the original corresponding location. If the matching silhouette point exists, the correspondence is weighted more. (We used $w_i = 100$ for silhouette points and $w_i = 1$ for non-silhouette points.) Then, the registration between the two frames for the current iteration can be given as a closed-form solution. The process iterates until it converges.

1) Registration Failure: The real-time registration is a crucial part of our algorithm for accurate reconstruction. Even with the hybrid approach using both color and depth features, the registration can fail, and it is important to detect the failure immediately and to recover the position of the sensor. The registration failure is detected either (1) if the pair-wise registration does not converge or (2) if there were not enough color and depth features. The first case can be easily detected as we run the algorithm. The second case is detected if the optical flow did not find homography (i.e. there is a lack of color feature) and there were not enough matched silhouette points (i.e. there is a lack of depth feature).

In cases of registration failure, the projected image turns red, indicating that the user should return the system's viewpoint to the most recently observed corner. This movement usually takes only a small amount of back-tracking, as the failure is detected within milliseconds of leaving the previous successfully registered area. Similar to the initialization step, the system extracts planes from $\mathbf{X}^t$ using RANSAC and matches the planes with the desired corner. We show the process of overcoming a registration failure in Figure 2 (b). The user then deliberately moves the sensor along the path with richer features or steps back to have a wider field of view.

B. Plane extraction

Based on the transformation $\mathcal{T}^t$, we extract axes-aligned planes and associated edges. The planes and detected features will provide higher-level information that relates the raw point cloud $\mathbf{X}^t$ to the global map $\mathbf{M}^t$. Because we are only considering the planes that satisfy the Manhattan-world coordinate system, we can simplify the plane detection procedure.

The planes that were visible from the previous frame can be easily found by using the correspondence. From the pair-wise registration (Sec. IV-A), we have the point-wise correspondence between the previous frame and the current frame. The plane label $\mathbf{P}^{t-1}$ from the previous frame is updated simply by being copied over to the corresponding location. Then, we refine $\mathbf{P}^t$ by alternating between fitting points and fitting parameters.

A new plane can be found by projecting remaining points for the x, y, and z axes. For each axis direction, we build a histogram with the bin size $20cm$ and test the plane equation for populated bins. Compared to the RANSAC procedure for initialization, the Manhattan world assumption reduces the number of degrees of freedom from three to one, making plane extraction more efficient.

For extracted planes, the boundary edges are also extracted; we detect groups of boundary points that can be explained by an axis-parallel line segment. We also keep the information of relative positions for extracted planes (left/right). As long as the sensor is not flipped upside-down, this provides an important cue to build a room with the correct topology, even when the connectivity between neighboring planes was not observed.

1) Data association: After the planes are extracted, the data association process finds the link between the global map $\mathbf{M}^t$ and the extracted planes to be $\mathbf{P}^t$, a 2-D array of plane labels for each pixel. The plane labels that existed from previous frame were automatically found while extracting the plane by copying over the plane labels using correspondences.

The plane labels for the newly detected plane can be found by comparing $\mathcal{T}^t(\mathbf{F}^t)$ and $\mathbf{M}^t$. In addition to the plane equation, the relative position with respect to other observed planes are used to label the one. If the plane was not observed before, a new plane will be added into $\mathbf{L}_r^t$ based on the left-right information. The adjacent walls should have alternating axis directions (a $x = x_i$ wall should be connected to a $y = y_j$ wall). If the two observed walls have the same axis direction, we add the unobserved wall between them on the boundary of the planes to form a complete loop.

After the data association step, we update the sequence of observation $\mathcal{S}$. The planes that have been assigned as previously observed are used for global adjustment (Sec. IV-C). If a new plane was observed, the room $\mathbf{R}_r^t$ will be updated accordingly (Sec. IV-D).

C. Global adjustment

Due to noise in the point cloud, frame-to-frame registration is not perfect, and error accumulates over time. This is a common problem in pose estimation. Large-scale localization approaches use bundle adjustment to contain error accumulation [7], [8]. Enforcing this global constraint involves detecting *landmark* objects, or stationary objects observed at different times during a sequence of measurements. Usually this global adjustment becomes an optimization problem in many dimensions. The problem is formulated by constraining the landmarks to predefined global locations, and by solving an energy function that encodes noise in a pose estimation of both sensor and landmark locations. The Manhattan world assumption allows us to reduce the error accumulation efficiently in real time by refining our registration estimate and by optimizing the global map.

1) Refining the registration: After data association, we perform a second round of registration with respect to the global map $\mathbf{M}^t$ to reduce the error accumulation in $\mathcal{T}^t$ by incremental, pair-wise registration. The extracted planes $\mathbf{P}^t$, if already observed by the system, have been assigned to the planes in $\mathbf{M}^t$ that have associated plane equations. For example, suppose a point $\mathcal{T}^t(\mathbf{x}_{u,v}) = (x, y, z)$ has a plane label $\mathbf{P}^t(u, v) = p_k$ (assigned to plane k). If plane k has normal parallel to the x axis, the plane equation in the global map $\mathbf{M}^t$ can be written as $x = x_0$ ($x_0 \in \mathbb{R}$). Then, the registration should be refined to minimize $\|x - x_0\|^2$. This can be found by defining the corresponding point for $\mathbf{x}_{u,v}$ as (x_0, y, z). The corresponding points are likewise assigned for every point with plane assignment in $\mathbf{P}^t$. Given the correspondence, we can refine the registration between the current frame $\mathbf{F}^t$ and the global map $\mathbf{M}^t$. This second round of registration reduces the error in the axis direction. In our example, the refinement is active while the plane $x = x_0$ is visible and reduces the uncertainty in the x direction with respect to the global map. The error in the x direction is not accumulated during the interval.

2) Optimizing the map: As error accumulates, the reconstructed map $\mathbf{M}^t$ may also require global adjustment in each axis direction. The Manhattan world assumption simplifies this global optimization into two separate, one-dimensional problems (we are ignoring the z direction for now, but the idea can be extended to a 3-D case).

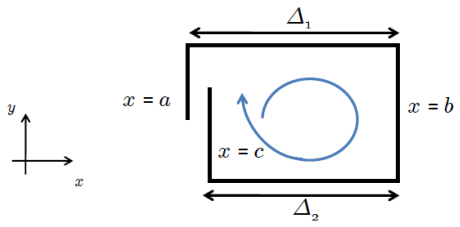

Fig. 5. As errors accumulate in $\mathcal{T}^t$ and in measurements, the map $\mathbf{M}^t$ becomes inconsistent. By comparing previous and recent measurements, the system can correct for inconsistency and update the value of c such that $c = a$.

Figure 5 shows a simple example in the x-axis direction. Suppose an overhead view of a rectangular room. There should be two walls whose normals are parallel to the x-axis. The sensor detects the first wall ($x = a$), sweeps around the room, observes another wall ($x = b$), and returns to the previously observed wall. Because of error accumulation, parts of the same wall have two different offset values ($x = a$ and $x = c$), but by observing the left-right relationship between walls, the system infers that the two walls are indeed the same wall.

To optimize the offset values, we track the sequence of observations $\mathcal{S}^x = \{a, b, c\}$ and the variances at the point of observation for each wall, as well as the constraints represented by the pair of the same offset values $\mathcal{C}^x = \{(c_{11}, c_{12}) = (a, c)\}$. We introduce two random variables, Δ_1 and Δ_2, to constrain the global map optimization. Δ_1 is a random variable with mean $m_1 = b - a$ and variance σ_1^2 that represents the error between the moment when the sensor observed the $x = a$ wall and the moment it observed the $x = b$ wall. Likewise, a random variable Δ_2 represents the error with mean $m_2 = c - b$ and variance σ_2^2.

Whenever a new constraint is added, or when the system observes a plane that was previously observed, the global adjustment routine is triggered. This is usually when the user finished scanning a room by looping around it and returning to the first wall measured. By confining the axis direction, the global adjustment becomes a one-dimensional quadratic equation:

$$\begin{aligned} &\min_{\mathcal{S}^x} \textstyle\sum_i \left(\frac{\|\Delta_i - m_i\|^2}{\sigma_i^2} \right) \\ &\text{s. t. } c_{j1} = c_{j2}, \ \ \forall (c_{j1}, c_{j2}) \in \mathcal{C}^x. \end{aligned} \tag{2}$$

D. Map update

Our algorithm ignores most irrelevant features using the Manhattan-world constraint. However, the system cannot distinguish architectural components from other axis-aligned objects using Manhattan world assumption. For example, furniture, open doors, parts of other rooms that might be visible, or reflections from mirrors may be detected as axis-aligned planes. We solve the challenging cases by allowing the user to manually specify the planes that he or she would like to include in the final model. This manual specification

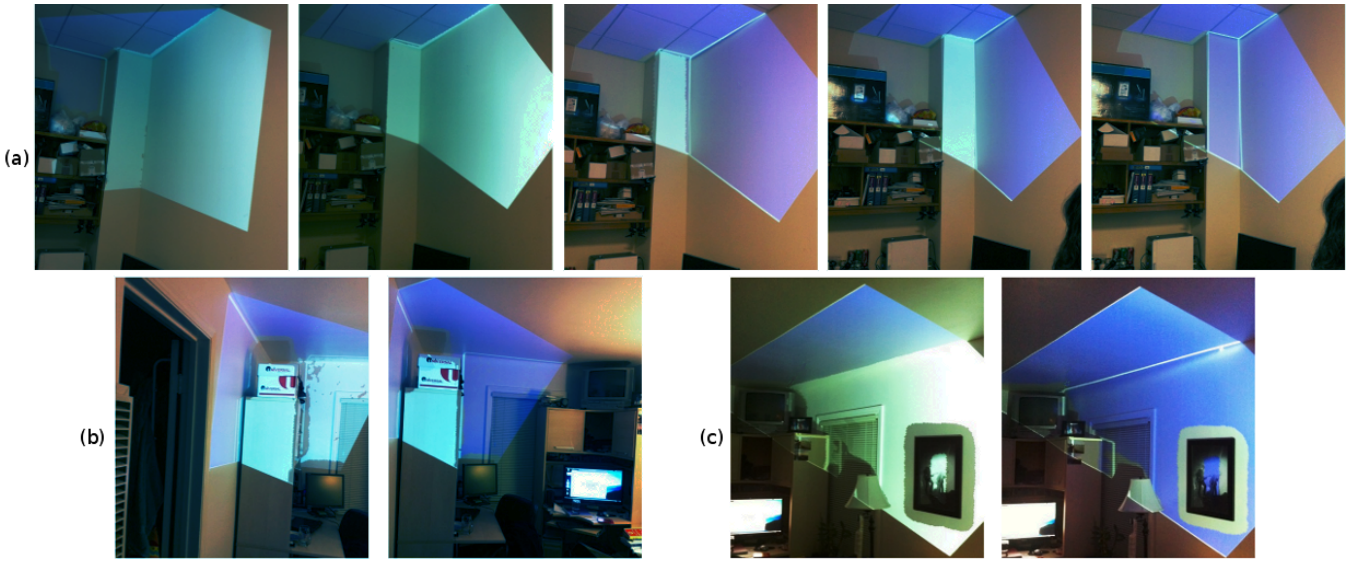

Fig. 6. Selection. In sequence (a), the user is observing two new planes in the scene (colored white) and one currently included plane (colored blue). The user selects one of the new planes by pointing at it and clicking. Then, the second new plane is added. All planes are blue in the final frame, confirming that all planes have been successfully selected. Sequence (b) shows a configuration where the user has decided not to include the large cabinet. Sequence (c) shows successful selection of the ceiling and the wall despite clutter.

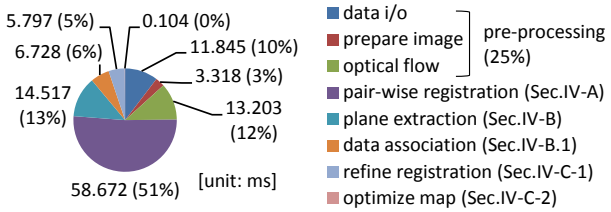

Fig. 7. average computational time for each step of the system.

consists of simply clicking the input button during scanning when pointing at a plane, as shown in Figure 6. If the user enters a new room, a right click of the button indicates the user wishes to include this room and to optimize it individually. The system creates a new loop of planes and any newly observed planes are added to the loop.

Whenever a new plane is added to $\mathbf{L}_r^t$ or there is user input to specify the room structure, the map update routine extracts a 2-D rectilinear polygon $\mathbf{R}_r^t$ from $\mathbf{L}_r^t$ with the help of user input. We start by adding all selected planes into $\mathbf{R}_r^t$ as well as whichever unselected planes in $\mathbf{L}_r^t$ are necessary to have alternating axis direction. The planes with observed boundary edges have priority to be added.

V. Evaluation

The current practice in architecture and real estate is to use a point-to-point laser device to measure distances between pairs of parallel planes. Making such measurements requires a clear, level line of sight between two planes, which may be time-consuming to find due to furniture, windows, and other obstructions. After making all the distance measurements, a user is required to manually draw a floor plan that respects the measurements. Roughly 10-20 minutes was needed to take the distance measurements in each apartment. Using our system, the data acquisition process took approximately 2-5 minutes per home to initiate, run, and generate the full floor plan. Table I summarizes the timing data for each data set. The average frame rate is 7.5 frames per second running on an Intel 2.50GHz Dual Core laptop. Figure 7 depicts the average computational time for each step of the algorithm. The pair-wise registration routine (Sec.IV-A) contributes more than half of the computational time followed by the pre-processing step of fetching a new frame and calculating optical flow (25%).

In Figure 8, we visually compare the reconstructed floor plans. The floor plans in blue are reconstructed using point-to-point laser measurements, and the floor plans in red are reconstructed by our system. For each home, the topology of the reconstructed walls agrees with the manually-constructed floor plan. In all cases the detection and labeling of planar surfaces by our algorithm enabled the user to add or remove these surfaces from the model in real time, allowing the final model to be constructed using only the important architectural elements from the scene.

The overlaid floor plans in Figure 8(c) show that that the relative placement of the rooms may be misaligned. This is because our global adjustment routine optimizes rooms individually, thus error can accumulate in transitions between rooms. The algorithm could be extended to enforce global constraints on the relative placement of rooms, such as maintaining a certain wall thickness and/or aligning the outer-most walls, but such global constraints may induce other errors.

Table I contains a quantitative comparison of the errors. The reported depth resolution of the sensor is 0.01m at 2m, and for each model we have an average of 0.075m error per wall. The relative error stays in the range of 2-5%, which shows that the accumulation of small registration error

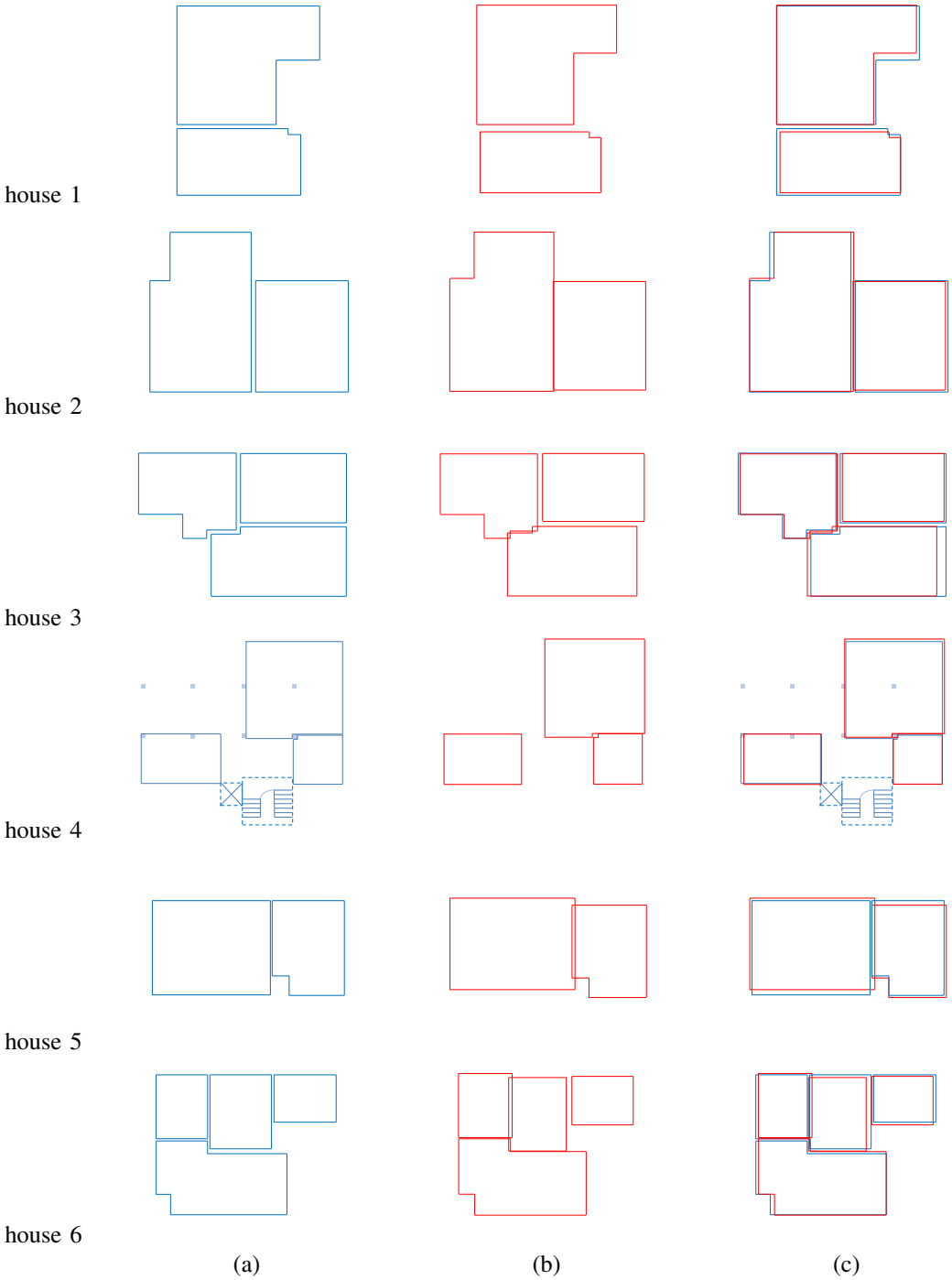

Fig. 8. (a) Manually constructed floor plans generated from point-to-point laser measurements, (b) floor plans acquired with our system, and (c) overlay. For house 4, some parts (pillars in large open space, stairs, and an elevator) are ignored by the user. The system still uses the measurements from those parts and other objects to correctly understand the relative positions of the rooms.

continues to increase as more frames are processed.

Fundamentally, the limitations of our method reflect those of the Kinect sensor, namely, the processing power of the laptop and the assumptions made in our approach. As the accuracy of depth data is worse than visual features, our approach exhibits larger errors compared to visual SLAM. Some of the uncertainty can be reduced by adapting approaches from well-explored visual SLAM literature. Still, we are limited when we cannot detect meaningful features. The Kinect sensor's reported measurement range is between 1.2 and 3.5m from an object; outside that range, data is noisy or unavailable. As a consequence, data in narrow hallways or large atriums is difficult to collect.

Another source of potential error is a user outpacing the operating rate of approximately 7.5 fps. This frame rate already allows for a reasonable data capture pace, but with

data set	no. of frames	run time	fps	average error	
				m	%
1	1465	2m 56s	8.32	0.115	4.14
2	1009	1m 57s	8.66	0.064	1.90
3	2830	5m 19s	8.88	0.053	2.40
4	1129	2m 39s	7.08	0.088	2.34
5	1533	3m 52s	6.59	0.178	3.52
6	2811	7m 4s	6.65	0.096	3.10
ave.	1795	3m 57s	7.54	0.075	2.86

TABLE I

ACCURACY COMPARISON BETWEEN FLOOR PLANS RECONSTRUCTED BY OUR SYSTEM, AND MANUALLY CONSTRUCTED FLOOR PLANS GENERATED FROM POINT-TO-POINT LASER MEASUREMENTS.

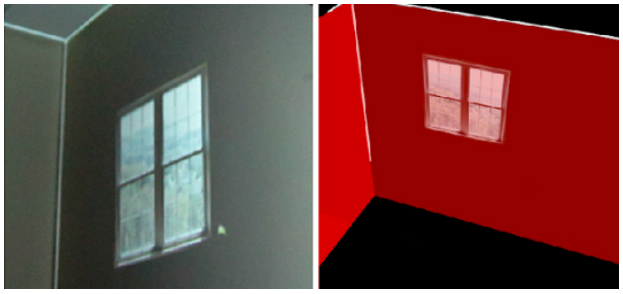

Fig. 9. The system, having detected the planes in the scene, also allows the user to interact directly with the physical world. Here the user adds a window to the room by dragging a cursor across the wall (left). This motion updates the internal model of the world (right).

more processing power, the pace of the system could always be guaranteed to exceed normal human motion.

VI. Conclusion and Future Work

We have presented an interactive system that allows a user to capture accurate architectural information and to automatically generate a floor plan. Leveraging the Manhattan world assumption, we create a representation that is tractable in real time while ignoring clutter. The current status of the reconstruction is projected on the scanned environment to enable the user to provide high-level feedback to the system. This feedback helps overcome ambiguous situations and allows the user to interactively specify the important planes that should be included in the model.

More broadly, our interactive system can be extended to other applications in indoor environments. For example, a user could visualize modifications to the space as in Figure 9, where we show a user clicking and dragging a cursor across a plane to "add" a window. This example illustrates the range of possible uses of our system.

References

[1] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox., "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *ISER*, 2010.

[2] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *CVPR*, 2010.

[3] A. P. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas, "Discovering higher level structure in visual slam," *IEEE Transactions on Robotics*, vol. 24, pp. 980–990, October 2008.

[4] J. M. Coughlan and A. L. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," in *ICCV*, pp. 941–947, 1999.

[5] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *ICCV*, pp. 80–87, 2009.

[6] C. A. Vanegas, D. G. Aliaga, and B. Benes, "Building reconstruction using manhattan-world grammars," in *CVPR*, pp. 358–365, 2010.

[7] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, Springer-Verlag, 2000.

[8] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium* (G. Lakemeyer and B. Nebel, eds.), Morgan Kaufmann, 2002.

[9] P. Mistry and P. Maes, "Sixthsense: a wearable gestural interface," in *SIGGRAPH ASIA Art Gallery & Emerging Technologies*, p. 85, 2009.

[10] S. Bae, A. Agarwala, and F. Durand, "Computational rephotography," *ACM Trans. Graph.*, vol. 29, no. 5, 2010.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[12] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Trans. PAMI*, vol. 14, pp. 239–256, 1992.

[13] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proc. 3DIM*, 2001.