Indoor Localization and Visualization Using a Human-Operated Backpack System

Timothy Liu, Matthew Carlberg, George Chen, Jacky Chen, John Kua, Avideh Zakhor
Video and Image Processing Lab, University of California, Berkeley
{timothy,carlberg,gchen,jackyc,jkua,avz}@eecs.berkeley.edu

***Abstract*—Automated 3D modeling of building interiors is useful in applications such as virtual reality and entertainment. Using a human-operated backpack system equipped with 2D laser scanners and inertial measurement units (IMU), we develop scan matching based algorithms to localize the backpack in complex indoor environments such as a T-shaped corridor intersection, a staircase, and two indoor hallways from two separate floors connected by a staircase. When building 3D textured models, we find that the localization resulting from scan matching is not pixel accurate, resulting in misalignment between successive images used for texturing. To address this, we propose an image based pose estimation algorithm to refine the results from our scan matching based localization. Finally, we use the localization results within an image based renderer to enable virtual walkthroughs of indoor environments using imagery from cameras on the same backpack. Our renderer uses a three-step process to determine which image to display, and a RANSAC framework to determine homographies to mosaic neighboring images with common SIFT features. In addition, our renderer uses plane-fitted models of the 3D point cloud resulting from the laser scans to detect occlusions. We characterize the performance of our image based renderer on an unstructured set of 2709 images obtained during a five minute backpack data acquisition for a T-shaped corridor intersection.**

I. Introduction

Three-dimensional modeling of indoor and outdoor environments has a variety of applications such as training and simulation for disaster management, virtual heritage conservation, and mapping of hazardous sites. Manual construction of these models can be time consuming, and as such, automated 3D site modeling has garnered much interest in recent years. Interior modeling in particular poses significant challenges, the primary one being indoor localization in the absence of GPS.

Localization has been studied by robotics and computer vision communities in the context of the simultaneous localization and mapping problem (SLAM) where a vehicle's location within an environment and a map of the environment are estimated simultaneously [1]. The vehicle is typically equipped with a combination of laser range scanners, cameras, and inertial measurement units (IMUs). Recent work in this area has shifted toward solving SLAM with six degrees of freedom (DOF) [2]–[4], namely position and orientation. SLAM approaches with laser scanners often use scan matching algorithms such as Iterative Closest Point (ICP) [5] to align scans from two poses in order to recover the transformation between the poses. Meanwhile, advances in visual odometry algorithms have led to camera based SLAM approaches [3], [6]. With a single camera, pose can be estimated only up to an unknown scale factor. This scale is generally determined using GPS waypoints, making it inapplicable to indoor environments unless objects of known size are placed in the scene. To resolve this scale ambiguity, stereo camera setups have gained popularity, as the extrinsic calibration between the cameras can be used to recover absolute translation parameters [3], [7].

Localizing a vehicle using scan matching and/or visual odometry and/or wheel odometry can result in significant drifts in navigation estimates over time. The error becomes apparent when the vehicle encounters a previously visited location, at which point it has traversed a loop. However, the estimated trajectory from localization algorithms may not form a loop. Such inconsistencies can be remedied by detecting when these loop closure events occur and solving optimization problems to close the loop [3], [8]–[10].

Previously, we developed a number of localization algorithms for a human-operated backpack system equipped with laser scanners and IMUs, and characterized their performance on a simple 30-meter hallway [11]. The motivation for using a backpack system rather than a wheeled robot or pushcart has been the ability to map complex environments such as staircases. In previous work, we manually detected the loops in the traversed path to enforce loop closure [11]. In this paper, we propose an automatic loop detection algorithm based on FAB-MAP [12] and keypoint matching [13]. We show that our localization algorithms perform just as accurately in more complex environments, such as staircases, as they do in simple flat hallway environments.

Though the localization errors resulting from these localization algorithms are quite low even in these complex environments, when the resulting recovered pose is used to texture map camera imagery onto the resulting 3D triangular mesh models, there is significant misalignment between successive images used to texture map neighboring triangles. This implies that the scan matching based localization algorithms are not pixel accurate. To cope with this problem, we propose an image based approach in which the pose from scan matching based localization is refined using camera imagery. The goal of this step is to ensure that images in the texture mapping process are sufficiently aligned so as to avoid visual artifacts.

We use backpack localization results to develop an image based renderer for virtual walkthroughs of indoor environments using imagery from cameras on the same backpack. Our renderer uses a three-step process to determine which

978-1-4244-5864-6/10$26.00 © IEEE

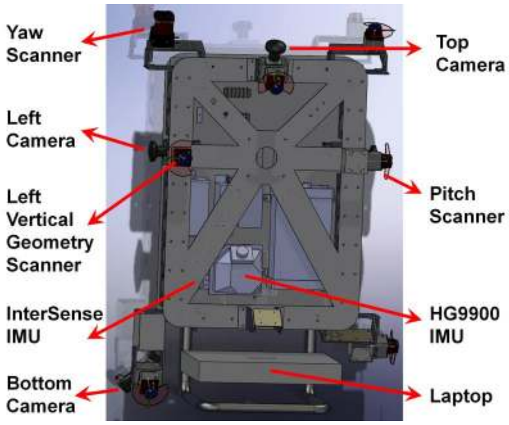


Fig. 1. CAD model of the backpack system.

image to display, and a RANSAC framework to determine homographies to mosaic neighboring images with common SIFT features. In addition, our renderer uses plane-fitted models of the 3D point cloud resulting from the laser scans to detect occlusions.

The outline of the paper is as follows. The architecture of our backpack system is described in Section II. Localization algorithms including loop detection and closure are included in Section III. In Section IV, we discuss our image based renderer, including image selection, mosaicing, and rendering. The conclusions are in Section V.

II. Architecture

We mount three 2D laser range scanners and two IMUs onto our backpack rig, which is carried by a human operator. Figure 1 shows the CAD model of our backpack system. The yaw scanner is a 40Hz Hokuyo UTM-30LX 2D laser scanner with a 30-meter range and a field of view of 270°. The pitch scanner and left vertical geometry scanner are 10Hz Hokuyo URG-04LX 2D laser scanners each with a 4-meter range and a field of view of 240°. These scanners are positioned orthogonal to each other. One IMU is a strap-down navigation-grade Honeywell HG9900 IMU, which combines three ring laser gyros with bias stability of less than 0.003°/hour and three precision accelerometers with bias of less than 0.245mm/sec^2. The HG9900 provides highly accurate measurements of all six degrees of freedom (DOF) at 200Hz and serves as our ground truth. The other IMU, an InterSense InertiaCube3, provides orientation parameters at the rate of 180Hz. As seen later, we localize the backpack over time using only the yaw scanner, the pitch scanner, and the InterSense IMU. In particular, we estimate the backpack's pose at a rate of 10Hz, the same rate as the pitch scanner. The left vertical geometry scanner is only used to build a 3D point cloud following localization.

We use a right-handed local coordinate system. With the backpack worn upright, x is forward, y is leftward, and z is upward. Referring to Figure 1, the yaw scanner scans the x-y plane, the pitch scanner scans the x-z plane, and the left vertical geometry scanner scans the y-z plane. Thus, the yaw scanner can resolve yaw rotations about the z axis and the pitch scanner about the y axis.

Assuming that the yaw scanner scans the same plane over time, we can apply scan matching on successive laser scans from the yaw scanner and integrate the translations and rotations obtained from scan matching to recover x, y, and yaw of the backpack over time. Likewise, assuming that pitch scanner scans the same plane over time, we can apply scan matching on successive laser scans from the pitch scanner to recover x, z, and pitch.

The assumption of scanning the same plane roughly holds for both the yaw and the pitch scanners. However, we have empirically found that coplanarity assumption remains more valid if the effective range of the yaw scanner is limited. In particular, points scanned that are closer to the yaw scanner appear to come from approximately the same plane between two successive scan times. However, points farther away from the yaw scanner can potentially come from two very different planes between two successive scan times, for example if between these times the backpack experiences a large pitch change. These scan points that clearly come from different planes between two scan times cannot be aligned by scan matching. Thus, for the experiments in this paper, we discard points farther than a certain threshold away from the yaw scanner. We have empirically found that limiting the yaw scanner range to 15 meters still allows for nearly all the yaw scanner's range data between two successive scan times to appear to roughly come from the same plane.

III. Localization

We use laser/IMU based localization algorithms from [11] to estimate the transformation between backpack poses at consecutive time steps. We compose these transformations to reconstruct the entire trajectory the backpack traverses. However, since each transformation is somewhat erroneous, the error in the computed trajectory can become large over time, resulting in loop closure errors. In Section III-A we propose an automatic loop closure detection method based on images collected by the backpack. Once loops are detected, we enforce loop closure using a nonlinear optimization technique in the Tree based netwORk Optimzer (TORO) [10] to reduce the overall localization error. In Section III-B, we apply the loop closure detection method from Section III-A in conjunction with TORO to experimentally characterize the performance of the laser/IMU based localization algorithms in [11] for complex environments such as staircases. In Section III-C, we combine an image based pose estimation technique with TORO to refine the laser/IMU localization results. As seen, the resulting 3D textured models from this localization approach are pixel accurate and as such, do not suffer from texture alignment issues prevalent in 3D models resulting from laser/IMU-only localization algorithms.

A. Loop Closure Detection

In this section, we describe an algorithm to automatically detect loop closures using camera imagery on the backpack.

Our algorithm is a modified version of Newman's FAB-MAP [12], a probabilistic approach of recognizing places via appearance.

Our modifications and extensions to the FAB-MAP algorithm are as follows. First, after building a vocabulary from training data and converting all scenes into words, we remove words that appear in all, one, or no scenes. This is because we calculate the co-occurrence of words for the Chow Liu tree [14] and words that appear in every image, only one image, or none of the images provide little information in distinguishing images. Second, location prior is left uniform, since performance is largely unaffected [12].

Computing the probability distribution of all images over all locations is considered one trial. We call a match between an image and a location an image pair, since a location originates from an image. This match occurs because the probability of it being a genuine loop closure is higher than a prespecified threshold; however, in practice it could be either a genuine loop closure or a false positive. In order to emphasize genuine loop closures and recognize false positives, we run 100 trials and record all image pairs with the number of times they appear in the trials. Figure 2 shows several image pairs with the most counts in dataset 1, which corresponds to a 5 minute walk in a T-shaped corridor with about 100 images. While in this example the two image pairs with the highest count from FAB-MAP correspond to genuine loop closures and the remaining ones do not, in general this is not the case. As such, some postprocessing is needed to detect genuine loop closures among the top ranked image pair candidates generated by FAB-MAP.

We now examine image pairs that have appeared the most and use keypoint matching [13] to determine whether they are correct matches. As shown in [13], correct and incorrect matches have different distributions of the ratio

$$\frac{d(\text{feature, nearest neighbor})}{d(\text{feature, }2^{\text{nd}}\text{ nearest neighbor})} \tag{1}$$

where $d(a,b)$ computes the Euclidean distance between a and b. Figure 3 shows the PDF of the above ratio for all the features in two image pairs of Figure 2 corresponding to a correct and an incorrect match. As seen, the PDF for genuine and incorrect matches are quite different. Unlike an incorrect match, for a genuine match a significant number and percentage of the features result in the ratio in Equation 1 being smaller than 0.6. Figure 4(a) shows the number of candidate image pairs across all 9 datasets as a function of the number of features satisfying the ratio in Equation 1; each dataset corresponds to a 5 to 10 minute data acquisition and consists of 100 images; there are a total of 13 genuine loop closure pairs for the 9 datasets. As seen, using the absolute number of features is not a reliable indicator of the correctness of a given image pair. By contrast, Figure 4(b) shows the same quantity as a function of the percentage of features satisfying the ratio in Equation 1. As seen, the percentage of features satisfying Equation 1 can be successfully used to distinguish between correct and incorrect candidate image pairs resulting from FAB-MAP for all 13 loop closures corresponding to 9 datasets.

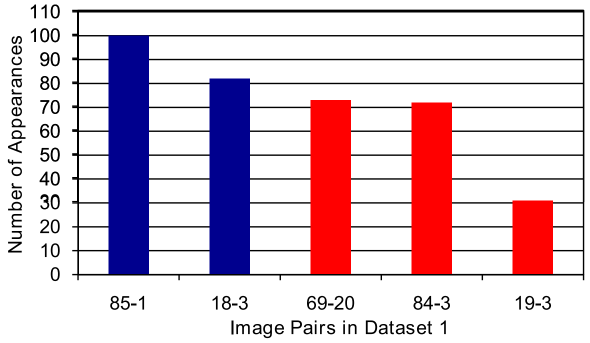
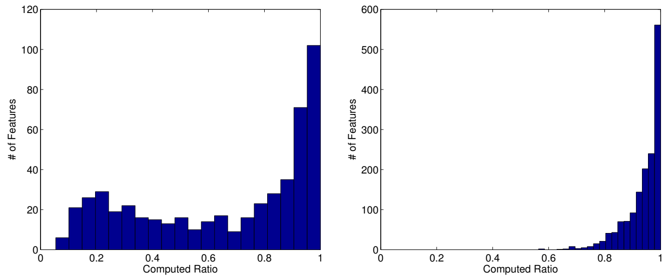
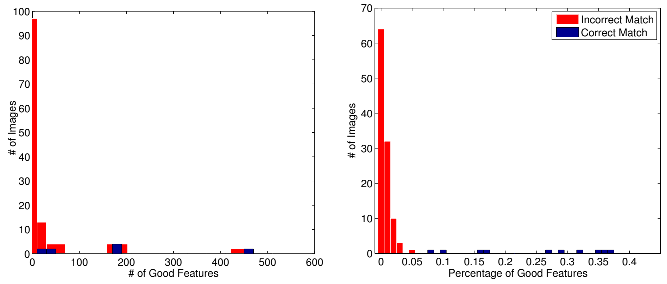


Fig. 2. Image pairs and their # of appearances in 100 trials for dataset 1.



(a) (b)

Fig. 3. The PDF of the ratio of distance of the nearest neighbor to the distance of the second-closest neighbor for two image pairs; (a) a correct pair, 85-1 in Figure 2; (b) an incorrect pair, 69-20 in Figure 2.



(a) (b)

Fig. 4. PDF of (a) the number, (b) the percentage, of features with ratio below 0.6.

B. *Laser/IMU Localization Results*

We use the 2×ICP+IMU and 1×ICP+IMU+Planar algorithms from [11] to characterize the performance of our laser/IMU based localization. In doing so, the loop closures are detected via the approach in Section III-A and TORO optimization is applied to the directed graph with transformations resulting from these two algorithms. Both localization methods use scan matching on the yaw scanner to estimate the backpack pose parameters x, y, and yaw over time. Also, both methods use the InterSense IMU to estimate the backpack roll and pitch over time. For 2×ICP+IMU, scan matching on the pitch scanner is used to estimate z over time. The 1×ICP+IMU+Planar method works only in environments with planar floors and fits a line to the floor to estimate z over time.

We test these two algorithms on four datasets: Dataset 1 is a T-shaped corridor intersection that includes a roughly 20-meter segment of a hallway. Datasets 2 and 3 are of a staircase roughly 4.5 meters in height. Dataset 4 consists of two roughly 15-meter hallway segments connected by a staircase roughly 4.5 meters in height. We first compare 1×ICP+IMU+Planar and 2×ICP+IMU results on dataset 1 as it is the only dataset with a strictly planar floor. For the other datasets, which include staircases, we use 2×ICP+IMU localization as it does not require a planarity assumption. Incremental pose errors are compared in the local coordinate frame specified in Section II. Global position and orientation errors are computed in a frame where x is east, y is north, and z is upward. Note that global errors result from accumulated local errors. As such, their magnitude is for the most part decoupled from the magnitude of local errors. In particular, local errors can either cancel each other out to result in lower global errors, or they can interact with each other in such a way so as to magnify global errors.

Global and incremental pose errors using 1×ICP+IMU+Planar and 2×ICP+IMU for dataset 1 are shown in Figure 5. The results are for loop closure detection, followed by TORO optimization. We see that the two methods are comparable with 1×ICP+IMU+Planar resulting in significantly lower global z error compared to 2×ICP+IMU. This is to be expected since in 1×ICP+IMU+Planar we estimate the absolute value of z, rather than the incremental t_z, at every time step. Thus, the error in z does not have a chance to accumulate over time. However, since 2×ICP+IMU does not make use of a planar-floor assumption, it extends to multi-floor datasets 2, 3, and 4.

Global and incremental pose errors using 2×ICP+IMU across all four datasets are shown in Figure 6. Again, the results are for loop closure detection followed by TORO optimization. We see that compared to dataset 1, datasets 2, 3, and 4 corresponding to the multi-floor datasets, have higher error in global roll and yaw. Other errors remain comparable to the single-floor case of dataset 1. Across all datasets, incremental yaw resulting from horizontal scan matching has lower error than incremental pitch and roll resulting from the IMU. The resulting estimated trajectories closely resemble ground truth trajectories, as shown in Figure 7.

For each dataset, the average position error of the estimated path is reported along with the ground truth path's length in Table I. We find that the average position error relative to the path length for each dataset is small, i.e. around 1% or lower.

The localization results can be applied to the scans from the left vertical geometry scanner in Figure 1 in order to generate a 3D point cloud, which is then processed by a plane fitting algorithm. An example of the 3D point cloud and the resulting 3D plane fitted model for the T-shaped corridor intersection is shown in Figures 8(a) and 8(b) respectively. The plane-fitted model is used later in Section IV to detect occlusions in the image based renderer. Figure 9 shows a triangulated and textured 3D model for two hallways connected by a staircase.

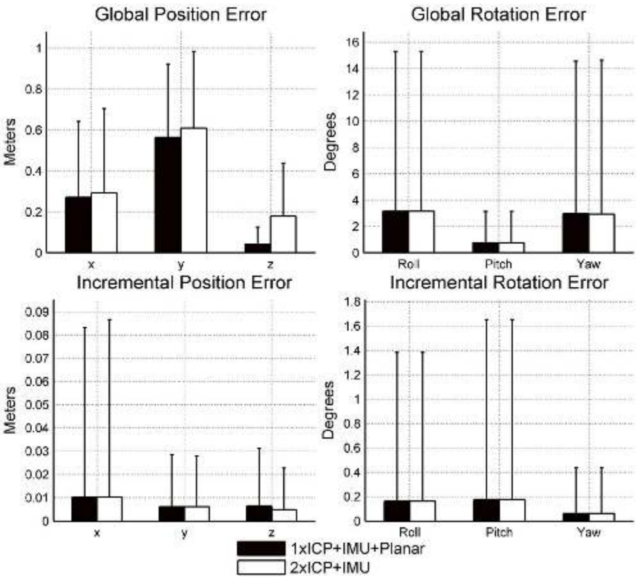


Fig. 5. Global and incremental RMS error characteristics using 1×ICP+IMU+Planar and 2×ICP+IMU on dataset 1. Markers above each bar denote peak errors.

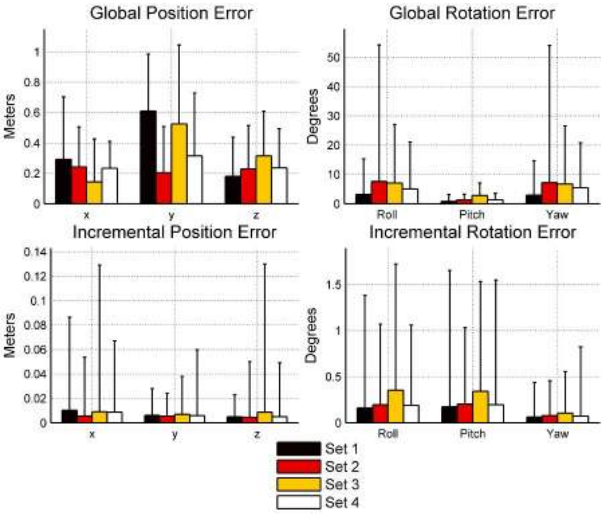


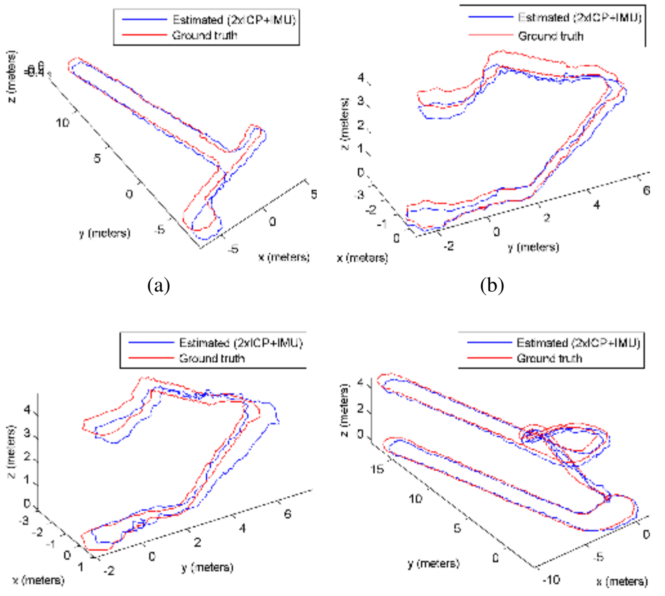
Fig. 6. Global and incremental RMS error characteristics using 2×ICP+IMU across all four datasets. Markers above each bar denote peak errors.

C. *Refining Laser/IMU Localization Using Images*

Using the pose information provided by the localization algorithms discussed in Section III-B, we can transform all captured laser scans into a single 3D coordinate frame. Since camera images are acquired at nearly the same time as a subset of the laser scans, nearest-neighbor interpolation of the pose parameters allows us to estimate the pose of every camera image. Therefore, to generate a 3D model, we (i) transform all laser scans from the floor scanner to a single world coordinate frame and use known methods to create a triangulated surface model from ordered laser data [15],

Dataset	Path length	Average position error
1	68.73 m	0.66 m
2	46.63 m	0.35 m
3	46.28 m	0.58 m
4	142.03 m	0.43 m

TABLE I
GROUND TRUTH PATH LENGTH VS. AVERAGE POSITION ERROR OF THE ESTIMATED PATH FOR EACH DATASET.



(a) (b)

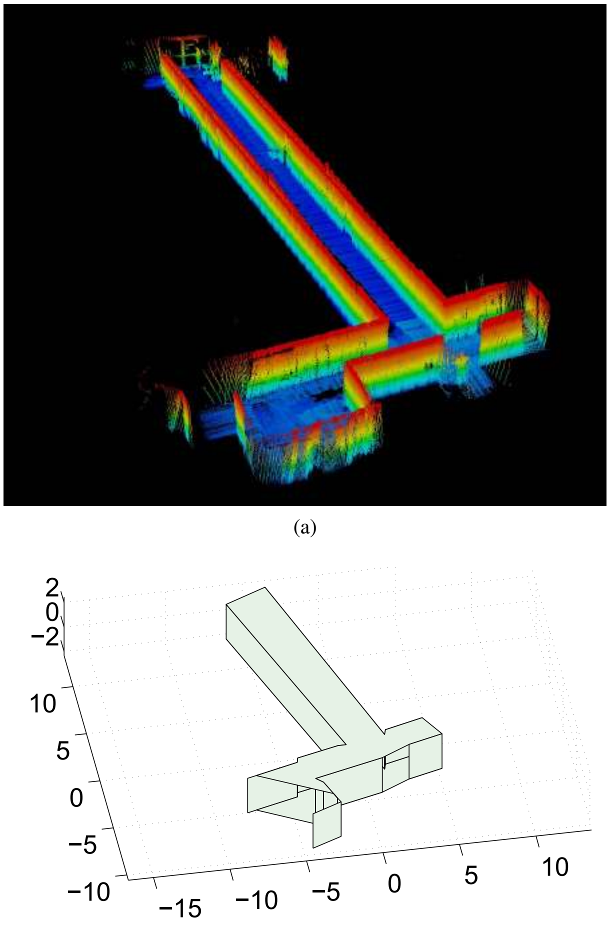
(c) (d)

Fig. 7. Estimated trajectories using 2×ICP+IMU plotted against ground truth for: (a) dataset 1; (b) dataset 2; (c) dataset 3; (d) dataset 4. The estimated trajectory using 1×ICP+IMU+Planar for dataset 1 is similar to that of 2×ICP+IMU.

and (ii) texture map the model by projecting laser scans onto temporally close images. However, we have found that the laser based localization algorithms alone are not accurate enough for building textured 3D surface models. For example, Figure 10 shows a screenshot of a model created by using the 1×ICP+IMU+planar localization results and the resulting texture misalignment. In this section we propose an image based approach to refine the laser/IMU localization results.

Our image based localization refinement scheme uses laser scans acquired at times $(\tau_1, \tau_2, \ldots, \tau_N)$, the estimated poses $\mathbf{X}$ as derived from the scan matching based localization algorithms and camera images acquired at times $(\tau'_1, \tau'_2, \ldots, \tau'_M)$. We assume that $M \ll N$, because we desire image pairs that exhibit enough baseline for reliable SIFT feature triangulation. In particular, for our two datasets over a 30-meter-long hallway, we have $N \approx 3000$ and $M \approx 80$. Further, we can assume that each image is acquired at approximately the same time as one of the laser scans, allowing us to estimate camera poses from $\mathbf{X}$ using nearest neighbor interpolation. In practice, our backpack system guarantees synchronization of laser scans and images to within 50 milliseconds.

Image based localization refinement is a two step process.



(a)

(b)

Fig. 8. (a) 3D point cloud from laser scanners colored by height and (b) a plane-fitted model of a T-shaped corridor intersection.

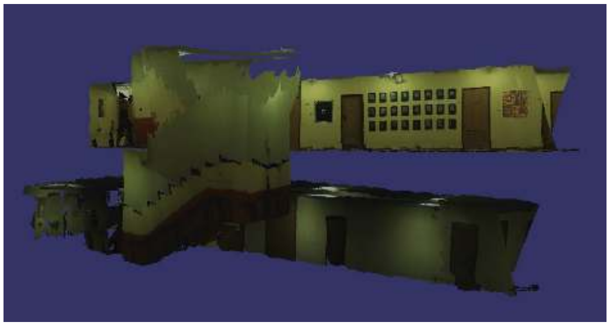


Fig. 9. 3D model of two hallways connected by a stairwell.

First, we estimate the transformations that relate pairwise camera poses. In particular, we estimate the $M-1$ transformations $T'_{1,2}, T'_{2,3}, \ldots, T'_{M-1,M}$ in addition to all loop closure transformations T'_{α_i,β_i}. Second, we obtain a directed graph by adding the estimated pairwise camera transformations as edges to the graph obtained from the scan matching algorithms. An example of such a directed graph with both sets of transformations is shown in Figure 11. Each small node corresponds to a laser scan, and each large node corresponds to a laser scan and

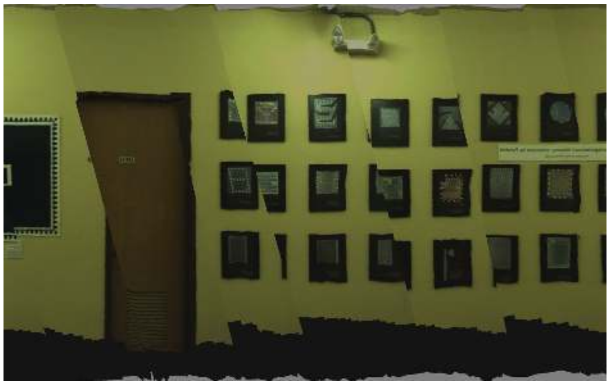


Fig. 10. *Screenshot of textured 3D model generated from localization data without image based refinement. Misalignments between textures from different images are apparent.*

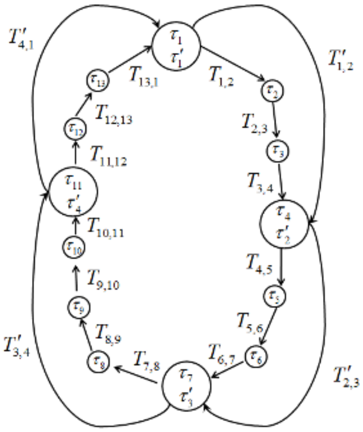


Fig. 11. *Example directed graph for the image based refinement algorithm.*

an image acquired at approximately the same time. Small arcs represent transformations resulting from the scan matching algorithms. Long arcs represent transformations estimated with the image based refinement algorithm to be described shortly. Once a new directed graph with both scan matching and image based edges has been constructed, we perform a round of TORO-based global optimization to redistribute the error among the nodes in order to obtain a set of refined pose estimates $\mathbf{X}$ [10]. To accomplish this, TORO requires as input the covariance error associated with each transformation shown in Figure 11. As shown later, these new pose estimates lead to better image alignment on the final 3D model because they incorporate both laser based and image based alignment.

1) Estimating Pairwise Image Transformations: We now describe the process for estimating the pairwise transformation $T'_{k,l}$ relating camera poses at time τ'_k and time τ'_l. If τ'_k and τ'_l are temporally closest to laser scans at times τ_r and τ_s respectively, $\widehat{T}_{r,s}(\mathbf{X})$ represents an initial estimate for $T'_{k,l}$, where $\mathbf{X}$ is obtained from the scan matching based localization algorithm initially used. Since our initial estimate for $T'_{k,l}$ is reasonably accurate, we propose to use an iterative process that identifies SIFT inliers, and improves the estimate of $T'_{k,l}$ by employing the Levenberg-Marquardt algorithm to minimize the Sampson reprojection error between the two images. For a set of corresponding SIFT features $\{x_q \leftrightarrow x'_q\}_{q=1,2,...,Q}$, the Sampson reprojection error is given by

$$\sum_q \frac{(x'^T_q F x_q)^2}{(Fx_q)_1^2 + (Fx_q)_2^2 + (F^T x'_q)_1^2 + (F^T x'_q)_2^2} \quad (2)$$

where F is the Fundamental matrix that encodes the epipolar geometry relating the two camera poses.

Our iterative refinement based on Sampson reprojection error can only estimate translation up to an unknown scale factor. However, it does provide us with a set of inlier SIFT features; therefore we can estimate the translational scale by aligning triangulated, inlier SIFT features from the side-looking left camera with the 3D laser points acquired by the left vertical geometry scanner. In particular, we perform a multi-resolution search over the unknown scale factor. For each scale, we find the distance between each triangulated SIFT feature and its nearest neighbor in the set of 3D laser points acquired by the floor scanner. We choose the scale factor that minimizes the median distance over the set of inlier SIFT features. The median distance criteria is chosen because it is robust to SIFT outliers that may have been incorrectly identified as inliers.

We have observed that in some cases our initial estimate for $T'_{k,l}$, resulting from one of the scan matching based localization methods, is inaccurate and leads to convergence issues for our iterative refinement method. In most cases, we can detect this situation because it results in either large Sampson reprojection error or poor alignment between the laser points and triangulated SIFT features. When such a situation is detected, we use the normalized five point algorithm [16] in a RANSAC framework to provide a new initial estimate for $T'_{k,l}$ and rerun our iterative refinement method. We only use this new estimate of $T'_{k,l}$ if it leads to a decrease in the Sampson reprojection error or provides better alignment between the laser points and triangulated SIFT features. Thus, the normalized five point algorithm provides a safety net for our iterative refinement scheme if the initial estimate of $T'_{k,l}$ from scan matching is poor.

2) Global Optimization: To obtain the final pose estimates used for 3D modeling, we run a final round of global optimization using TORO [10]. The pairwise camera pose transformations are incorporated as edges in the TORO graph. The edges added to the graph span multiple nodes, as shown by the long arcs in Figure 11. The graph input to the TORO optimizer is $\mathcal{G} = (\{X_1, X_2, \ldots, X_N\}, \mathcal{E})$, where $\{X_1, X_2, \ldots, X_N\}$ are the pose estimates. The set $\mathcal{E}$ includes edges associated with the following transformations: (i) the $N-1$ pairwise transformations $T_{i,i+1}$ and the L loop closure transformations T_{α_i,β_i} obtained via the scan matching based methods and (ii) the $M-1$ pairwise transformations $T'_{i,i+1}$ and the L loop closure transformations T'_{α_i,β_i} obtained via the image based pose estimation.

TORO requires that each edge in the graph have a corresponding 6-DOF transformation and a covariance matrix. It is not straightforward to derive a closed-form covariance matrix for our image based transformations since each one is obtained by first minimizing the Sampson reprojection error followed by minimizing the 3D distance between triangulated SIFT features and nearby laser points. However, the covariances for the laser based scan matching algorithms can be computed using Censi's method [17]. Using our ground truth, we can also perform a one time calibration process whereby for each diagonal element of the covariance matrix we compute the relative scale factor between the laser based and the image based technique. In particular, over a test run of data acquisition, we first estimate laser based transformations and their covariance matrices without loop closures and then estimate image based transformations without loop closures. For both techniques, we also compute the RMS error for each of the 6 pose parameters by comparison with ground truth provided by the Honeywell HG9900 IMU. For practical situations in which the ground truth is not available, we use this relative scaling factor together with Censi's covariance estimate of the laser based localization to arrive at the covariance estimate for image based localization. In particular, to obtain the covariance matrices for the image based transformations, we average the diagonal covariance matrices from all laser based transformations for a given path and scale them based on the square of the relative RMS error between the two techniques. Intuitively, in the global optimization, this covariance estimation method emphasizes parameters estimated accurately by image based techniques, e.g. translation along the direction of motion, while deemphasizing parameters estimated more accurately by laser based techniques, e.g. yaw.

3) Image Based Refinement Results: In all tested cases, the image based localization refinement does result in a significant improvement in the visual quality of the resulting 3D textured models. Screenshots of the textured 3D model for the scene in Figure 10 are shown in Figure 12. As seen, the misalignments exhibited in Figure 10 are significantly reduced. The texture mapped 3D models for a number of datasets using the localization results of Section III can be downloaded from [18].

IV. Image Based Rendering

Image based renderers use a set of images to represent a 3D environment. Renderers vary based on the relationship between the number of input images and the amount of known geometry [19]–[28].

Our approach to image based rendering is to utilize all data products from the backpack acquisition system: high resolution images from cameras, estimated camera poses from localization algorithms, and the relevant geometry computed from 3D point clouds resulting from laser scanners.

For the T-shaped corridor intersection, each of the 3 cameras in Figure 1 provide 903 images during a five minute walk, and the localization algorithms of Section III estimate the 6 dimensional pose of a camera for each image. We use the localization results by converting each pose parameter x, y, z, roll, pitch, and yaw into a 3-vector pose representation for each camera on the backpack as seen in Figure 13. Specifically, the estimated camera pose is represented by 3 vectors in 3D: a vector (a,b,c) for the position of the camera, a vector (a',b',c') for the center of projection of the camera, which represents the orientation, and a normal vector to (a',b',c') denoted by (a'',b'',c'') to represent the up direction.

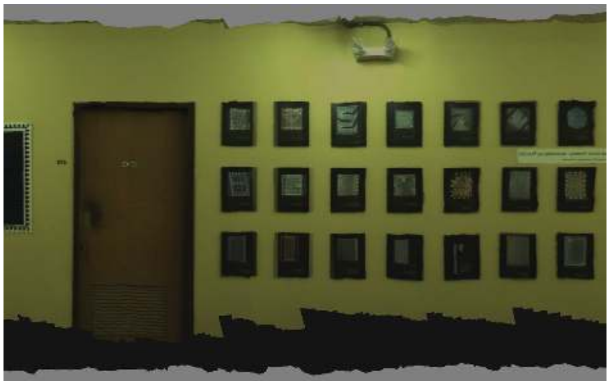


Fig. 12. *Screenshots of textured 3D model generated with 1×ICP+IMU+planar followed by image based pose estimation and texture blending. The 3D model exhibits good texture alignment in comparison with Figure 10, generated with 1×ICP+IMU+planar.*

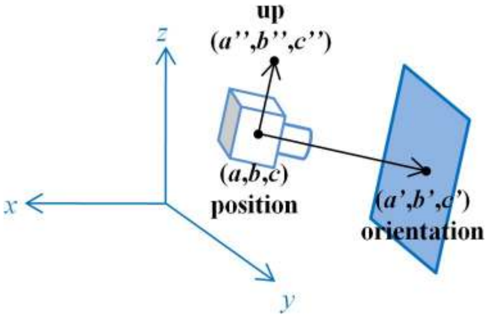


Fig. 13. Camera pose. The position vector is the camera's world coordinates. The orientation vector is the center of projection of the image. The up vector defines the rotation of the camera.

A. Selecting the Image to Render

1) Finding the Best Image: We use a 3-step process to determine which images to render based on the viewer's pose and the set of estimated camera poses. In step 1, if the position vector for a specific camera pose is within a threshold radius of the viewer's position vector, the associated image is added to set A of images to be potentially rendered. The second step involves pruning images in set A that are oriented in the wrong direction to obtain set A'. The dot product between the viewer's orientation vector and the camera's orientation vector provides a metric for how close a given image is to the viewer's image plane. The resulting set of images A' is close to the viewer in both orientation and position; the "best" image in set A' is chosen to be the one with the closest position vector to the viewer's. With a tight dot product threshold in step 2, the chosen image has a camera position closest to the viewer's and an image plane in the same direction as the viewer's orientation with minimal deviation. The sets of images A and

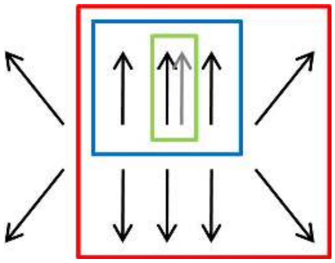


Fig. 14. Finding the best image to render. The grey vector is the viewer's pose and the black vectors represent camera poses. Red indicates neighboring images, blue similarly oriented images, and green closest image.

A' and the "best" image are shown in Figure 14.

2) Rendering the Image: The renderer uses OpenGL's texture mapping functionality to render an image from disk. By defining a plane in the scene, the renderer converts the image pixel array into a texture map of pixel values. The plane is congruent to the image plane of the viewer; therefore, the plane lies on the $z = 1$ plane in OpenGL World Coordinates. The pixel values at indices $\{(0,0),(0,h),(w,0),(w,h)\}$ correspond to the {upper left, upper right, lower left, lower right} texture coordinates of the plane where w and h stand for width and height of the image. OpenGL interpolates the pixel values according to the coordinates in the texture map. This rendering model of converting images to textures is more efficient than storing images into memory; the viewer always sees an image plane perpendicular to the viewer's lens axis.

B. Image Mosaicing for Increased Field of View

Similar to most image based renderers, our image based renderer uses matching features between images to mosaic images together to provide an increased field of view [29]. In doing so, we inherently assume that close by images are related to each other by a 3×3 homography. We determine proximity of images by taking advantage of camera pose information as derived from backpack localization results described in Section III.

Due to the large unstructured data set input to the the renderer, we have taken various steps to optimize for scalability. Features and homographies take an inordinate amount of time to compute; therefore, these processes are precomputed offline. The online or real-time procedure then renders multiple images by stitching them together with the precomputed homographies. The result is that an increase in the number of images or the amount of known geometry only increases the amount of offline calculations, not that of the online rendering.

The offline procedure finds SIFT features for each image and calculates a 3×3 homography between nearby images within the RANSAC framework [16], [30]. The renderer first creates a kd-tree data structure to store the SIFT features in image A. It then uses the kd-tree to traverse the set of SIFT features of a neighboring image, B, to find the nearest neighbors of each node in the kd-tree. If a SIFT feature from image B is similar or close to a feature in image A, then a point correspondence is found between the two images. The process to estimate a 3×3 homography requires enough unique point correspondences to calculate a projective transformation [16]. The resulting SIFT features and homographies are stored on disk for optimal real-time rendering.

C. Online Real-Time Rendering

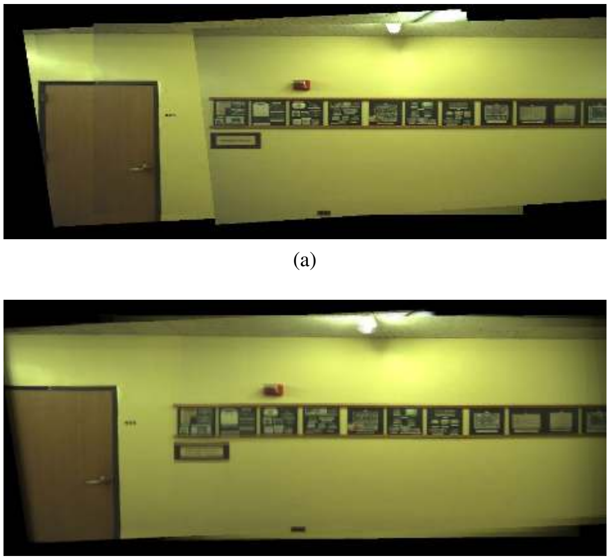
The online process loads the homographies from disk into memory to stitch relevant images to the "best" image chosen in Section IV-A1 for a given view. The renderer performs a similar procedure to determine images considered to be neighbors by taking into account position and orientation relative to the "best" image. These neighbors are cached for future searches, limiting the search calculation to a single pass across all images.

The camera pose information for each image is available from the backpack localization results as described in Section III. In addition to camera poses, the homography parameters such as the number of inliers and the residual error resulting from the homography compilation are taken into account in choosing neighboring images for stitching purposes. Inherently, these transformations assume that the images are coplanar, but in practice the scene is not always a flat environment. Therefore, homographies that transform the image by a rotation of more than 45 degrees of any axis are not applied as they likely correspond to non-planar scenes.

1) Culling and Intersection: If we were to estimate a homography for each pair of images, the run time would become prohibitively long, i.e. on the order of O(n^2) where n is the number of images. However, optimizations can be made by exploiting the known geometry of the environment. Specifically, the backpack localization results can be used to generate a 3D point cloud made up of scan points of the left vertical geometry laser scanner on the backpack. An example of this is shown in Figure 8(a). We can then create a model of the scene by fitting planes to the 3D point cloud, as shown in Figure 8(b). An intersection test can then be used to determine whether two images can be considered neighbors. If a line starting at the position vector of camera pose $P_1 = (x_1, y_1, z_1)$ and ending at the position vector of camera pose $P_2 = (x_2, y_2, z_2)$ intersects a plane represented by equation $Ax + By + Cz + D = 0$ in the model, the image at pose P_1 is occluded from the camera at pose P_2 and vice versa; this is true regardless of the viewer's orientation. This can be quantified in the intersection test equation below:

$$0 < t = \frac{A \cdot x_1 + B \cdot y_1 + C \cdot z_1 + D}{A \cdot (x_1 - x_2) + B \cdot (y_1 - y_2) + C \cdot (z_1 - z_2)} < 1 \quad (3)$$

Consider the line $(x,y,z) = P = P_1 + t * (P_2 - P_1)$ for an arbitrary parameter t; then an intersection point (x,y,z) must satisfy both the plane equation and a line equation. Solving for the variable t is the same as finding t at which the ray $(P_2 - P_1)$ travels from P_1 and intersects the plane with parameters {A, B, C, D}. If $0 < t < 1$, then the line intersects the plane between the two position vectors. In this case, the two images associated with the camera poses are occluded by a plane, and are no longer considered neighbors.


(a)

(b)

Fig. 15. The results of mosaicing images together (a) before and (b) after alpha blending. Note the elimination of the boundary artifacts.

2) Alpha Blending: Neighboring images can have inconsistent lighting or distortion that affect the mosaic; as such, the boundaries between images can be fairly pronounced, diminishing the visual appeal as seen in Figure 15(a). We choose to use a variant of feathering to alpha blend images together in OpenGL. To do so, we divide each plane into a series of one pixel wide planes, and apply a triangular weighting function horizontally to the *i*th one pixel wide plane as follows

$$\alpha_i = \begin{cases} \frac{i}{w/2} & \text{if } 0 \leq i < \frac{w}{2} \\ 1 - \frac{i-w/2}{w/2} & \text{if } \frac{w}{2} \leq i < w \end{cases} \tag{4}$$

where w is the width of the image.

OpenGL renders a pixel value based on the following equation

$$\forall C \in z_{buffer}, \quad C_{dst} \longleftarrow \alpha_{src} C_{src} + (1 - \alpha_{src}) C_{dst} \tag{5}$$

where C_{dst} and C_{src} represent the destination and source colors respectively. The initial destination color is black or $(0,0,0)$ in RGB format. Therefore, the residual darkening of pixel values in each image are negated due to the blending of images. OpenGL then successively applies this blending function to a series of one pixel wide planes, C_j, and their associated alpha values, α_j, as follows:

$$R_0 = (0,0,0)$$
$$R_j \longleftarrow \alpha_j C_j + (1 - \alpha_j) R_{j-1} \quad j = 1, \ldots, N \tag{6}$$

where N is the total number of one pixel planes for that view, R_j is the display value after the first j one pixel planes have been blended, and R_N is the final display value for that view. An example of alpha blending applied to the stitched imagery in Figure 15(a) is shown Figure 15(b). As seen, the boundary artifacts in Figure 15(a) are removed in Figure 15(b).

D. Image Based Rendering Results

We describe the capability of our proposed renderer on a T-shaped hallway with 2709 wall, ceiling, and floor images at 1338×987 pixels from 3 cameras. The rendering machine has 8 Intel Xeon 2.66 Ghz CPUs with 4GB of RAM running 64-bit Ubuntu 8.04 using an nVidia Quadro FX 4600 graphics card with 768MB of memory. The multithreaded renderer takes up to two hours to process SIFT features for 2709 images and up to six hours to find homographies among all such images. The renderer can display single camera images at approximately 20 frames per second. An initial cost of stitching images together reduces the frame rate to 10 frames per second for two stitched images. As expected, an inverse relationship exists between the number of images stitched together and the frame rate. The optimal field of view with high frame rate is around 5 images stitched together resulting in 5 frames per second.

The viewer can navigate the image based renderer using the keyboard and mouse to control translation and orientation respectively. Rotating the view allows the viewer to look up, down, and sideways, corresponding to each of the 3 cameras on the backpack: the top, bottom, and left cameras. In our current implementation, the displayed views are limited to the images taken along the path of the backpack because view interpolation has not yet been implemented. With the estimated pose of the camera, each image is transformed to the world coordinate frame; the images are approximated to have straight lines and right-angle corners. A map of the environment from the plane fitted models directs the user to navigate throughout the scene. The goal of the navigation is to allow the user to quickly and efficiently view the walls, floor, and ceiling at any position within the scene.

Figure 16 shows the renderer's capability to display complex scenes from the T-shaped corridor intersection. These views are particularly difficult to generate because the perspective does not conform to a typical coplanar scene. Still, the image stitching provides enough detail to provide photorealistic depth approximation of the scene. In addition, Figure 17 shows the renderer's ability to change the view to look up and down. A rendered video sequence for the T-shaped corridor intersection can be found in [18].

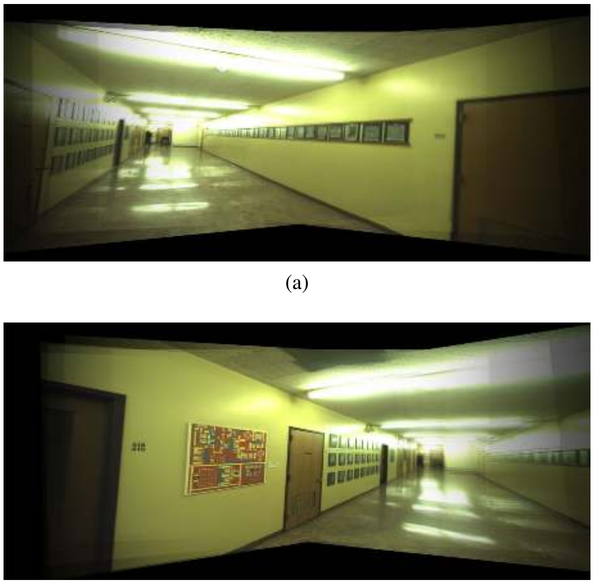
V. Conclusions

We have described a human operated backpack data acquisition system used for virtual walkthroughs inside buildings. We propose two methods for doing so. The first involves explicit 3D textured model construction and the second involves image based rendering. Both approaches require accurate backpack localization; we have shown that a combination of laser scanners, IMUs, and camera imagery can lead to a sufficiently accurate localization for both approaches.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

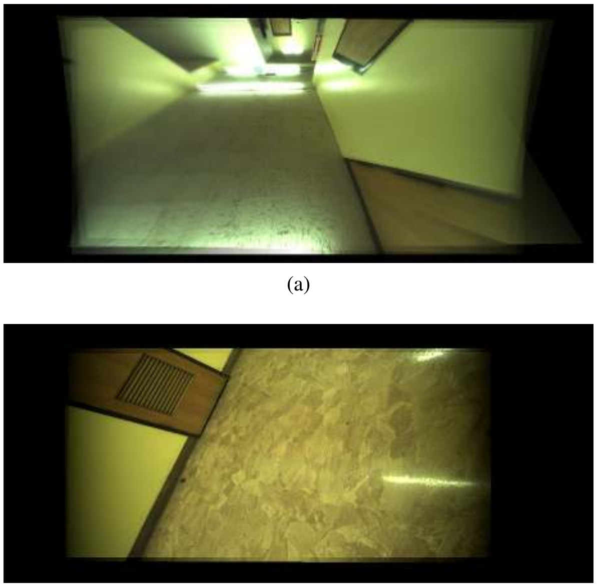
[2] D. Borrmann, J. Elseberg, K. Lingemann, A. Nuchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, pp. 130–142, 2008.



(a)

(b)

Fig. 16. Final mosaiced images of end of the hallway looking (a) slightly left and (b) slightly right.



(a)

(b)

Fig. 17. Rendered mosaics of the (a) ceiling and (b) floor.

[3] V. Pradeep, G. Medioni, and J. Weiland, "Visual loop closing using multi-resolution SIFT grids in metric-topological SLAM," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[4] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a spinning 2D laser," in *Proc. of the IEEE international conference on Robotics and Automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 4244–4251.

[5] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent and Robotic Systems*, vol. 18, pp. 249–275, 1994.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[7] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[8] M. Cummins and P. Newman, "Probabilistic appearance based navigation and loop closing," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007.

[9] K. Granstrom, J. Callmer, F. Ramos, and J. Nieto, "Learning to detect loop closure from range data," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2009.

[10] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3D," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[11] G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, and A. Zakhor, "Indoor localization algorithms for a human-operated backpack," in *3D Data Processing, Visualization, and Transmission*, 2010.

[12] P. Newman and M. Cummins, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] C. I. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, 1968.

[15] M. Carlberg, J. Andrews, P. Gao, and A. Zakhor, "Fast surface reconstruction and segmentation with ground-based and airborne LIDAR range data," in *Proceedings of the Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008.

[16] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[17] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007.

[18] http://www-video.eecs.berkeley.edu/research/indoor.shtml.

[19] H. Shum and S. B. Kang, "A review of image-based rendering techniques," K. N. Ngan, T. Sikora, and M.-T. Sun, Eds., vol. 4067, no. 1. SPIE, 2000, pp. 2–13.

[20] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system," in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1995, pp. 39–46.

[21] D. G. Aliaga and I. Carlbom, "Plenoptic stitching: a scalable method for reconstructing 3D interactive walk throughs," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 2001, pp. 443–450.

[22] P. Debevec, Y. Yu, and G. Boshokov, "Efficient view-dependent image-based rendering with projective texture-mapping," Berkeley, CA, USA, Tech. Rep., 1998.

[23] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 2001, pp. 425–432.

[24] R. Koch, B. Heigl, and M. Pollefeys, "Image-based rendering from uncalibrated lightfields with scalable geometry," in *Proceedings of the 10th International Workshop on Theoretical Foundations of Computer Vision*. London, UK: Springer-Verlag, 2001, pp. 51–66.

[25] M. Uyttendaele, A. Criminisi, S. B. Kang, S. Winder, R. Szeliski, and R. Hartley, "Image-based interactive exploration of real-world environments," *IEEE Comput. Graph. Appl.*, vol. 24, no. 3, pp. 52–63, 2004.

[26] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *Int. J. Comput. Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[27] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski, "Finding paths through the world's photos," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–11, 2008.

[28] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3D," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, 2006.

[29] R. Szeliski, "Image alignment and stitching: a tutorial," *Found. Trends. Comput. Graph. Vis.*, vol. 2, no. 1, pp. 1–104, 2006.

[30] 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, no. 6, pp. 381–395, 1981.