You Are Here: Mimicking the Human Thinking Process in Reading Floor-Plans

Hang Chu, Dong Ki Kim, Tsuhan Chen
Cornell University
{hc772, dk683}@cornell.edu, tsuhan@ece.cornell.edu

Abstract

A human can easily find his or her way in an unfamiliar building, by walking around and reading the floor-plan. We try to mimic and automate this human thinking process. More precisely, we introduce a new and useful task of locating an user in the floor-plan, by using only a camera and a floor-plan without any other prior information. We address the problem with a novel matching-localization algorithm that is inspired by human logic. We demonstrate through experiments that our method outperforms state-of-the-art floor-plan-based localization methods by a large margin, while also being highly efficient for real-time applications.

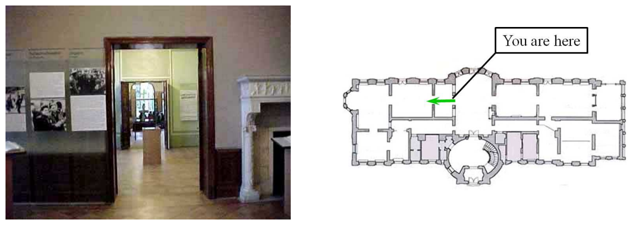


Figure 1: Our system takes a video stream and a floor-plan, and outputs the position and orientation of the current frame in the floor-plan.

1. Introduction

Floor-plans contain useful structure information about building interiors, and they are easy to acquire [38] as all buildings have the plans. Floor-plan related problems, therefore, have received a lot of attention from the computer vision research community [2, 4, 12, 29, 11, 24].

Floor-plan is also widely used in daily life. It is commonly used as a guide in large buildings, such as museums, malls, and laboratories. There are two common scenarios in using the floor-plan. In the first scenario, the floor-plan is found at a fixed location. There is often a ”You are here” arrow in the plan to help a viewer quickly find out where he or she is located. In the second scenario, a tourist has the floor-plan in hand. In this case, there is no the ”You are here” arrow in the plan. Instead, the tourist needs to find out where he or she is by walking around, comparing the observed scenery with the floor-plan, and eventually figuring out the correct location.

Our goal is to help human users by solving the localization problem fully automatically. More specifically, our system stores a floor-plan as prior, takes a video stream of what a tourist sees, and estimates the tourist's current position and orientation in the floor-plan. Figure 1 shows an example of the input and output of our system.

Several reasons make our goal challenging to accomplish. Firstly, the floor-plan does not provide any information about color or texture. Thus, it is hard to use methods based on image feature matching, which rely on previously stored features. Secondly, buildings are composed of repetitive structures. Those repetitive structures, such as corner and corridor, can be observed at multiple locations. Thirdly, the floor-plan can be inconsistent with the real-world. It outlines the building structure, but does not include the furniture. Thus, it is difficult to apply existing conventional techniques in our task:

SLAM Vision-based methods for Simultaneous Localization and Mapping (SLAM) [9, 36, 20, 8, 31, 6, 1] are able to create representational feature maps, and estimate the observer's motion. Although in some applications floor-plans are used, an annotated feature map is still needed to perform localization. Thus, it is difficult to use those methods to localize in a building that has not been mapped beforehand by similar algorithms. SLAM methods are specialized in effective camera tracking and map creation, rather than finding associations between the observed structure and the floor-plan.

Image Retrieval Image retrieval methods localize the camera by matching image features with a database of images with known positions [26, 43, 16], or a prior 3D reconstruction of the environment [22, 15], which further can be made memory efficient by state-of-the-art compression processing methods [28, 5]. However, in our case we do not have such rich prior information. Only a floor-plan is known when the system starts, which makes our problem essentially different from image retrieval.

Vehicle Localization Vision-based vehicle localization methods [14, 3, 40] find the vehicle's current location by analyzing the video stream captured on the vehicle and the topology of the road network. It is difficult to directly transfer these methods to apply to our task, because in the indoor environment the camera moves freely in a 3D space, while vehicle cameras moves constrainedly on the lane.

We solve the problem by gaining inspirations from the human thinking process: a very common localization approach a human would use is to observe one room's size and structure, compare the observation with the floor-plan, and come up with a few possible options. Then move to another room and do the same process, until being certain of the position. Our algorithm mimics this human thinking process. In other words, we help a human user solve the localization problem in the human way.

In this paper we propose: 1. A novel and useful task of indoor localization with only a camera and a floor-plan, without any other prior information. Thus it can be used in any building that has a floor-plan and does not require feature map reconstruction as the preparation. 2. An efficient algorithm that localizes the camera in a floor-plan by mimicking the human thinking process.

2. Related Work

In this section, we examine related previous work in two categories: computer vision research that uses floor-plans, and indoor localization techniques.

Floor-plans in Computer Vision

The floor-plan data has been catching more and more attention in the computer vision research community. It is used in several recent works. In Martin-Brualla et al. [2], floor-plans are used to solve the 3D jigsaw puzzle, which is to find the correct layout of a set of disconnected pieces of 3D reconstruction. In Cabral et al. [4] and Furukawa et al. [12], the floor-plan structure is reconstructed from a set of indoor images. In Liu et al. [29], floor-plan priors are used to accurately register image textures onto walls. In [45, 47, 11, 32] computer vision techniques are applied for indoor structure estimation and indoor scene understanding from images or videos. Though those works are inspirational and useful in their own application scenarios, they can not be directly applied to solve our problem. The work most similar to ours is [24], where an omnidirectional camera is used for localization in a floor-plan. Besides using an unconventional camera, their method is based on overlapping frame detection, which requires closed loops in the camera motion trajectory. Thus, it is difficult to use their method in our task.

Indoor Localization

Indoor localization has been an active area of research in the mobile computing community. Various indoor localization approaches have been proposed, such as indoor localization based on signal beacons [39], magnetometer [7], and sound/ultrasound [42, 41]. Although these methods show promising results, they suffer from an inevitable problem of requiring specialized infrastructures or sensors, which makes them difficult to scale up.

Another line of work makes use of widely existing wireless signals such as GSM and WiFi [35, 19], or the ubiquitous geomagnetism [46]. Methods of this type show better scalability as they do not require additional dedicated beacons or sensors, and have been made publicly available by mobile software such as Google Maps Indoor. However, these methods are not free from mapping survey stages, i.e. signal fingerprint maps are needed for localization. This creates a bottleneck for scalability as well as maintenance difficulties.

The approach that is the closest to ours uses only self-motion estimation and floor-plans [25, 23, 34]. This approach shows strong scalability as neither special sensor nor mapping stage is needed. Also, maintenance of this approach is easy because the building structure seldom changes. The problem of this approach is that a long distance is needed for localization to converge—imagine finding the way with eyes closed. Thus, we propose to improve this approach by giving the system the ability to see. We show in the experiments that visual information significantly improves the localization performance, and our method outperforms the theoretical upper-bound of motion-only methods.

Beyond the mobile computing community, indoor localization has also been studied in [27, 18, 49, 30]. Those approaches use vision sensors along with various other types of sensors, such as WiFi receiver, depth sensor, inertial sensors, and LIDAR. Although those methods provide reliable and effective solutions for indoor localization, it is expensive for them to scale to large number of buildings and large number of users. The problem of self-localization with only a camera and without any other prior database but a floor-plan, while being attractive and potentially useful, remains unstudied.

3. System Pipeline

This section describes the overall pipeline of our system, as illustrated in Figure 2. Our system consists of three major steps.

Motion Estimation and Reconstruction

We use the publicly available software of Semi-Dense Visual Odometry (SDVO) [10, 9] to simultaneously esti-

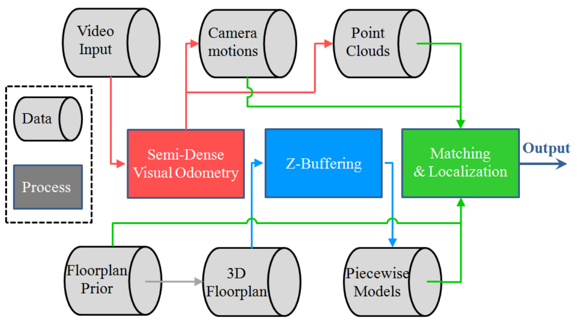


Figure 2: An overview of our system.

mate the camera motion and reconstruct the 3D structure from a video taken by the camera. Comparing to other existing methods such as the well known structure from motion (SfM) [37, 21, 44] and multi-view stereo (MVS) [13], SDVO has the advantage of producing rich information of the building structure as well as being computationally inexpensive. In contrast, the reconstruction of SfM is too sparse to conduct effective analysis with, and the reconstruction of MVS consumes significant amount of computational power, which makes it not applicable for real-time computation.

Generating Piecewise Floorplan Models

We generate the 3D floor-plan model by lifting up the 2D floor-plan. We set multiple unique viewpoints in the 3D floor-plan, then for each viewpoint, we obtain a piecewise floor-plan model by setting a virtual camera at the viewpoint, and applying the z-buffering technique in computer graphics to preserve only the visible part. The z-buffering procedure produces a depth image. Given a depth map and the camera intrinsic, one can construct a point cloud where one point corresponds to one pixel in the depth image. However, such point cloud is spatially non-uniform, i.e. nearby objects have more pixels, thus have higher point density than faraway objects. To overcome this problem and to uniformly represent the piecewise floor-plan, we sample each pixel with a probability proportional to the area of its back-projected square region. More precisely, $p_i \propto depth_i^2$, where p_i is the sampling probability of pixel i. Figure 3 shows an example of piecewise model generation.

Matching and Localization

New video frames are used for an online process of matching and localization. In the beginning, the algorithm does not have an accurate answer. Every position in the indoor space has the same possibility. As the camera moves, 3D models are reconstructed every two seconds. Then matching is performed using the reconstructed model, which helps the estimation converge to the correct current camera position. Matching and localization will be described in Section 4 and 5.

4. Matching by Mimicking Human Logic

When reading the floor-plan, humans often focus on the overall shape of the room, the structure of walls, and the room's size and space. Our matching algorithm tries to mimic this human thinking process, and it consists of three steps: full point cloud matching, reliable structural line matching, and conservative free space matching.

Full Point Cloud Matching

Full point cloud matching matches the point cloud reconstructed from camera $M \in \mathbb{R}^{3\times m}$ and each piecewise floor-plan model $N^{(i)} \in \mathbb{R}^{3\times n_i}$, where m and n_i being numbers of points. We perform full point cloud matching for two reasons. First, we want to search for piecewise models that match well with the reconstruction. However, there can be a slight offset even for the well-matched piecewise models because the actual camera can have an arbitrary pose, in contrast to the piecewise models that are captured with fixed camera viewpoints. Thus we perform alignment between the two point clouds, i.e. it finds $R \in \mathbb{R}^{3\times 3}$ and $t \in \mathbb{R}^{3\times 1}$ such that $M' = RM + t$ is aligned with $N^{(i)}$. Second, we want to compute similarity between the aligned point clouds by $S_{FULL}^{(i)}(M', N^{(i)})$, which is used to help localization.

We use the well known iterative closest point (ICP) method [48] to align the two point clouds. We constrain that the alignment is valid only if it is within a viewpoint grid of $N^{(i)}$. In other words, M can only be translated or rotated slightly to be alignable with $N^{(i)}$, otherwise M is too far away from $N^{(i)}$ to be considered for matching, thus similarity is zero. Moreover, the number of piecewise models is relatively large, and for most of them, $N^{(i)}$ and M are fairly far away. Thus, we only perform ICP for M and a selected subset of piecewise models to avoid unnecessary computation. A piecewise model is selected with M using ICP, only if it satisfies

$$\left(\sum_{k=1}^{m} \mathbf{1}\{||M_k - f_n(N^{(i)}, M_k)|| < d_k\}\right) \geq \alpha m \tag{1}$$

where $\alpha \in (0,1)$, $f_n(N^{(i)}, M_k)$ returns the nearest neighbor of M_k in $N^{(i)}$ using k-d tree, and d_k is the maximum motion with within-grid rotation and translation, which is defined as:

$$d_k = \beta||M_k|| + \gamma \geq \max_{R,t} ||RM_k + t - M_k|| \tag{2}$$

The selection process is controlled by three constants α, β and γ.

For a piecewise model $N^{(i)}$, if (1) is satisfied, we apply ICP to align M with it. Then we measure the similarity between two aligned point clouds as

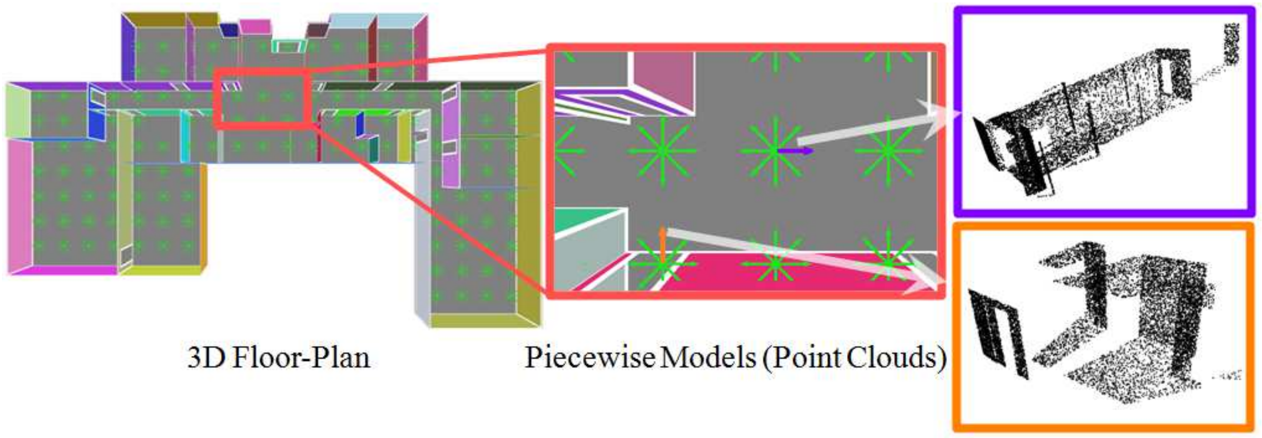


Figure 3: An example of generating the piecewise models.

$$S^{(i)}_{FULL}(M', N^{(i)}) = \frac{1}{m} \sum_j \frac{\lambda_1}{\lambda_j} I_j \tag{3}$$

where $\lambda_{j+1} > \lambda_j > ... > \lambda_1 > \lambda_0 = 0$ are constants of distance thresholds, and I_j counts inliers between λ_j and λ_{j+1}

$$I_j = \sum_{k=1}^{m} \mathbf{1}\{\lambda_j \le ||M'_k - f_n(N^{(i)}, M'_k)|| < \lambda_{j+1}\} \tag{4}$$

Reliable Structural Line Matching

The SDVO reconstruction is based on image edges, thus the full point cloud contains outlines of any objects, such as chairs and desks. This causes troubles for accurate matching because the floor-plan only describes the empty building and does not have any information about outlines of any objects. To tackle this problem, we perform reliable structural line matching: matching using only long straight lines in the SDVO reconstruction, which often correspond to building corners or doorframes, and lines in the piecewise floor-plan models.

Reliable structural lines are defined as long line segments in the depth image. We find the reliable structural lines by first detecting a set of line segments in the depth image. Then we greedily find line segment pairs that are near to each other with a similar angle, and merge them to get longer line segments. Note that we detect reliable structural lines using the 2D depth image rather than the 3D point cloud. This is not only because line detection is often more effective in 2D than 3D, but also because lines that are nearer to the camera are more likely to be selected. Nearer lines often have larger displacement between images, and their depth estimations tend to be more accurate and reliable. Figure 4 shows an example of reliable structural line detection.

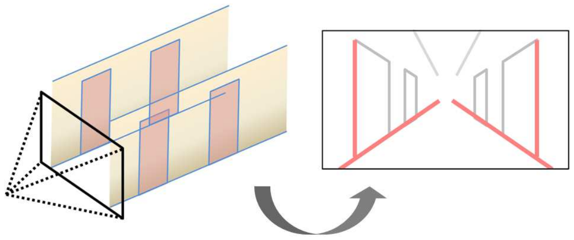


Figure 4: Toy example of reliable structural lines (red).

After reliable structural lines are extracted, we represent them as a point cloud, and use the similar method described through Equation (1)-(4) to determine its similarity to the piecewise models, i.e. $S^{(i)}_{RSL}(M'_{RSL}, N^{(i)}_{RSL})$, where M'_{RSL} is the aligned point cloud of reliable structural lines (if alignable) and $N^{(i)}_{RSL}$ is the point cloud of edges of a piecewise model.

Conservative Free Space Matching

Free space is the space from the camera to the nearest obstacle. Because indoor objects are not included in the floor-plan, the free space observed by the camera will not always match exactly with the free space measured using the floor-plan. However, the observed free space should always be a subset of the free space measured at the correct position in the floor-plan. This property can be used for faster and more accurate localization.

Conservative free space matching takes a reconstructed point cloud and an indoor camera pose as inputs. For the point cloud, we set several directions and estimate the distance to the nearest object in each direction. In the *i*-th direction, we first compute a discrete 1D signal C^i, where $C^i(d)$ equals to the number of pixels that has depth d. If one pixel has a depth value of d, its depth is computed by its inverse depth $1/d$, which is proportional to the between-frame pixel displacement. Assuming the error in pixel displace-

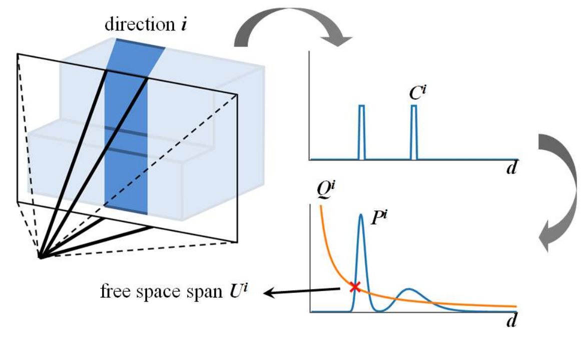


Figure 5: An example of finding the free space span.

ment is Gaussian, the error in the inverse depth estimation is also Gaussian. Thus, the probabilistic depth distribution is computed as

$$P^i(d) = \sum_{d'} C^i(d') G(\frac{1}{d'} - \frac{1}{d}) \tag{5}$$

where $G(x)$ characterizes the Gaussian error in inverse depth estimation $\mathcal{N}(0, \sigma_{id})$. Once we have $P^i(d)$, we compute the free space span as

$$U^i = \begin{cases} \underset{d<D_{max}}{\operatorname{argmin}} P^i(d) \geq Q^i(d) & \text{if such } d \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $Q^i(d) = \eta/d$ is the threshold. The free space estimation is conservative as we set U^i as zero whenever all $P^i(d)$ is below the threshold. This is due to the existence of texture-less objects, such as a blank wall. Texture-less objects are not reconstructed in a monocular vision system, which leaves areas with unknown depth. If no sufficiently dense obstacle is detected in a certain direction, we assign zero free space in that direction to minimize the false positive. An example of finding the free space span is shown in Figure 5.

Similarly to $U = \{U^i\}$, the free space span of a camera pose in the 3D floor-plan $V = \{V^i\}$ is computed with a very small value of σ_{id}. We compute the final similarity of conservative free space matching by measuring how well the subset rule is obeyed:

$$S_{FS}(U, V) = \prod_i \exp(-\delta \cdot \mathbf{1}\{U^i > V^i\}) \tag{7}$$

5. Localization

Localization is performed by applying particle filter using the floor-plan, camera motions, and similarities computed from the three types of matching described above. To begin with, we first briefly summarize the properties of the three types of matching similarities, as listed in Table 1. When the value of S_{FULL} or S_{RSL} is high, the possibility that the matched piecewise model being the correct one is medium, as the structure of full point cloud and reliable structural line is often distinctive but not unique. As S_{FS} measures the inclusion of free space, it has limited selectivity when the value is high, thus low confidence. When S_{FULL} is low, the matched piecewise model can still be the correct one due to the existence of not-mapped objects. It is similar when S_{RSL} is low, but the chance of matching with the correct piecewise model is smaller as the outline of a common object is less likely to become a reliable structural line. When S_{FS} is low, the rule of free space inclusion is violated. Thus, the matched camera pose is very likely to be incorrect as our free space estimation is conservative. Table 1 summarizes by listing the effectiveness of distinguishing the correct location in the first column and the effectiveness of excluding incorrect locations in the second column.

Table 1: Comparisons between different cues from experiments: high similarity values promote correct locations, and low similarity values eliminate incorrect locations.

	high val. conf.	*low val. conf.*
S_{FULL}	22.4% (*medium*)	7.9% (*low*)
S_{RSL}	17.7% (*medium*)	14.5% (*medium*)
S_{FS}	6.8% (*low*)	89.2% (*very high*)

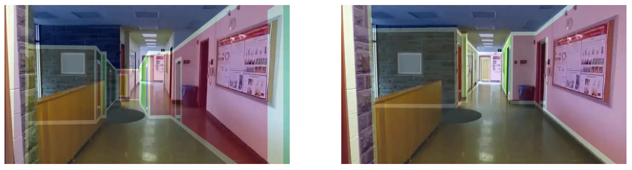
We formulate localization as a particle filtering process, where each particle represents a 6-DoF hypothesis of the current camera pose. Initially, the particles are distributed uniformly in the indoor space. As the camera starts moving, camera motions are estimated and new 3D point clouds are reconstructed and matched every several frames. We apply the estimated camera motion to the particles, and adjust the possibilities of particles using the reasoning presented in Table 1. More formally, the algorithm takes a video stream as input, the floor-plan as prior, and estimates the camera pose of each input frame. Algorithm 1 outlines the localization process.

6. Dataset

There are many existing datasets for indoor location [17, 33]. However, most of them do not apply to our scenario for two reasons. Firstly, they are captured by a fixed-height platform and do not allow fully free 3D camera motion. Secondly, not all datasets provide ground-truth location labels in reference to the floor-plan. Thus, we created our own dataset to test our algorithm. We use a hand-held cellphone camera to capture videos when walking through different rooms. We allow free 3D camera motion during

Algorithm 1 The online-localization process

```
Input: video stream
Prior: floorplan
Output: camera poses of frames
 1: Generate piecewise models
 2: Initialize random particles with equal weights
 3: Take in a new frame, update reconstruction $M_j$, estimate motion
 4: Apply motion to particles
 5: if $M_j$ is complete then
 6:     Compare $M_j$ with all piecewise models, compute $\{S^{(i)}_{FULL}\}$ and $\{S^{(i)}_{RSL}\}$
 7:     Compute $M_j$'s free space $U_j$
 8:     for each particle do
 9:         Compute floorplan free space $V$, compute $S_{FS}$
10:         Adjust weight according to $S_{FS}$, the spatially nearest $\{S^{(i)}_{FULL}\}$ and $\{S^{(i)}_{RSL}\}$
11:     end for
12:     Kill particles with low weights
13:     Replace dead particles by weight-based resampling
14:     Begin reconstructing $M_{j+1}$
15: end if
16: Compute maximum-likelihood camera pose estimation with current particle distribution
17: Go back to 3
```



(a) (b)

Figure 6: An example of the ground-truth labeling process: first roughly estimate the camera pose, and overlay its view in the 3D floor-plan with the actual image, as shown in (a). Then change the 6-DoF camera pose with an interactive interface, until the two views are consistent as shown in (b).

the capturing. Our dataset consists of six videos sequences, including $21,700$ image frames, with total walking distance of $205m$. We manually created 586 6-DoF camera location labels as the ground-truth. Figure 6 describes the location labeling process.

7. Experiments

We ran our experiments on a PC with a Intel-i7 processor. The algorithm was implemented in serial C++ code. We tested our algorithm in a building of size $875m^2$. The building floor-plan only marks walls and doors. We created the 3D floor-plan using room height of $3m$ and door height of $2.3m$. We set a $2m$ grid in the building, and set 12 piecewise model viewpoints with different orientation at each grid. 1608 piecewise models were generated within 10 minutes of computation. The total file size of all generated piecewise models was $74.3MB$.

To evaluate the performance of algorithms, three types of evaluation metrics were used: *Succeed Distance* measures the total walking distance from the beginning of the video to localization success. We define localization to be success if, for all afterwards frames, the estimated position and orientation remain within $1.5m$ and $20°$ to the ground-truth label. *Position Accuracy* and *Orientation Accuracy* measure the average position and orientation error after localization success, respectively in meters and degrees. In all three metrics, the primary goal of our task is to minimize *Succeed Distance*. The shorter it is, the faster the correct location can be found, and the less users need to walk. *Position Accuracy* and *Orientation Accuracy* are secondary compared to *Succeed Distance*, because the accuracies are always below $1.5m$ and $20°$, and it is more important to find out the correct overall location than to improve the precision by several centimeters or degrees.

We applied our algorithm to all six videos in our dataset. In Figure 7-9 we show examples of the three proposed matching methods. It can be seen that all matching methods provide useful information of the camera location. Full point cloud matching and reliable structural line matching produced both good and bad results. Bad matching occurred due to the ambiguity of room structure information, and the inconsistency between the floor-plan and the real world, as we discussed in the introduction. Despite of the possibility of producing bad matching results, the two proposed matching methods improve the final localization performance when all matching results of the whole video are considered, as shown in Table 2. The conservative free space matching is affected by indoor objects. It provides a good way of excluding incorrect hypotheses.

We compared our method with two baseline methods. First is the Mobile Computing Theoretically Best (MCTB) method [25, 23, 34]. [25, 23, 34] propose various approaches to estimate the motion trajectory more accurately, with an inertial sensor and/or a camera. The estimated motion is then used for localization with a particle filtering process similar to ours. Thus, the theoretically best performance of them is achieved when the motion estimation has zero-error. We directly used motions obtained from the ground-truth location labels for the MCTB method. Second is the Scan Matching Particle Filter (SMPF) method [6], where we obtained 2D scans from the estimated depth images and used [6] to match scans with the floor-plan. Beside our dataset, we also evaluated on a public dataset of TUMindoor [17]. We used five videos in the Ground II level, with total walking distance of $606m$. For each video, we

Table 2: *Succeed Distance* (smaller the better), *Position Accuracy*, and *Orientation Accuracy* of the Mobile Computing Theoretically Best (MCTB) method, the Scan-Matching Particle Filter (SMPF) method, and the proposed method, on a public TUMindoor dataset and our dataset.

	Succ. Dis.		*Pos.*		*Orien.*	
dataset	TUMindoor[17]	Ours	TUMindoor[17]	Ours	TUMindoor[17]	Ours
MCTB[25, 23, 34]	$51.92m$	$26.73m$	$0.65m$	$0.82m$	$1.63°$	$5.52°$
SMPF[6]	$16.62m$	$25.77m$	$\mathbf{0.51m}$	$0.72m$	$1.40°$	**$5.21°$**
Proposed	$\mathbf{15.77m}$	$\mathbf{19.02m}$	$0.53m$	$\mathbf{0.50m}$	**$1.29°$**	$5.39°$

Table 3: *fps* of S_{FULL}, S_{RSL}, S_{FS}, localization and total algorithm, as well as memory consumption and *Success Distance* of our full method and its real-time, 33% Piecewise Model version (purple curve in Figure 11). Input *fps* is 29.

	S_{FULL}	S_{RSL}	S_{FS}	*loc.*	*total*	*memory*	*Succ. Dis.*
full	16	148	$> 1k$	355	14	$83.0M$	$\mathbf{19.02m}$
33%PM	42	266	$> 1k$	394	**33**	$\mathbf{26.5M}$	$19.39m$

ran all algorithms five times because of the random particle initialization. Table 2 lists the results. It is noticeable that SMPF and our method outperform MCTB on both datasets. For TUMindoor dataset, our method shows a similar performance to SMPF. However, our method outperforms SMPF when applied to our dataset, especially in the primary goal of *Success Distance*. TUMindoor dataset contains no objects or furniture, but only building structures. Our dataset, on the other hand, contains more practical scenarios with many objects and furniture. This evaluation demonstrates our method's robustness and effectiveness to more complex and realistic environments. For the primary goal of *Succeed Distance*, our method achieves 28.8% and 26.2% improvement compared to MCTB and SMPF, respectively. Figure 10 shows the average errors in position and orientation against the walking distance. The compared methods show high error at the end because they fail to find the correct location in two videos. It can be seen that our algorithm decreases the errors faster, which means the hypotheses converge to the correct solution more effectively. Please refer to the video attachment for more detailed results.

Our method significantly outperforms [25, 23, 34], but it also requires more computation due to performing more matching. In the second experiment, we evaluate the efficiency of our method. Table 3 lists the frame rate of our matching and localization procedures. The video frame rate of our data is 29. It can be seen that the bottleneck is computing full point cloud matching S_{FULL}. To achieve real-time localization, we applied two strategies to improve the frame rate of S_{FULL}: 1. use less reconstructions and compute S_{FULL} less frequently, i.e. compute S_{FULL} only once or twice in every three reconstructions; 2. use less piecewise models, i.e. cut the number of piecewise models by 33% and 66%. Figure 11 shows the experiment results. It can be seen that using less piecewise models is the better

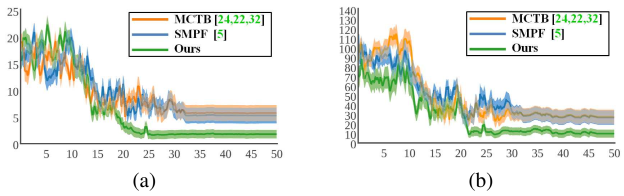


Figure 10: Errors (y-axis) against walking distance (x-axis in m), our dataset. (a) position error (m), (b) orientation error (°). Area shows 20% standard deviation.

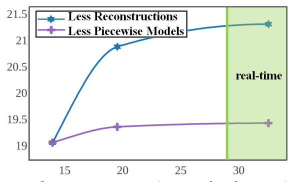


Figure 11: *Succeed Distance* (y-axis in m) and *fps* (x-axis) of speedup strategies.

strategy, the performance only degrades slightly even using only 33% of all piecewise models. This is because we sampled piecewise models every 30°, and using 33% of them keeps piecewise models with 0°, 90°, 180°, and 270° orientations. These piecewise models obey the manhattan-like structure of the building and capture the most useful structural information of the building. Despite slight degradations, all results in Figure 11 are better than the theoretically best performance of [25, 23, 34]. It also should be noted that linear speed-up can be easily achieved by parallelization.

8. Future Work

Our future work will be focused in two aspects. First as shown in Figure 11, in the localization process some piecewise models are more useful than the others, which evokes

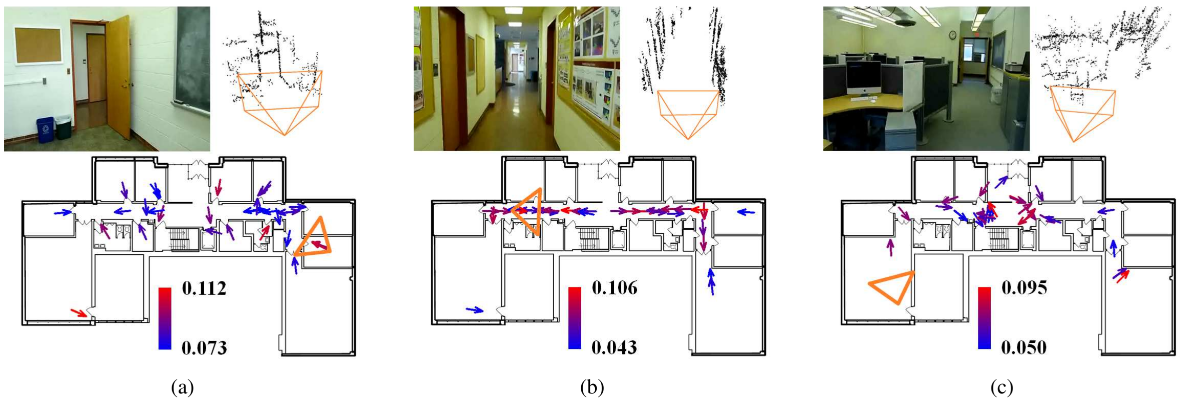


Figure 7: Input image, reconstructed point cloud, and full point cloud matching results. Arrows show 30 matches with highest similarity S_{FULL}, color corresponds to similarity. Orange triangle shows the ground-truth camera pose. (a) and (b) show good matching results, (c) shows a failure case because no arrows are near the ground-truth.

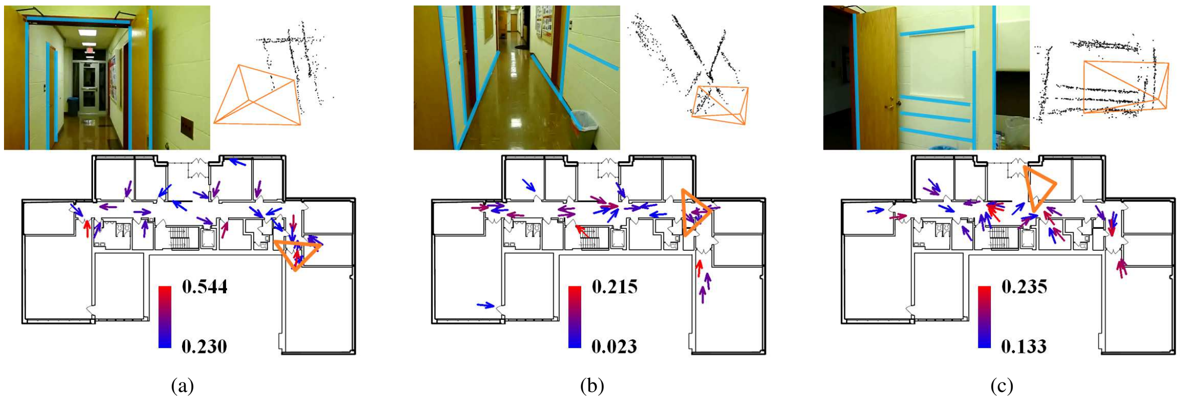


Figure 8: Input image and detected reliable structural lines, point cloud of lines, and matching results. Arrows show 30 matches with highest similarity S_{RSL}, color corresponds to similarity. Orange triangle shows the ground-truth camera pose. (a) and (b) show good matching results, (c) shows a failure case case because no arrows are near the ground-truth.

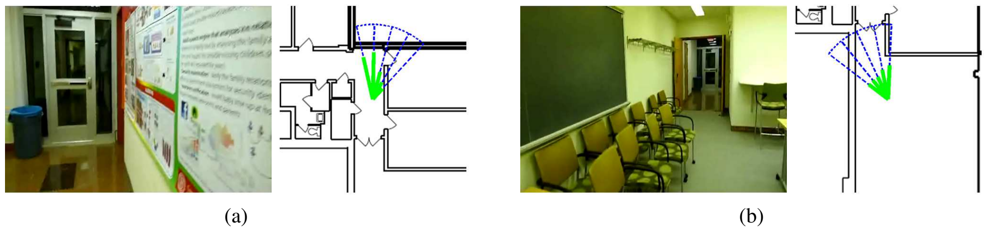


Figure 9: Detected free space plotted at the ground-truth location. In (a) most free space is detected, in (b) only a subset of free space is detected due to the existence of not-mapped objects.

the problem of effective piecewise model selection. Second, rgb-based 3D reconstruction is scale-free, which evokes the problem of automatic scale calibration.

9. Conclusion

We introduced a new and useful task of localizing in the floor-plan by using only a camera without any other prior information. We proposed a novel algorithm that is inspired by human logic. We demonstrated the effectiveness and efficiency of our method through experiments.

References

[1] M. C. Bosse. *ATLAS: a framework for large scale automated mapping and localization*. PhD thesis, MIT, 2004. 1

[2] R. Brualla, Y. He, B. Russell, and S. Seitz. The 3d jigsaw puzzle: mapping large indoor spaces. In *ECCV*, 2014. 1, 2

[3] M. Brubaker, A. Geiger, and R. Urtasun. Lost! leveraging the crowd for visual self-localization. In *CVPR*, 2013. 2

[4] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *CVPR*, 2014. 1, 2

[5] S. Cao and N. Snavely. Minimal scene descriptions from structure from motion models. In *CVPR*, 2014. 1

[6] A. Censi. An accurate closed-form estimate of ICP's covariance. In *ICRA*, 2007. 1, 6, 7

[7] J. Chung et al. Indoor location sensing using geo-magnetism. In *Mobile Sys. App. and Srv.*, 2011. 2

[8] A. J. Davison, I. D. Reid, et al. MonoSLAM: real-time single camera slam. *TPAMI*, 29(6):1052–1067, 2007. 1

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular slam. In *ECCV*, 2014. 1, 2

[10] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *ICCV*, 2013. 2

[11] A. Furlan, S. Miller, D. G. Sorrenti, L. Fei-Fei, and S. Savarese. Free your camera: 3d indoor scene understanding from arbitrary camera motion. In *BMVC*, 2013. 1, 2

[12] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Recon. building interiors from images. In *ICCV*, 2009. 1, 2

[13] Y. Furukawa and J. Ponce. PMVS. 3

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 2

[15] A. Guzman-Rivera et al. Multi-output learning for camera relocalization. In *CVPR*, 2014. 1

[16] J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008. 1

[17] R. Huitl et al. TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. In *ICIP*, 2012. 5, 6, 7

[18] S. Ito et al. W-RGB-D: Floor-plan-based indoor global localization using a depth camera and wifi. In *ICRA*, 2014. 2

[19] Y. Jiang et al. Ariel: Automatic wifi based room fingerprinting for indoor localization. In *Ubiquitous Computing*, 2012. 2

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *IJRR*, 31(2):216–235, 2012. 1

[21] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007. 3

[22] T. Kroeger and L. Van Gool. Video registration to sfm models. In *ECCV*, 2014. 1

[23] K.-C. Lan and W.-Y. Shih. Using smart-phones and floor plans for indoor location tracking. *IEEE Trans. on Human-Machine Systems*, 44(2):211–221, 2014. 2, 6, 7

[24] A. Levin and R. Szeliski. Visual odometry and map correlation. In *CVPR*, 2004. 1, 2

[25] F. Li, C. Zhao, et al. A reliable and accurate indoor localization method using phone inertial sensors. In *Ubiquitous Computing*, 2012. 2, 6, 7

[26] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012. 1

[27] J. Z. Liang et al. Image-based positioning of mobile devices in indoor environments. *Multimodal Location Estimation of Videos and Images*, pages 85–99, 2015. 2

[28] W. Lin et al. Macroblock classification method for video applications involving motions. *IEEE Trans. on Broadcasting*, 58(1):34–46, 2012. 1

[29] C. Liu, A. Schwing, K. Kundu, R. Urtasun, and S. Fidler. Rent3D: Floor-plan priors for monocular layout estimation. In *CVPR*, 2015. 1, 2

[30] T. Liu et al. Indoor localization and visualization using a human-operated backpack system. In *Indoor Pos. and Indoor Nav.*, 2010. 2

[31] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *IJCV*, 94(2):198–214, 2011. 1

[32] G. Pintore and E. Gobbetti. Effective mobile mapping of multi-room indoor structures. *The Visual Computer*, 30(6-8):707–716, 2014. 2

[33] Poly. Univ. of Catalonia. Barcelona robot lab dataset. 5

[34] J. Qian et al. Optical flow based step length estimation for indoor pedestrian navigation on a smartphone. In *Pos. Loc. and Nav. Symposium*, 2014. 2, 6, 7

[35] A. Rai et al. Zee: zero-effort crowdsourcing for indoor localization. In *Mobile Computing and Networking*, 2012. 2

[36] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous loc. and mapping at the level of objects. In *CVPR*, 2013. 1

[37] N. Snavely. Bundler. 3

[38] StreetEasy.com. http://streeteasy.com/. 1

[39] Y. Sun, M. Liu, and M. Q.-H. Meng. Wifi signal strength-based robot indoor localization. In *Information and Automation*, 2014. 2

[40] A. Taneja, L. Ballan, and M. Pollefeys. Never get lost again: vision based nav. using street images. In *ACCV*, 2014. 2

[41] S. P. Tarzia et al. Indoor localization without infrastructure using the acoustic background spectrum. In *Mobile Sys. App. and Srv.*, 2011. 2

[42] M. Uddin and T. Nadeem. SpyLoc: a light weight localization system for smartphones. In *Sensing, Communication, and Networking*, 2014. 2

[43] D. Van Opdenbosch et al. Camera-based indoor positioning using scalable streaming of compressed binary image signitures. In *ICIP*, 2014. 1

[44] C. Wu. VisualSFM. 3

[45] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 2

[46] C. Zhang, K. Subbu, J. Luo, and J. Wu. GROPING: Geomagnetism and crowdsensing powered indoor navigation. *IEEE Trans. on Mobile Computing*, 14(2):387400, 2015. 2

[47] Y. Zhang, S. Song, P. Tan, and J. Xiao. PanoContext: A whole-room 3d context model for panoramic scene understanding. In *ECCV*, 2014. 2

[48] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 13(2):119–152, 1994. 3

[49] Z. Zhu et al. High-precision localization using visual landmarks fused with range data. In *CVPR*, 2011. 2