Towards Indoor Localization with Floorplan-Assisted Priors

Hang Chu and Tsuhan Chen*

***Abstract*— In this paper we investigate the visual indoor localization problem. We take a video sequence as input and a floorplan as prior. Our framework offers immediately available localization service, which requires neither expensive mapping stages nor dedicated sensors. We propose to use (1) architectural component detection to detect doors and corridors with simple features, (2) virtual range finders to explore free space, and (3) matching between the 3D floorplan and visual inputs to fine-tune the camera pose. Our method improves the localization performance in both global localization convergence rate and accuracy. We test and demonstrate the proposed method on real-world videos.**

I. INTRODUCTION

Indoor localization is a key enabling technology for many applications such as navigating to a specific room in an unfamiliar building, finding the fastest exit path during emergencies, and targeted advertisement in shopping malls. GPS provides reliable localization in outdoor environments, but it fails to accommodate most indoor situations; Other indoor localization methods have been explored but they normally require prior mapping stage or special sensors, which makes them difficult to scale-up to thousands of buildings and popularize to millions of users.

We are particularly interested in the following scenario: a person enters an unfamiliar building and gets lost. We want to help localize the person, but she does not carries any unusual device, and the building has not been mapped by any Simultaneous Localization And Mapping (SLAM) system before. This poses a challenging problem.

We present a robust and accurate solution to the problem. We use two types of data source. The first data source is camera. Camera has become a widely available sensor for many people as the increasing popularity of smartphones and development of egocentric vision systems, which makes visual input-only localization systems easy to popularize. The second data source we use is floorplan. This is because most buildings are constructed precisely according to the architectural floorplan map, and floorplans are archived after construction is complete. Floorplans of some buildings, especially public buildings, are already publicly available. For buildings whose floorplan is not publicly available, collecting the archived floorplans is significantly more inexpensive than having all buildings mapped by a SLAM system. Our method is based on these two widely available resources, it takes a video sequence captured by a camera as input and a floorplan as prior. The user moves holding the camera, it takes a while until the algorithm reaches global localization convergence.

*Hang Chu and Tsuhan Chen are with the School of Electrical and Computer Engineering, Cornell University, Ithaca NY, USA hc772@cornell.edu, tsuhan@ece.cornell.edu

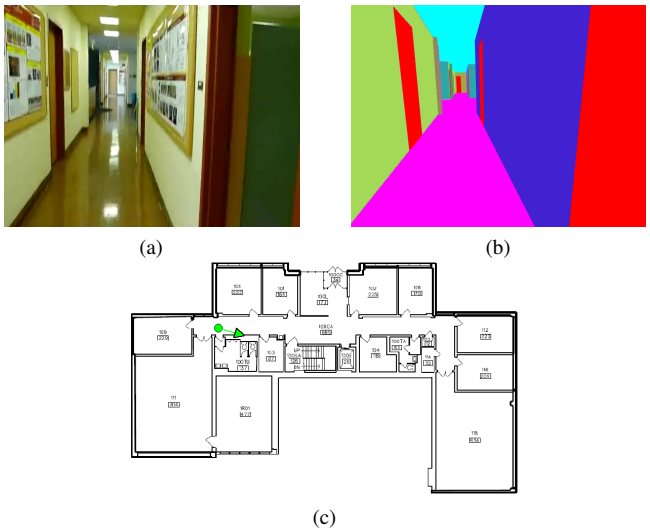


Fig. 1: Demonstration of the proposed method. (a) The current input video frame. (b) A 3D navigation view showing ceiling, floor, walls and doors. (c) Estimated position and orientation in the floorplan. (b) and (c) are available after localization converges.

After convergence, the algorithm keeps producing position and pose of the current frame. Fig.1 demonstrates the inputs and outputs of our method.

In our task, we only know the floorplan of the building, and we do not have neither prior position information nor other constructed maps. Thus we address a global localization problem. Giving a video and a floorplan, a straightforward localization approach is: first derive the camera's motion trajectory from the input video, then match the motion trajectory with the floorplan to find a reasonable path, i.e. place the trajectory on the floorplan. A good placement does not hit any walls. If the trajectory is sufficiently long and the building's structure is not strictly Manhattan-like, the algorithm will eventually converge to the correct position. However, this approach obviously has limitations of low convergence rate and accuracy.

The above simple approach can be improved if we fully utilize all the information embedded in the input data. Besides walls, a floorplan often shows other architectural components such as doors, corridors, and the building's overall structure. Not only the motion trajectory, but also other useful visual clues can be recovered from the input video sequence. Hence in this paper, we propose to use:

- Architectural component detection that identifies if the camera sees certain architectural component such as

door and corridor, which improves localization convergence rate.
- Virtual range finders that are derived from a non-depth camera, which expands the free space and improves both localization convergence rate and accuracy.
- Matching between the 3D floorplan and the visual input, which fine-tunes camera poses after localization convergence and improves localization accuracy.

Finally we combine all proposed techniques with the camera motion, and match them with the floorplan using a particle filtering process. Our localization method is free from prior mapping surveys and can be immediately deployed in any building that has a floorplan.

II. Related work

To address the indoor localization problem, various approaches have been proposed. To name a few, indoor localization based on WiFi beacons [34,5], magnetometer [10], and ultrasound [32] have been recently brought to our attention. Although this category of approaches demonstrate localization capability, only a few of them have achieved meter-level accuracy. Besides, these methods often requires specialized infrastructures and/or sensors, which inflicts additional cost on providing indoor localization service in large scale. Furthermore, for localization methods that uses WiFi fingerprints such as [34,5], a building fingerprint map is needed. This brings about the problem of expensive prior mapping survey, and the problem of handling device heterogeneity. Mapping-free WiFi-based methods have also been proposed such as in Chintalapudi et al. [9], where the algorithm computes the receiver location, the WiFi access point locations, and the signal propagation model simultaneously. This method overcomes the problems of mapping cost and device heterogeneity, but it still suffers from poor signal quality in WiFi-scarce buildings.

In a recent work of Li et al. [24], an indoor localization method using smartphone IMU and a floorplan is proposed. This method solves indoor localization using the motion trajectory in combination with step detection, stride length estimation, and heading inference techniques. Instead of IMU+floorplan in their work, we use camera+floorplan. This change gives us more useful information: IMU only provides motion, camera provides both motion and images. In our approach with more information, more useful analysis can be done to improve the localization performance. Comparing two approaches, a notable limitation of our approach might be that the user needs to hold the camera during the localization process. However, this will no longer be a concern for wearable egocentric vision devices.

SLAM methods [12,27,30] have been studied extensively. Recent development shows improvement in efficiency and maximum map size [20,26]. However, most SLAM methods can only localize themselves in maps that have been previously created with a similar sensor setup, which means they are not free from map construction surveys. This brings about severe cost issue for deploying localization service in large scale. Moreover, it is difficult for traditional SLAM methods to maintain the map up-to-date, as room conditions such as furniture placement can change over time. In contrast, our method uses widely available existing map data. This spares the efforts for large scale map construction survey. Also our method requires minimal map maintaining efforts as only the architectural structure information is needed.

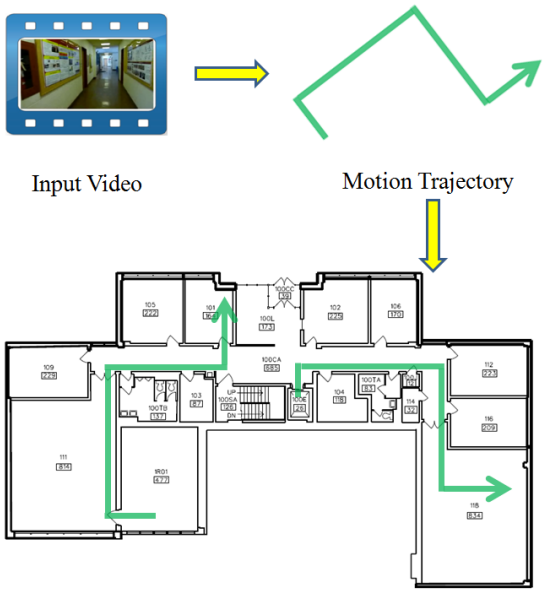
In the work of Brubaker et al. [6] a vehicle self-localization system using a camera and a map is proposed. Comparing their method with ours, we address the indoor localization problem, while [6] addresses the outdoor vehicle localization problem. Although indoor localization and outdoor vehicle localization share similarities, they are different problems in mainly two aspects. First, in indoor environments all the points in the entire open space can be a possible position, while in the outdoor vehicle localization problem possible positions can only be on the roadlanes. This suggests that in our problem the solution space is much larger than in [6]. Thus simplifying the map into a structure graph as in [6] does not work well in indoor environments. Second, most indoor environments are structured and patterned, indicating the possibility of extracting semantic scene information to improve localization performance. In [4,8] visual-based navigation methods are proposed, but their methods work well in outdoor environments rather than indoor environments. Also these methods requires a landmark image database and are hence not free from mapping surveys.

In [15,18,31] indoor visual localization methods using visual odometry are proposed. These methods are based on visual landmark maps and cannot get rid of prior mapping stages. In [2] a localization method using a building floorplan and soft object detection is proposed. Objects such as trash can, phone booth, and ticket machine are detected to improve the motion-based baseline. This method shows promising results. However, prior knowledge of the object positions are needed to use this method, which limits its usage in large scale. In the work of Ito et al. [19] an indoor localization framework that uses depth-aided visual odometry and a floor plan is proposed. Besides the usage of expensive sensors, their framework also relies on WiFi signal strength maps for the initialization of localization. These suggest high expense for using their method in practise.

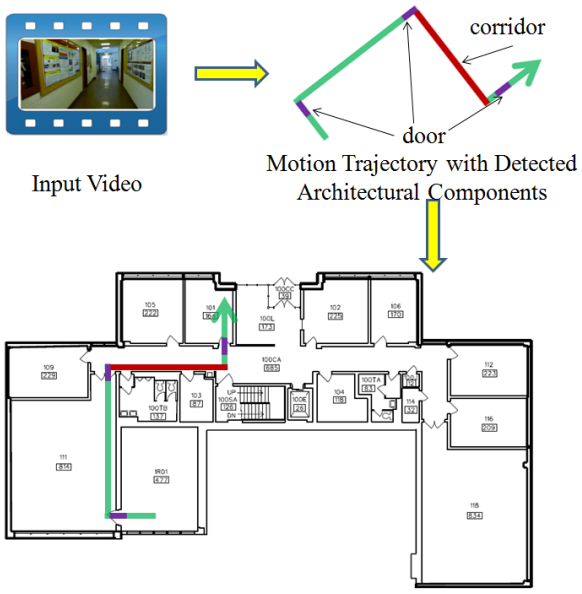
Visual odometry and 3D reconstruction from a video sequence have been studied in [1,13,14,16,17,21,28]. There is also a large body of literature that deals with the problem of improving reconstruction quality using map constraints, such as the work of Robertson et al. [29] and Kaminsky et al. [22]. Although these two lines of work do not deal with the indoor localization problem, they provide useful knowledge and valuable references for our work.

III. Approach

The input and prior of our framework are a video sequence $I = \{I_t\}$ and a floorplan M. The relative camera motion $\Delta \boldsymbol{x}_t$ is computed between frames. For each input frame, we also compute two sets of numbers. Firstly the probabilities of the current frame is looking at each type of architectural component $\alpha_t = \{\alpha_t^i\}$, where i represents the architectural



(a)



(b)

Fig. 2: (a) Using only the camera motion, even with perfect measurement multiple solutions still exist after a long distance. (b) With architectural component detection, only the correct solution remains.

component type. Secondly the current virtual range finder readings $\beta_t = \{\beta_t^j\}$, where j indexes the virtual range finders with different pointing directions. We use particle filter for the localization process. At the starting frame particles are uniformly spread across the entire indoor space. Then at each frame, particles are propagated with $\Delta \boldsymbol{x}_t$, reweighted by α_t and β_t, and resampled based on the updated weights. This procedure repeats and particles converge at T_{cvg}. After T_{cvg}, an fine-tuned camera pose $\boldsymbol{\gamma}_t$ is also computed by matching

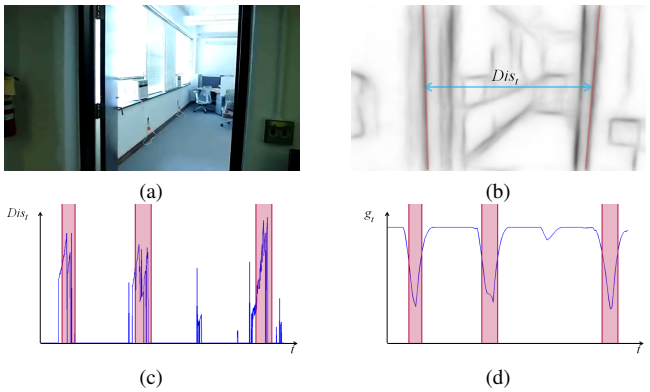


Fig. 3: (a) Input video frame. (b) An example of Dis_t. (c) Example of the Dis signal of a video, with ground truth of going through a door marked in red. (d) Comparing Dis with templates.

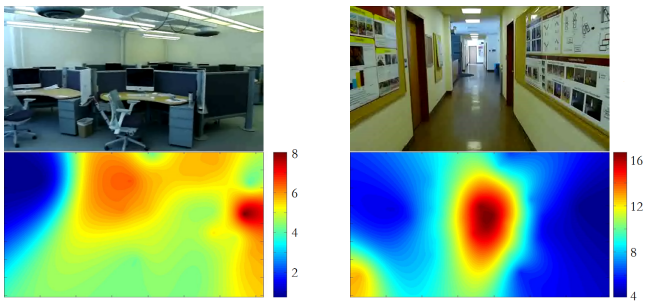
the input frame with the floorplan, and added into the particle filtering procedure.

In the following part of this section, we will introduce in detail architectural component detection, virtual range finders, matching between 3D floorplan with visual input, and combining all described techniques into the localization process.

A. Architectural component detection

In this paper we consider two architectural components: door and corridor. We detect these two components as they are usual in most floorplans, and can be detected effectively. In indoor localization, the information contained in camera motion is insufficient to infer the location. Using only the motion, even with a fairly long trajectory and zero-noise perfect motion measurement, there still could be more than one possible locations. One such example is shown in Fig.2(a). We improve this by incorporating architectural component detection, as shown in Fig.2(b). With architectural component information we can effectively reject incorrect solutions, which increases the localization convergence rate.

Door detection is not a new problem, it has been studied in [3,33]. However, as different doors vary in sizes, colors, and textures, it is difficult to use traditional object-detection type of methods to achieve satisfactory generality and accuracy. In our case, as we have a video sequence instead of single still images. It is possible to detect a sub-sequence where the camera moves through a door, rather than applying object detector for doors in all single frames. For each input frame, we find the rightmost and leftmost long vertical lines from the image edge map computed using [11]. If at least two long vertical lines are observed, the horizontal distance between them is computed as Dis_t, otherwise Dis_t is set zero. When the camera moves through a door Dis increases. Thus we compute the correlation between the Dis signal and a linearly increasing template in different scales Q^k as



(a) (b)

Fig. 4: Input video frame and computed rough depth map RDM, (a) for a non-corridor example and (b) for a corridor example.

$$g_t = \sum_k \sum_{0 \le t' \le L_k} |Dis_{t-t'} - Q^k_{L_k - t'}| \tag{1}$$

where L_k denotes length of the k-th template. An example of the process is shown in Fig.3. The usage of multiple template scales ensures robust detection in different moving speed. The architectural component probability for door α^1_t is then computed as

$$\alpha^1_t \simeq \max\{g_t\} - g_t \tag{2}$$

We detect corridor frames using a rough depth map. The rough depth map is constructed from estimated depths of individual feature points. Many visual odometry/structure from motion methods can be used for feature point depth estimation. We derive the rough depth map RDM using

$$RDM = \sum_m D_m \cdot \mathcal{N}(\boldsymbol{p}_m, \sigma^2_{RDM}) \tag{3}$$

where D_m and $\boldsymbol{p}_m$ are estimated depth and image coordinates of the m-th feature point. As shown in the examples in Fig.4, RDM of a corridor image shows a typical pattern of high depth value in the image central area, while a non-corridor RDM could have any arbitrary appearance. We fit a rotated 2D gaussian function to the 30% pixels with highest depth value in the RDM. Magnitude, central position, horizontal/vertical variances, and rotation angle of the fitted function are recorded to form the feature vector. A linear-SVM [7] is used to learn the model of typical corridor RDM pattern and compute corridor probabilities. Finally, the sequence of single frame corridor probabilities is filtered by a median filter over time to reject lone outlier frames, which gives the final corridor probabilities α^2_t.

B. Virtual range finders

A virtual range finder (VRF) reading represents the distance to the nearest obstacle in its viewing direction. However, the obstacle can be either a wall, or any other arbitrary object that is not shown by the architectural floorplan, such as a table or a chair. This means that unlike traditional SLAM methods, we cannot directly associate range readings with the map. Alternatively, we use VRF as a free space detector: even if VRF detects a non-wall object before it reaches a wall, it still tells us the existence of free space from camera to the detected object. In Fig.5, an illustration of known free space with and without VRF is shown. We can not determine of what is outside the free space (a wall or a non-wall object), but we are certain that within the scope of detected free space, no wall should exist. In this way, VRF helps eliminating incorrect particles more efficiently, which improves localization accuracy and convergence rate.

(a) (b)

Fig. 5: (a) Free space without using virtual range finders (VRF). (b) Free space using VRF. No solid object should exist within the free space, which provides more information for location estimation.

To compute VRF readings, we first calibrate the reconstructed point cloud coordinates. This procedure is performed because the reconstructed point cloud is originally under the coordinate system of the first frame, and the first frame may have any arbitrary pose. We exploit the fact that the camera is held by a person, thus it remains approximately the same height level while it moves. To calibrate the point cloud coordinates, we compute

$$\hat{\boldsymbol{r}}_t = \underset{\boldsymbol{r}}{\operatorname{argmin}} \sum_{1 \le t' \le t} |z(\boldsymbol{x}_{t'}, \boldsymbol{r})| \tag{4}$$

where $\boldsymbol{r}$ defines a 3D rotation and $z(\boldsymbol{x}_{t'}, \boldsymbol{r})$ denotes the z-coordinate of the camera position $\boldsymbol{x}_{t'}$ rotated by $\boldsymbol{r}$. We use the Levenberg-Marquardt Algorithm to solve the minimization problem. The calibrated point cloud is computed by first estimating the optimal $\hat{\boldsymbol{r}}_t$ at every frame, and then rotating the original point cloud by $\hat{\boldsymbol{r}}_t$.

We sum over the calibrated point cloud in the z dimension to compress it into 2D, which forms the top-down view 2D point density map. We use five VRFs pointing at left, left-forward, forward, right-forward, and right. The signal $vrf^j_t(d)$ denotes the point density values along the j-th VRF pointing direction. At last we compute VRF readings as

$$\beta^j_t = \begin{cases} \operatorname{argmin}_d\{vrf^j_t(d) \ge c_d\} & \text{if } \max\{vrf^j_t(d)\} \ge c_d \\ 0 & \text{if } \max\{vrf^j_t(d)\} < c_d \end{cases} \tag{5}$$

where c_d is the threshold constant. In this way, the VRF shows the distance to the nearest obstacle, and shows zero when it senses nothing in its direction within the max

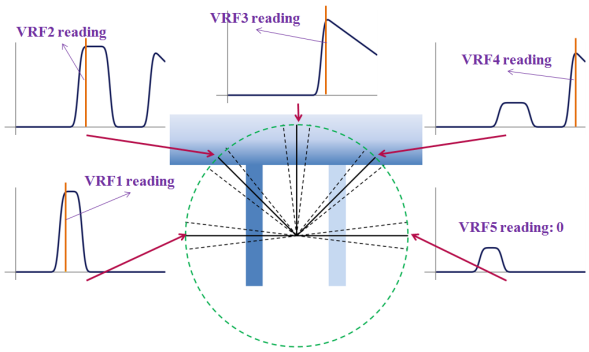


Fig. 6: A toy example of computing VRF readings. Top-down view of the point cloud and 5 VRFs. Blue shows the point cloud, deeper color indicates higher point cloud density. VRF detects the distance to the nearest dense point cloud. Zero is returned if VRF detects nothing within the maximum range.

distance. Note that the second part is a little counter-intuitive: when $\max\{vrf_t^j(d)\} < c_d$, intuitively it seems that nothing is detected and β_t^j should be set as the maximum range. However, this is not always true. As the point cloud is reconstructed with monocular vision input, no point will be reconstructed for featureless objects such as a blank wall. When $vrf_t^j(d)$ detects nothing, it could be either there is a huge free space ahead, or the VRF is looking at an undetectable blank wall. Thus we set β_t^j as the safe choice to make sure the correctness of all detected free space. Fig.6 shows a toy example of computing VRF readings.

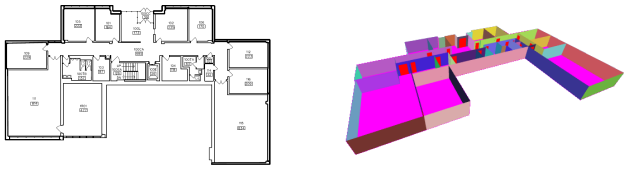
C. Matching 3D floorplan with visual input

Floorplan shows the building's 2D structure. Assuming the room height and door height are known, the 2D floorplan can be lifted into a 3D building model as shown in Fig.7. We match the visual input with the 3D floorplan after localization convergence to further fine-tune the camera pose. After localization convergence, the estimated camera pose is roughly correct. The goal of 3D floorplan-visual input matching is to find the optimal camera pose near the initial estimation, such that at that pose, the building structural lines produced by the 3D building structure model match those in the observed in the input image. Thus we have

$$\boldsymbol{\gamma}_t = \underset{\boldsymbol{x}'}{\operatorname{argmin}} \sum_{(u,v)} I_s^{(u,v)}(\boldsymbol{x}') \cdot dist^{(u,v)}(edge(I_t)) \qquad (6)$$

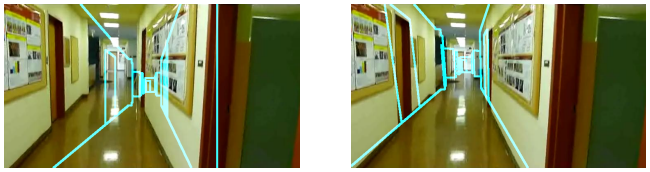
where $I_s(\boldsymbol{x}')$ is the structural line image seen from pose $\boldsymbol{x}'$ in the 3D floorplan, $dist(I)$ denotes the distance transform function of an image, (u, v) denotes the pixel coordinates.

As the computation of the structural line image $I_s(\boldsymbol{x}')$ involves visibility verification, it is difficult to solve the optimization problem analytically. We solve this using a iterative bounded single-dimensional greedy search procedure. In each iteration the optimal value of one dimension of the pose vector is estimated using parabolic interpolation combined with golden section search. Fig.8 shows an example of structural line matching.



(a) (b)

Fig. 7: (a) The 2D floorplan. (b) The lifted 3D floorplan assuming room height and door height are known.



(a) (b)

Fig. 8: The input image overlaid with the seen structural line. (a) shows the initial pose, (b) shows the final optimized pose.

D. Localization procedure

We use particle filter for final localization. A particle represents a current camera pose hypothesis $\boldsymbol{x}_t^n$. Particles are initialized with random positions within whole indoor free space, random orientation angles, and zero tilt angles. For each input video frame, we compute the camera motion $\Delta\boldsymbol{x}_t$, the architectural component probabilities $\{\alpha_t^i\}$, the VRF readings $\{\beta_t^j\}$. At each frame, we first assign equal weights for all existing particle, then we apply the camera motion, and update the particle weights based on how well the current particle matches with estimated features. After all weight updates are done, we resample all particles according to the weights. The weights are updated as

$$w_t^n := w_t^n \cdot \prod_i pow(c_\alpha^i, \max\{A^{ni} - \alpha_t^i, 0\}) \qquad (7)$$

$$w_t^n := w_t^n \cdot pow(c_\beta, \sum_j 1\{B^{nj} < \beta_t^j - \varepsilon_\beta\}) \qquad (8)$$

where n is the particle index, $A^{ni} = \{0, 1\}$ indicates whether the n-th particle belongs to the i-th architectural component, B^{nj} is the distance from the n-th particle to the nearest wall in the floorplan looking from the j-th VRF's pointing direction, ε_β is the VRF measurement noise, $\{c_\alpha^i\}$ and c_β are positive constants less than 1. It is worth noting that A^{ni} can be easily extracted if the original digital architectural floorplan design is available. If only an image version of the floorplan is available, architectural components can be labeled using symbol recognition methods [25]. However, this is beyond the scope of this paper.

Given the particle distribution, the current camera pose estimation $\hat{\boldsymbol{x}}_t$ is determined as the pose that has highest number of adjacent particles, we denote this number N_t. We determine the convergence time as the first time point where $|\Delta\hat{\boldsymbol{x}}_t| < 2|\Delta\boldsymbol{x}_t|$ and $N_t > 0.3N_{total}$ are both satisfied for a certain number of frames frames. After convergence

Algorithm 1 Particle filtering localization procedure

Input: $\{I_t\}$, $floorplan$
Output: $\{\hat{\boldsymbol{x}}_t\}$

Initialize N_{total} particles with random $\boldsymbol{x}_1^n$, $w_1^n = 1/N_{total}$
for each input frame I_t **do**
  Set all weights $w_t^{(n)} = 1/N_{total}$
  Compute $\Delta\boldsymbol{x}_t$, $\{\alpha_t^i\}$, $\{\beta_t^j\}$
  for each particle **do**
    $\boldsymbol{x}_t^n = \boldsymbol{x}_{t-1}^n + \Delta\boldsymbol{x}_t + \boldsymbol{\varepsilon}_m$
    Update weights using Eq.(7), Eq.(8)
    if detect convergence **then**
      Compute $\boldsymbol{\gamma}_t$
      Update weights using Eq.(9)
    end if
  end for
  Resample particles according to $\{w_t^n\}$
  Estimate $\hat{\boldsymbol{x}}_t$
end for

is reached, $\boldsymbol{\gamma}_t$ is computed by matching 3D floorplan with visual input. Besides Eq.(7) and Eq.(8), the particle weights are also updated with

$$w_t^n := w_t^n \cdot \mathcal{N}(\boldsymbol{x}_t^n - \boldsymbol{\gamma}_t, \sigma_{SLM,t}^2) \tag{9}$$

where the variance $\sigma_{SLM,t}$ is computed from all poses measured during the optimization process. The localization procedure is summarized in Algo.1. It should be noted that the localization is an online procedure, where only current and previous information is used at each input frame.

IV. Experimental Results

We apply our method to an indoor video dataset. The dataset contains 2 video sequences recorded using a 25 fps color camera. 12004 unique frames of 720×400 images are recorded. To quantitatively measure errors of localization algorithms, we set 480 measuring points and manually create ground-truth camera 3D poses. We use the LSD-SLAM method [13] for measuring camera motion and creating/updating 3D point cloud.

As there are very few existing methods that is capable of localizing in an unmapped building with only a floorplan and a camera. We design the following 4 methods and compare to them to show the effectiveness of our method: (1) The motion-only method (Mo) that only uses the measured camera motion for localization, which is the most straightforward solution one might come up with; (2) The best possible performance that state-of-the-art IMU-based localization method [24] can achieve, where we assume [24] uses a perfect motion sensor by directly using the ground-truth trajectory as the input of their method; (3) The method that uses both camera motion and architectural component detection (Mo+ACD), which is the proposed method without virtual range finders and 3D floorplan matching; (4) The method that uses camera motion, architectural component detection, and virtual range finders (Mo+ACD+VRF), which is the proposed method only without 3D structural line matching. As particles are initialized randomly, we test each method by running 10 trails and record the average performance. Results are shown in Fig.9, Fig.10, and Table 1.

TABLE I: Qualitative results for different algorithms. The first column shows the average total moving distance from the first frame to localization convergence. The second column shows the average position error after localization convergence. The second column shows the average camera heading direction error after localization convergence.

	Cvg/m	Position/m	Heading/$^\circ$
Mo	32.10	3.58	21.81
[24] Upper Bound	29.09	1.77	14.66
Mo+ACD	21.37	0.63	14.00
Mo+ACD+VRF	**19.83**	0.44	13.27
Proposed	**19.83**	**0.31**	**9.07**

From Fig.10 and Table 1, it can be seen that the proposed method outperforms all compared methods in two aspects: the proposed method can not only localize the camera faster (which is indicated by shorter moving distance to convergence), but also estimate camera poses more accurately. Using the method in [24], even if we assume we have the means to measure/estimate the motion trajectory perfectly (to which vast effort has been spent but the problem still remains unsolved), the upper limit such method can achieve would only beat the baseline that uses a noisy motion estimation. By using semantic information extracted from image data (Mo+ACD, Mo+ACD+VRF, and proposed), localization performance can be easily improved by a large margin.

In Fig.9, the integrated camera motion along with architectural component detection results and virtual range finder values are shown. It can be seen that most door and corridor frames are detected successfully. As our architectural component detection procedure is based on relatively simple features (lines and rough depth map), some failure cases also occur. For example in the second video (right in Fig.9), the door detector is triggered when the camera passes two building corners. However, comparing Mo+ACD with Mo (Table 1 and Fig.10), it can be seen that the true positives of ACD improves localization performance more than its false positives harms the performance.

In the first video (left in Fig.9), the VRF only detects a very narrow free space when it passes the big room due to objects such as desks and chairs occludes the virtual beam. By adding VRF, we achieve slightly better convergence rate than Mo+ACD (Table 1 and Fig.10). This is because although VRF helps in ruling out bad particles faster, the constraint provided by ACD is already fairly strong such that VRF cannot accelerate converging significantly. However, adding VRF proofs to be more useful in improving the positioning accuracy. VRF confines particles always certain distance away from walls, which reduces the error propagation effects.

The proposed method shows the same convergence rate

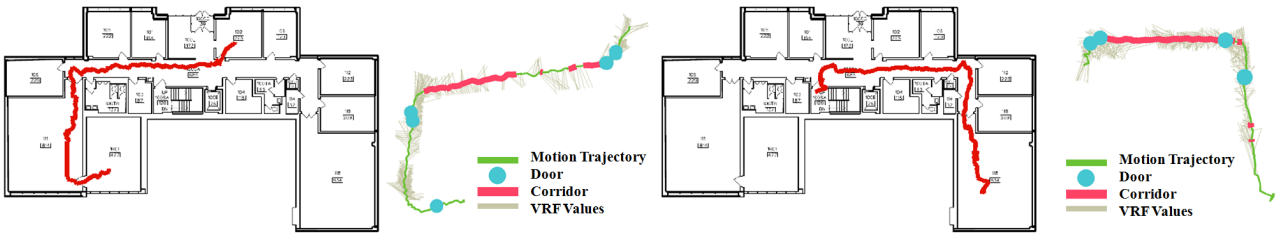


Fig. 9: Experimental results of applying different algorithms to our dataset. The ground-truth trajectories and the of measured camera motion, detected architectural components and estimated virtual range finder (VRF) readings.

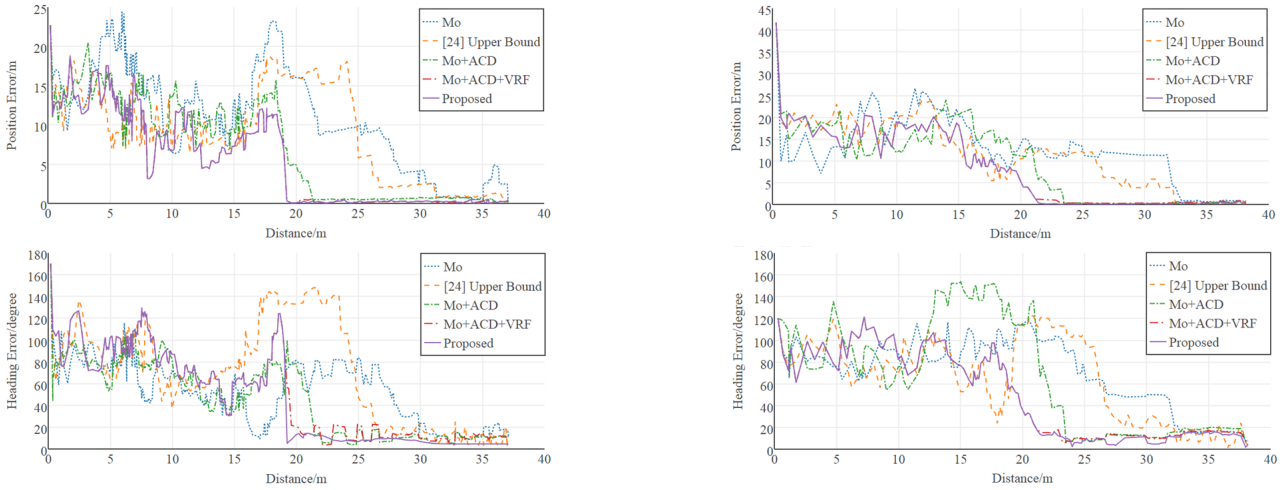


Fig. 10: Error diagrams of applying different algorithms to our dataset. Left column shows the position error of $\{\hat{\boldsymbol{x}}_t\}$ in meter over total moving distance since the first frame. Right column shows the heading direction error of $\{\hat{\boldsymbol{x}}_t\}$ in degree over total moving distance since the first frame.

as Mo+ACD+VRF because 3D floorplan matching is only triggered after convergence. By additionally searching for a camera pose that produces better matching between the image and the 3D floorplan, the proposed method achieves higher localization precision in both position and camera heading direction, this can be seen in Table 1 and Fig.10.

As a by-product of the 3D floorplan matching process, our method is able to not only produce the camera's 3D pose, but also an animated virtual 3D view after localization converges. In Fig.11, we show examples of input video frames and generated virtual 3D views. The top three image pairs show examples where camera pose is accurately estimated. The bottom three image pairs show failure cases where the 3D floorplan matching is interfered by irrelevant non-structural textures in the input image, such as edges of the wooden frame in the input image being matched to a door edges in the 3D floorplan model.

More details can be found in the video attachment.

V. Conclusion

A indoor localization method using only a camera as input and a floorplan as prior was proposed. To improve localization performance, the proposed method uses architectural component detection, virtual range finders, and matching between the 3D floorplan and visual inputs. The proposed method was tested on real-world videos.

References

[1] P. F. Alcantarilla, L. M. Bergasa, and F. Dellaert, Visual odometry priors for robust EKF-SLAM. *ICRA*, 2010.

[2] R. Anati, D. Scaramuzza, K. G. Derpanis, and K. Daniilidis, Robot localization using soft object detection. *ICRA*, 2012.

[3] D. Anguelov, D. Koller, E. Parker, and S. Thrun, Detecting and modeling doors with mobile robots. *ICRA*, 2004.

[4] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys, Handling urban location recognition as a 2D homothetic problem, *ECCV*, 2010.

[5] P. Bahl and V. Padmanabhan, Radar: an in-building RF-based user location and tracking system. *INFOCOM*, 2000.

[6] M. Brubacker, A. Geiger, and R. Urtasun, Lost! Leveraging the crowd for probabilistic visual self-localization. *CVPR*, 2013.

[7] C.-C. Chang and C.-J. Lin, LibSVM: a library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2(3):27, 2011.

[8] D. M. Chen, G. Baatz, et al., City-scale landmark identification on mobile devices. *CVPR*, 2011.

[9] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, Indoor localization without the pain. *Intl. Conf. on Mobile Computing and Networking*, 2010.

[10] J. Chung, M. Donahoe, et al., Indoor localization sensing using geomagnetism. *Intl. Conf. on Mobile Systems, Applications, and Services*, 2011.

[11] P. Dollár and C. L. Zitnick, Structured forests for fast edge detection. *ICCV*, 2013.

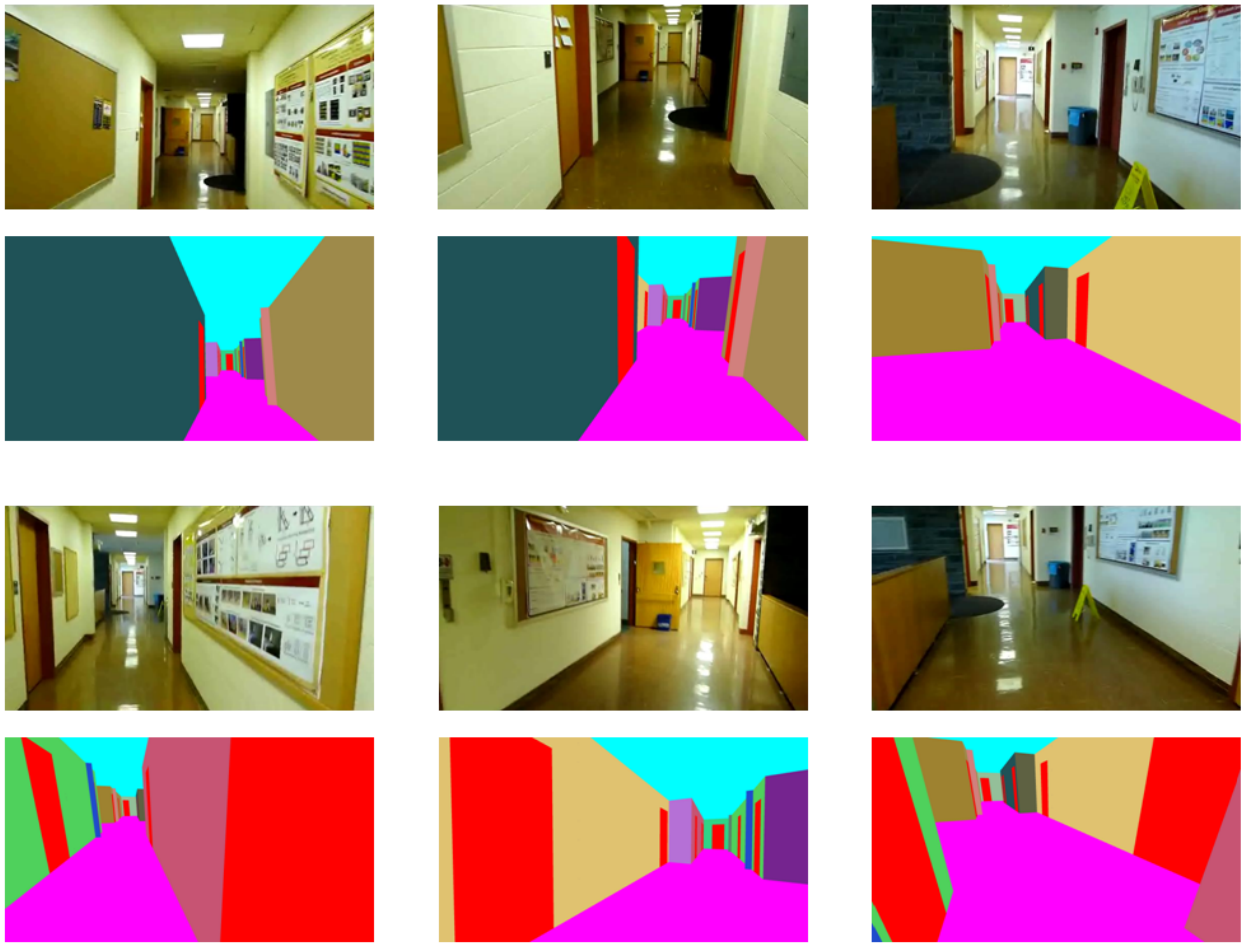


Fig. 11: Examples of generated virtual 3D views. First and third rows show the current input video frame, second and fourth rows show the virtual 3D view generated using the estimated camera pose after localization converges.

[12] H. Durrant-Whyte and T. Bailey, , Simultaneous localization and mapping: part I. *Robotics & Automation Magazine*, 13(2):99-110 2006.

[13] J. Engel, T. Schöps, and D. Cremmers, LSD-SLAM: large-scale direct monocular SLAM. *ECCV*, 2014.

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

[15] N. Ganganath and H. Leung, Mobile robot localization using odometry and Kinect sensor. *Emerging Signal Processing Applications*, 2012.

[16] A. Geiger, J. Ziegler, and C. Stiller, Stereoscan: dense 3D reconstruction in real-time. *IV*, 2011.

[17] R. Hartley and A. Zisserman, Multiple view geometry in computer vision. *Cambridge University Press*, 2003.

[18] S. Hilsenbeck, A. Moller, R. Huitl, G. Schroth, M. Kranz, and E. Steinbach, Scale-preserving long-term visual odometry for indoor navigation. *Indoor Positioning and Indoor Navigation*, 2012.

[19] S. Ito, F. Endres, M. Kuderer, G. D. Tipaldi, C. Stachniss, and W. Burgard, W-RGB-D: Floor-plan-based indoor global localization using a depth camera and wifi. *ICRA*, 2014.

[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.

[21] M. Kaess, K. Ni, and F. Dellaert, Flow separation for fast and robust stereo odometry. *ICRA*, 2009.

[22] R. S. Kaminsky, N. Snavely, S. M. Seitz, and R. Szeliski, Alignment of 3D point clouds to overhead images. *CVPR Workshops*, 2009.

[23] G. Klein and D. Murray, Parallel tracking and mapping for small AR workspaces. *ISMAR*, 2007.

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

[25] J. Lladós, E. Martí, and J. J. Villanueva, Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *TPAMI*, 23(10):1137-1143, 2001.

[26] 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):198214, 2011.

[27] M. Montemerlo, S. Thrun, D. Koller, B.Wegbreit, et al., Fast-SLAM: a factored solution to the simultaneous localization and mapping problem. *AAAI*, 2002.

[28] D. Nistér, O. Naroditsky, and J. Bergen, Visual odometry. *CVPR*, 2004.

[29] D. Robertson and R. Cipolla, Building architectural models from many views using map constraints. *ECCV*, 2002.

[30] S. Se, D. Lowe, and J. Little, Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *IJRR*, 21(8):735758, 2002.

[31] J. Straub, S. Hilsenbeck, et al., Fast relocalization for visual odometry using binary features. *ICIP*, 2013.

[32] S. P. Tarzia, P. A. Dinda, et al., Indoor localization without infrastructure using the acoustic background spectrum. *Intl. Conf. on Mobile systems, Applications, and Services*, 2011.

[33] X. Yang and Y. Tian, Robust door detection in unfamiliar environments by combining edge and corner features. *CVPR Workshops*, 2010.

[34] M. Youssef and A. Agrawala, The horus WLAN location determination system. *Intl. Conf. on Mobile systems, Applications, and Services*, 2005.