Towards Indoor Localization with Floorplan-Assisted Priors

Anonymous CVPR submission

Abstract

In this paper we investigate the problem of visual indoor localization. We take a video sequence as input and a floorplan as prior. Our framework offers immediately available localization service, which does not require sophisticated methods such as those based on simultaneous localization and mapping (SLAM). We propose to use 1) architectural component detection that detects doors and corridors with simple features, 2) virtual range finders that expands the free space, and 3) matching between the 3D floorplan and visual inputs that fine-tunes the camera pose. Our method shows significant improvement in the final localization performance in terms of both localization accuracy and convergence rate. We test and demonstrate the proposed method on real-world videos. The labeled dataset and source code will be available online.

1. 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. While GPS provides reliable localization service in most outdoor environments, its signal is often blocked inside a building, which leaves indoor localization an open problem.

In this paper, our goal is to find a robust and accurate indoor localization solution. We are particularly interested in two types of data sources. The first data source is camera. With the increasing popularity of smartphones and development of egocentric vision systems, camera becomes a widely available sensor for many people. The second data source is floorplan. As a common practice in the construction industry, most buildings are constructed precisely according to the architectural map, i.e. the floorplan. Most floorplans are archived after the construction is completed, some of them are publicly available especially for public buildings. We build our framework based on these two types of widely available resources. Our framework takes a video sequence captured by a camera as input and a floorplan as prior. After localization converges, it produces position and pose of the current frame. Fig.1 illustrates inputs and outputs of our framework.

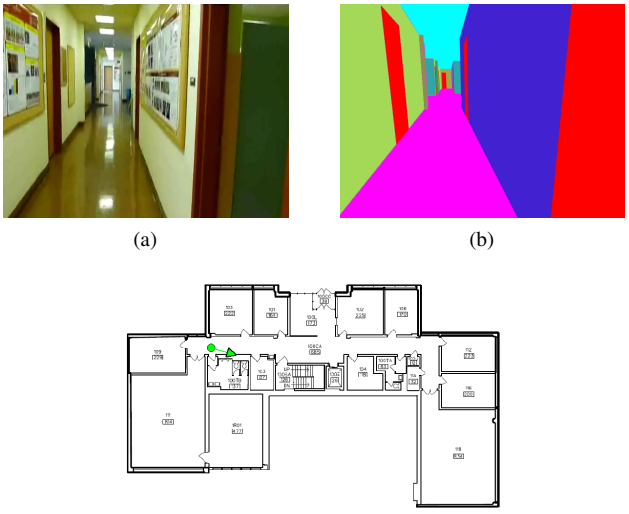


Figure 1: Illustration of the proposed framework's functionality. (a) The current input video frame. (b) A 3D navigation view showing ceiling, floor, walls and doors. (c) Estimated position and orientation in the floorplan. (b) and (c) are available after localization converges.

In our task only floorplan of the building is known. Neither prior position information nor prior map construction survey is required. Thus our task falls into the global localization problem category. Given a video and a floorplan, a straightforward localization approach is to 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 is a placement that 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 at the correct position. However, this approach has limitations for its low convergence rate and limited accuracy.

The simple approach described above can be significantly 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 and corridors, and the building's overall structure. Using the input video sequence, not only the motion trajectory, but also other useful visual clues can be recovered. 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.
- A process for matching between the 3D floorplan and the visual input, which fine-tunes camera poses after localization convergence and improves localization accuracy.

We finally 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 map construction surveys and can be immediately deployed in any building that has a floorplan.

2. 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 people's 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 cost for prior map construction survey, and the problem of handling device heterogeneity. Mapping-free WiFi-based methods have also been proposed, such as the work of 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 as an IMU can only offer motion, but a camera provides both motion and image. With more information provided by the camera, more analysis can be made to improve the localization process in our approach. 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.

Simultaneous Localization And Mapping (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 serious cost issue for deploying localization service in large scale. Moreover, for traditional SLAM methods it is difficult 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 our method, we address the indoor localization problem, while [6] deals with the outdoor vehicle localization problem. Although indoor localization and outdoor vehicle localization share some 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 analysing the scene to improve localization performance. In [8, 4] 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 map construction surveys.

In [15, 31, 18] 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 trashcan, phone booth, and ticket machine are detected to improve the motion-based baseline. This method shows

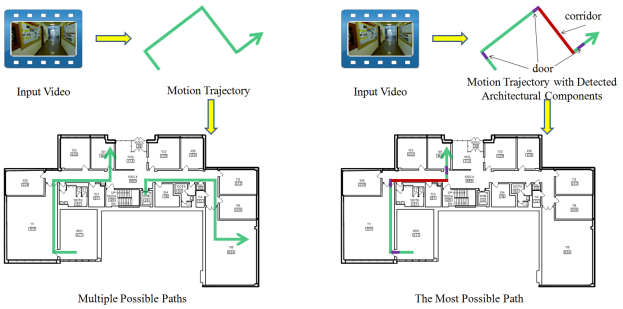


Figure 2: (a) Using only the camera motion, even with perfect measurement there still could be multiple solutions after walking a long distance. (b) With architectural component detection, only the correct solution remains.

promising results. However prior knowledge of the object positions are necessary in 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 sensor, their framework also relies on WiFi signal strength maps for localization initialization. These suggest high cost for using their method in practise.

Visual odometry and 3D reconstruction from a video sequence have been studied in [17, 28, 1, 16, 21, 14, 13]. 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.

3. 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 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 uniformly spread the entire indoor space. Then at each frame, particles are propagated using $\Delta \boldsymbol{x}_t$, reweighted using α_t and β_t, and resampled based on the updated weights. This process goes on and particles converge at T_{cvg}. After T_{cvg}, an fine-tuned camera pose $\boldsymbol{\gamma}_t$ is also computed by matching the input frame with the floorplan, and added into the particle filtering process.

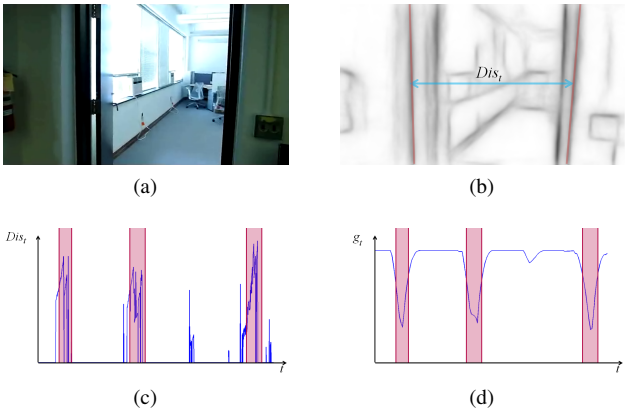


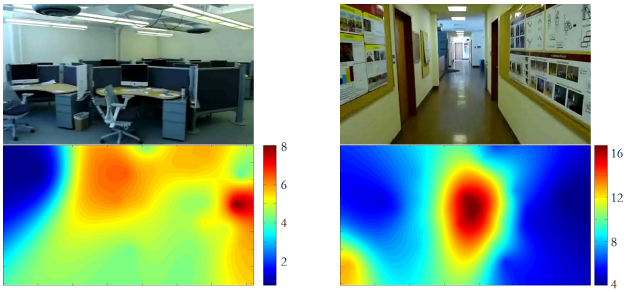
Figure 3: (a) Input video frame. (b) An example of Dis_t. (c) Example of the Dis signal of a video, with ground truth of going through a door marked in red. (d) Comparing Dis with templates.

In the following part of this section, we will introduce the architectural component detection, the virtual range finder, the structural line matching, and combining all described techniques into the localization process in detail.

3.1. Architectural component detection

In this paper we define two types of architectural components: door and corridor. These two types are used as they can be often found in a floorplan, as well as can be detected effectively. In indoor localization, the information contained in camera motion can be insufficient to infer the location. Using only the motion, sometimes even with a fairly long trajectory and zero-noise, perfect motion measurement, there could be still more than one possible locations. One such example is shown in Fig.2(a). This can be improved by incorporating architectural component detection, as shown in Fig.2(b). With architectural component information false-positives are effectively rejected, 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, using traditional object-detection type of method is difficult to achieve satisfactory generality and accuracy. In our case, we deal with a video sequence instead of a single still image. We detect a sub-sequence where the camera moves through a door, rather than running the door objects detector in any particular solitary frame. 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 forms an increasing slope shape. The Dis signal is compared with



(a) (b)

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

a linearly increasing template in different scales Q^k as

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

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

$$\alpha^1_t \simeq \max\{g_t\} - g_t \quad (2)$$

We perform architectural component detection for corridor 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. It is worth mentioning that our approach is not tied to a specific odometry/reconstruction algorithm. The choice of method is flexible, it can be either a PTAM-like [23] method with SIFT feature points, or a semi-dense method [13] with edge points. We derive the rough depth map RDM using

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

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 hill shape in the central area, while a non-corridor RDM has 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 typical corridor RDM shape and compute corridor probabilities. Finally, the sequence of single frame corridor probabilities is filtered by moving-average to compute architectural component probabilities for corridor α^2_t.

(a) (b)

Figure 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 puts further constraints on location estimation.

3.2. Virtual range finders

A virtual range finder (VRF) reading represents the distance to the nearest obstacle in its 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 center to the non-wall object. In Fig.5, an illustration of known free space with and without VRF is shown. It is uncertain that what is outside the free space, but inside the free space no wall should exist. In this way, VRF helps eliminating false particles more efficiently, which improves localization accuracy and convergence rate.

To compute VRF readings, we first calibrate the reconstructed point cloud coordinates. This procedure is performed as the original reconstructed point cloud coordinates are defined in coordinates relatively to the first frame, and the first frame has an arbitrary pose. We exploit the additional knowledge that the camera moves roughly on the same height level to calibrate the point cloud coordinates. We compute

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

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 $\boldsymbol{r}$. The point cloud coordinates are calibrated using the optimal rotation $\hat{\boldsymbol{r}}_t$ at every frame. We use the Levenberg-Marquardt Algorithm to solve the minimization problem.

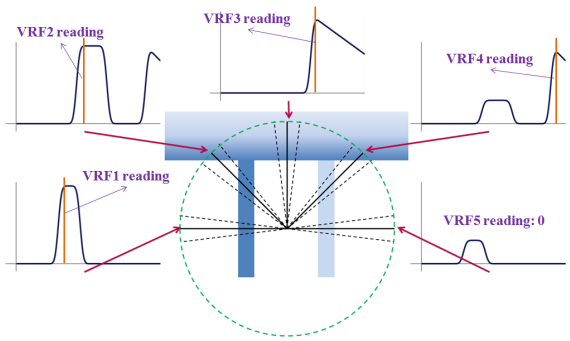


Figure 6: A toy example of computing VRF readings. Top-down view of point cloud and 5 VRFs. Blue shows point cloud density, deeper color indicates higher point cloud density. VRF detects the distance to the nearest dense point cloud. Zero is returned if VRF detects nothing within a certain distance.

The calibrated point cloud is compressed into 2D to form the 2D point density map. We use five VRFs pointing at left, left-forward, forward, right-forward, and right. The signal $vrf_t^j(d)$ denotes the point density values along the j-th VRF pointing direction. At last we compute VRF readings as

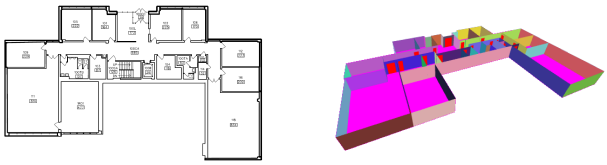
$$\beta_t^j = \begin{cases} \text{argmin}_d\{vrf_t^j(d) \geq c_d\} & \text{if } \max\{vrf_t^j(d)\} \geq c_d \\ 0 & \text{if } \max\{vrf_t^j(d)\} < c_d \end{cases} \tag{5}$$

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 distance. Fig.6 shows a toy example of computing VRF readings.

3.3. Matching 3D floorplan with visual input

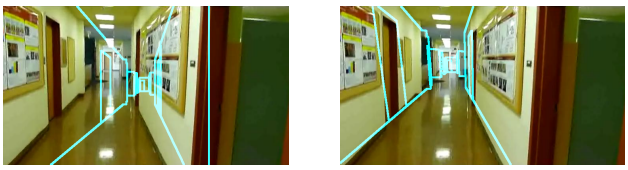
Floorplan shows the building's 2D structure. Assuming the room height and door height are known, the 2D floorplan can be lifted into a 3D floorplan model as shown in Fig.7. 3D floorplan is matched with visual input after localization convergence to further fine-tune the camera pose. After localization convergence, the camera pose is roughly correctly estimated. The goal of the 3D floorplan-visual input matching process 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 the observation of the input image. Thus we have

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



(a) (b)

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



(a) (b)

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

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.

3.4. Localization procedure

We use particle filter for the localization procedure. 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 $\{\beta_t^j\}$. Then for all particles, we propagate the camera motion, compute particle weights that characterises to how well each particle matches with current measurements, and resample all particles according to the weights. The weights are computed as

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

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

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 all past c_{cvg} frames. After convergence 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

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

where the variance $\sigma_{SLM,t}$ is computed from all poses measured during the optimization process. Finally, the whole 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.

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**
  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 $\hat{\boldsymbol{x}}_t$
  Set all weights $w_{t+1}^{(n)} = 1/N_{total}$
end for

Table 1: 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/°
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**

4. 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 included. 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.

To show the effectiveness of our method, we compare with 4 other methods: (1) The motion-only method (Mo) that only uses the measured camera motion for localization; (2) The best possible performance that state-of-the-art IMU-based localization method [24] can achieve, where we directly use the ground-truth trajectory as 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. Each method are tested by 10 trails. Results are shown in Fig.9 and Table 1.

From Fig.9 and Table 1, it can be seen that the proposed method outperforms all compared methods in two ways: 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 mo-

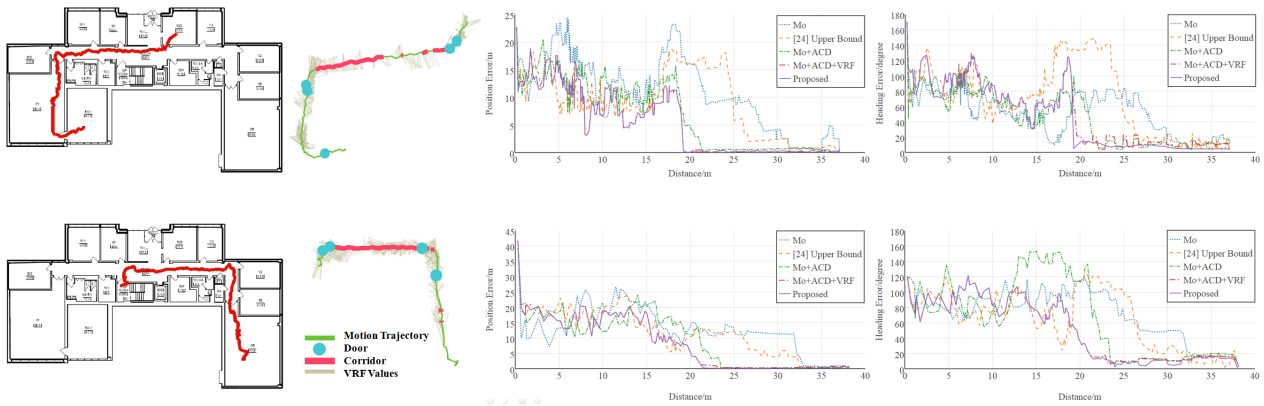


Figure 9: Experimental results of applying different algorithms to our dataset. First row and second row correspond to first and second video sequence. In each row, the first column shows the ground-truth trajectory. The second column shows the integration of measured camera motion, detected architectural components and estimated virtual range finder (VRF) readings. The third row shows the position error of $\{\hat{\boldsymbol{x}}_t\}$ in meter over total moving distance since the first frame. The last row shows the heading direction error of $\{\hat{\boldsymbol{x}}_t\}$ in degree over total moving distance since the first frame.

tion estimation. By using the video image data (Mo+ACD, Mo+ACD+VRF, and proposed), localization performance can be easily improved by a large margin.

In the second column of Fig.9, the integrated camera motion along with architectural component detection 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, the door detector is triggered when the camera passes two building corners. However, comparing Mo+ACD with Mo, it is revealed that the true positives of ACD improves localization performance more than its false positives harms the performance.

In the first video, 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. 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 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.

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.10, we show examples of input video frames and generated virtual 3D views. Please see the demonstration video in the supplemental material for more details.

5. Conclusion

In this paper, a method for visual indoor localization using a video sequence as input and a floorplan as prior was proposed. The proposed method uses 1) architectural component detection that detects doors and corridors with simple features, 2) virtual range finders that expands the free space, and 3) matching between the 3D floorplan and visual inputs that fine-tunes the camera pose, to improve localization performance significantly. 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. In *ICRA*, 2010. 3

[2] R. Anati, D. Scaramuzza, K. G. Derpanis, and K. Daniilidis. Robot localization using soft object detection. In *ICRA*, 2012. 3

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

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

[5] P. Bahl and V. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM*, 2000. 2

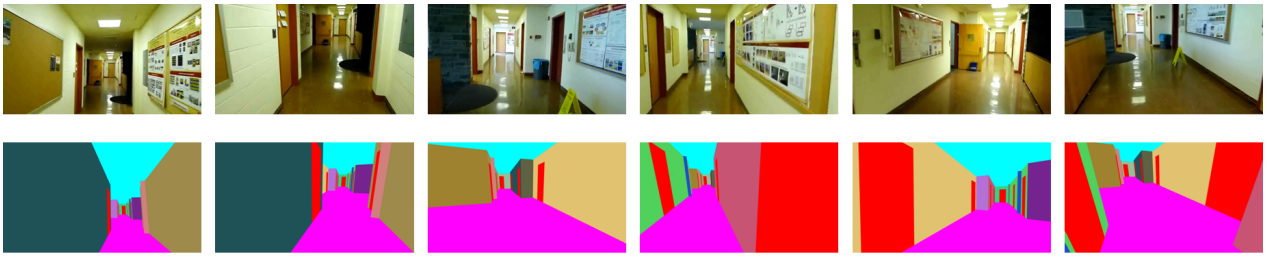


Figure 10: Example virtual 3D views produced by our method. The first row shows the current input video frame, the second row shows the virtual 3D view generated using the estimated camera pose after localization converges. The first to third columns show examples where pose is accurately estimated, the last three columns show failure cases where the 3D floorplan matching is interfered by irrelevant non-structural textures in the input image.

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

[7] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011. 4

[8] D. M. Chen, G. Baatz, et al. City-scale landmark identification on mobile devices. In *CVPR*, 2011. 2

[9] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor localization without the pain. In *International Conference on Mobile Computing and Networking*, 2010. 2

[10] J. Chung, M. Donahoe, et al. Indoor location sensing using geo-magnetism. In *International Conference on Mobile systems, Applications, and Services*, 2011. 2

[11] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 4

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

[13] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014. 3, 4, 6

[14] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *ICCV*, 2013. 3

[15] N. Ganganath and H. Leung. Mobile robot localization using odometry and kinect sensor. In *Emerging Signal Processing Applications*, 2012. 2

[16] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011. 3

[17] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[18] S. Hilsenbeck, A. Moller, R. Huitl, G. Schroth, M. Kranz, and E. Steinbach. Scale-preserving long-term visual odometry for indoor navigation. In *Indoor Positioning and Indoor Navigation*, 2012. 2

[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. In *ICRA*, 2014. 3

[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. 2

[21] M. Kaess, K. Ni, and F. Dellaert. Flow separation for fast and robust stereo odometry. In *ICRA*, 2009. 3

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

[23] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, IEEE and ACM International Symposium on (ISMAR)*, 2007. 4

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

[25] J. Lladós, E. Martí, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(10):1137–1143, 2001. 6

[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):198–214, 2011. 2

[27] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI*, 2002. 2

[28] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *CVPR*, 2004. 3

[29] D. Robertson and R. Cipolla. Building architectural models from many views using map constraints. In *ECCV*, 2002. 3

[30] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *IJRR*, 21(8):735–758, 2002. 2

[31] J. Straub, S. Hilsenbeck, et al. Fast relocalization for visual odometry using binary features. In *ICIP*, 2013. 2

[32] S. P. Tarzia, P. A. Dinda, et al. Indoor localization without infrastructure using the acoustic background spectrum. In *International Conference on Mobile systems, Applications, and Services*, 2011. 2

[33] X. Yang and Y. Tian. Robust door detection in unfamiliar environments by combining edge and corner features. In *CVPR Workshops*, 2010. 3

[34] M. Youssef and A. Agrawala. The horus wlan location determination system. In *International Conference on Mobile systems, Applications, and Services*, 2005. 2