SoFIT: Self-Orienting Camera Network for Floor Mapping and Indoor Tracking

Yanchen Liu, Jingping Nie, Stephen Xia, Jiajing Sun, Peter Wei and Xiaofan Jiang
Department of Electrical Engineering, Columbia University
{yl4189, jn2551, stephen.xia, js5504, pw2428}@columbia.edu, jiang@ee.columbia.edu

***Abstract*—We present SoFIT, an easily-deployed and privacy-preserving camera network system for occupant tracking. Unlike traditional camera network-based systems, SoFIT does not require a person to calibrate the network or provide real-world references. This enables anyone, including non-professionals, to install SoFIT. Once installed, SoFIT *automatically* localizes cameras within the network and generates the floor map leveraging movements of people using the space in daily life, before using the floor map and camera locations to track occupants throughout the environment. We demonstrate through a series of deployments that SoFIT can localize cameras with less than 4.8cm error, generate floor maps with 85% similarity to actual floor maps, and track occupants with less than 7.8cm error.**

***Index Terms*—Camera Network, Camera Self-Localization, Floor Map Generation, Indoor Tracking**

I. Introduction

Localization and tracking are important in many applications [1]–[4], including in the current COVID-19 pandemic to ensure that social distancing requirements are being met. Although there are many works that explore indoor and outdoor camera-based localization and tracking [5], most of them require a long setup and calibration phase. For example, a camera network-based localization system requires a person to, not only install the cameras, but also calibrate the system to the floor plan and location of cameras. There is also a privacy concern with using dense deployments of cameras.

We propose SoFIT, an *easily-deployed, low-cost, privacy-sensitive camera network system for indoor occupant tracking*. Unlike previous works, SoFIT completely automates and removes the need for people to manually specify camera positions, provide real-world references, and calibrate the system to the floor map of the environment. Installing SoFIT only requires placing cameras on the ceiling and turning them on, allowing even non-professionals to install SoFIT. SoFIT uses the movements of people going about daily life to automatically *self-localize* cameras, generate the *floor map* of the environment, and begin tracking occupants. Though a person could intentionally walk through the camera deployment to speed up the localization and floor map generation process, it is by no means necessary. Additionally, SoFIT performs most computations at the edge and does not transmit raw images to a third-party cloud server. SoFIT only transmits generated floor maps and locations of people to a cloud server rather than raw images. Our contributions are summarized next:

- We propose SoFIT, an easily-deployed camera network system for indoor occupant tracking, that requires no setup aside from placing cameras on the room ceiling.
- We introduce novel algorithms and architectures that allow SoFIT to *self-localize* cameras in the network and generate the *floor map* of the environment based solely on the movements of occupants without any additional information that needs to be supplied by the installer.
- We demonstrate that SoFIT can efficiently localize cameras within its network with less than 4.8cm error, generate floor maps with 85% accuracy, and begin tracking occupants with less than 7.8cm error in as quickly several seconds to several minutes after installation depending on occupant traffic patterns. We also show through simulations, using real data, that SoFIT can scale to tens of cameras without accumulating large amounts of error. SoFIT accomplishes all of this without requiring any additional information provided by a person.

II. Related Works

There are many existing works on occupant tracking, especially for human dense commercial areas [6]–[8]. Many of these use existing infrastructures, such as surveillance cameras. Non-camera based sensing systems may work well in outdoor scenarios [9], [10]. However, multi-path effects and moving human objects lead to poor performances. SoFIT provides a quick and easy way to set up inexpensive camera modules for occupant tracking over large areas.

A. Camera Network Self-Localization

Knowing the location and orientation of cameras in the real world is required for any camera network-based tracking system; this process is also known as extrinsic calibration. This either needs to be specified by the person installing the system or estimated. Existing works that perform extrinsic calibration often require an additional reference (e.g., a chessboard [11]) or additional sensors/actuators such as a moving robot [12]. Other works try to map corresponding objects observed in one camera to another camera using the SIFT feature [13]. However, SIFT features perform poorly in images with periodic or rotationally symmetric patterns, such as floor tiles in office spaces and labs.

B. Floor Plan Generation

To perform indoor localization and tracking a floor map of the environment is required; many works assume that

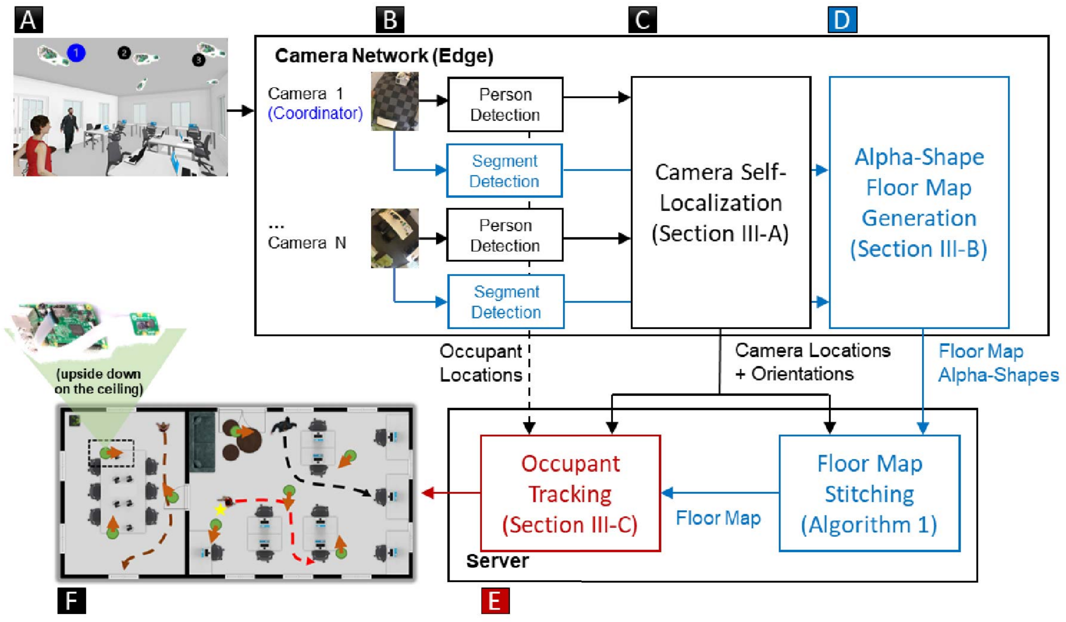

A
B
C
D
E
F
Camera Network (Edge)
Camera 1
(Coordinator)
Person Detection
Segment Detection
...
Camera N
Person Detection
Segment Detection
Camera Self-Localization (Section III-A)
Alpha-Shape Floor Map Generation (Section III-B)
(upside down on the ceiling)
Occupant Locations
Camera Locations + Orientations
Floor Map Alpha-Shapes
Occupant Tracking (Section III-C)
Floor Map
Floor Map Stitching (Algorithm 1)
Server

Fig. 1: SoFIT system overview. (A): Example set up of cameras. (B): Each edge device extracts presence of people and line segments that correspond to the floor map. (C): Using the movement of people between different views, SoFIT localizes each camera without any input from a person. (D): Each edge device extracts alpha-shapes corresponding to the portion of the floor map it can see, and SoFIT uses each camera's view of the floor to stitch together a full floor map. (E): SoFIT uses the localized cameras, generated floor map, and detected people to track occupants, an example of which is shown in (F).

the floor map is given or estimate it using camera-based methods [14], [15]. However, these methods often require an offline calibration phase or a user or robot to hold the camera and walk around to take images of different parts of the room. SoFIT does not require a person to generate a floor map and, instead, utilizes a novel algorithm to stitch together portions of the floor detected from each camera depending on the location of each camera.

III. System

Figure 1 shows the architecture of SoFIT. First, all cameras are placed on the ceiling, facing towards the floor. We place facing downwards and perpendicular to the floor, to reduce the effects of occlusions caused by furniture and to improve privacy; cameras pointing in this position capture less of a person's face than a camera mounted on a wall. Additionally, the cameras can be placed *semi-randomly*; the only requirement is that each camera is close enough to at least one other camera so that their field of view overlaps. This is to ensure that each camera, at some point, has the opportunity to observe at least one person at the same time as another camera to localize itself. Next, SoFIT analyzes the view of each camera, extracts floor map boundaries, and stitches them together as more cameras localize themselves to generate the floor map. Finally, SoFIT begins to track and map occupant locations onto the floor map. One assumption we make is that each

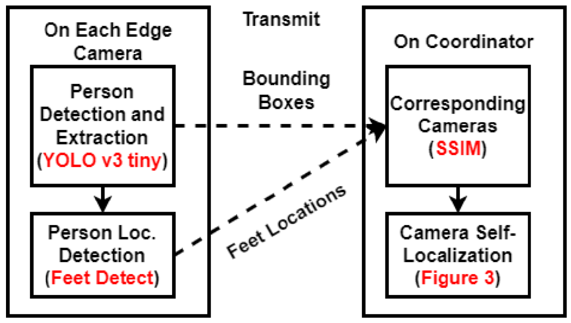

On Each Edge Camera
Person Detection and Extraction (YOLO v3 tiny)
Person Loc. Detection (Feet Detect)
Transmit
Bounding Boxes
Feet Locations
On Coordinator
Corresponding Cameras (SSIM)
Camera Self-Localization (Figure 3)

Fig. 2: SoFIT's logic flow for self-localizing cameras.

camera in the network is at the same height because many indoor environments have the same ceiling height throughout.

A. Camera Self-Localization

Once all cameras are installed, no additional information or setup is required. SoFIT begins by localizing all cameras relative to one another, as shown in Figure 2. One of the cameras is designated as the *coordinator*, which SoFIT uses as the reference to localize all other cameras. The idea behind enabling self-localizing cameras is to **identify moving occupants** in one camera that also appears in other cameras and use their movements to **estimate the location and orientations** of cameras relative to each other.

Identifying and corresponding occupants between different cameras: To use the movements of one person to

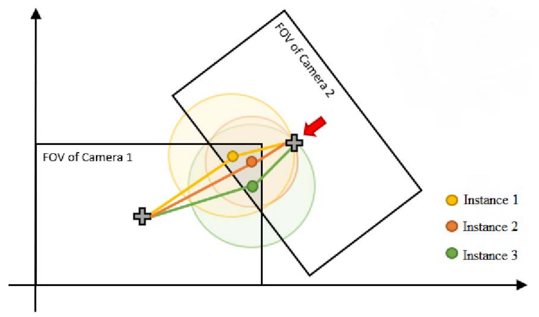


Fig. 3: Example of SoFIT localizing camera 1 with respect to camera 2. SoFIT utilizes at least three different locations of a person observed by both cameras to localize the cameras.

self-localize two cameras with respect to each other, SoFIT needs to first identify that two cameras are observing the same person. To accomplish this, we need to first *detect and extract* a person in each camera, and then confirm that the person captured by each camera *corresponds* to the same person.

To *detect and extract* all observed people, each camera module runs the YOLO real-time object detection deep neural network (DNN) to detect and extract the *bounding boxes* of the people present. Additionally, each camera module extracts the pixel coordinate *location* of each detected person. The location we detect depends on which part of the body we take the measurement (e.g., location of the head, torso, or feet). We decide to measure the location of a person by detecting the location of a person's *feet*. This is because in SoFIT, every camera is facing downward and perpendicular to the ground, which causes all vertical lines in the physical world (including a typical walking person), to converge the *vanishing point* of the camera's FOV. In these circumstances, the *vanishing point* is also the center of the camera's FOV. This also means that portions of a person's body that are closer to the ground (i.e., feet) will inevitably remain in the camera's frame at more locations than a body part that is higher above ground (i.e., head), so we choose to detect and track each person's feet.

To detect a person's feet, we take the bounding box of a detected person and use morphological transformations to find the contour of the person. Then, the feet is located at the point on the contour closest to the vanishing point of the view.

Next, each edge camera transmits only the *bounding boxes* and *locations* of each person to the coordinator. The coordinator then uses each detected person to find *corresponding* and overlapping cameras by finding people who appear in two cameras at the same time. To do this, we take bounding boxes of people observed from different cameras and compute the structural similarity index (SSIM) [16] to see if they correspond to the same person. Cameras that observed the same person at the same time will then use the movements of that person to localize themselves in the following step. We would like to note that SSIM only confirms whether the two bounding boxes of people supplied correspond to the same person and not the identity of the person.

Camera Self-Localization: Figure 3 demonstrates how the

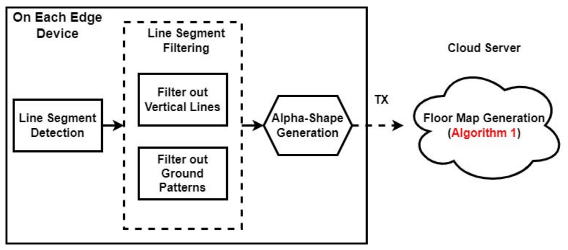


Fig. 4: SoFIT's logic flow for generating the floor map.

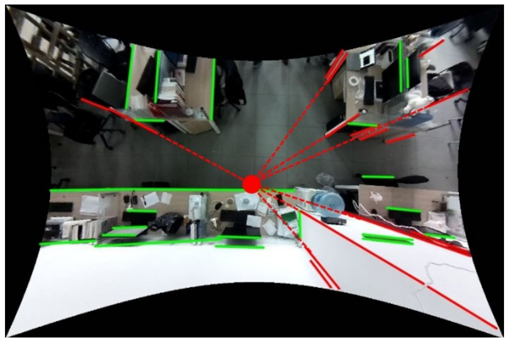


Fig. 5: An example of SoFIT finding floor map boundaries in one of its cameras by leveraging the vanishing point (red dot) to filter out lines that do not correspond to the floor map.

SoFIT coordinator finds the relative location of one camera (camera 2) with respect to another (camera 1). At each point where the same person is identified in both cameras, there is a limited number of positions where camera 2 could be located, which is along the circumference of the circle centered around the *location* of the detected person. To find the exact location, SoFIT needs to observe at least three consecutive frames where the person is at different locations; the location of camera 2 with respect to camera 1 is at the location where the three circles generated for each frame intersect. To improve robustness and confidence in our camera location estimates, SoFIT uses slightly more frames (nine). SoFIT continues to localize more cameras with respect to other cameras until every camera has been localized. The speed at which all cameras can be localized depends heavily on how much human traffic moves through each portion of the environment; though a person intentionally walking through the deployment could help localize each camera faster, it is also possible to allow SoFIT to localize cameras more naturally by observing the daily movements of people. Additionally, this procedure does not require cameras to know which camera views overlap; identifying people who appear at the same time in multiple views accomplishes this.

B. Floor Map Generation

Once cameras begin localizing themselves with respect to each other, SoFIT begins to analyze the FOV of localized cameras for lines and shapes that make up portions of the floor map and stitch them together. To accomplish this, SoFIT makes the assumption that most office space and apartments have rect-

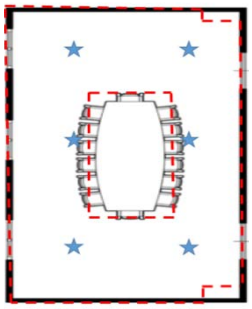
Algorithm 1: Floor Map Generation

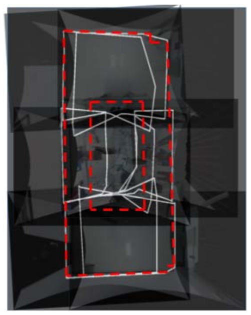
```
Input:
    $\mathbf{N_c}$ : number of cameras.
    $\mathbf{C} \in \mathcal{R}^{N_c \times N_c \times 2}$: camera location matrix.
    $\Theta \in \mathcal{R}^{N_c \times N_c}$: camera orientation matrix.
    $\mathbf{Map} \in \mathcal{R}^{N_c \times H \times W}$: alpha-shape matrix.
    ($H$ and $W$: height and width of the alpha-shape
    generated by each camera)
Output:
    $\mathbf{Map}_{all}$: integrated floor map (e.g., Figure 6(d))
1  Initialize $\mathbf{C}_{all} = []$, # all coordinates
2  $\Theta_{all} = []$, # all angles;
3  for $(i,j),\ \forall i,j \in \{1,\cdots,N_c\}$ do
4      $ref$ = the first not-NULL element in column $j$;
5      if $\mathbf{C}[i][j] != \mathbf{C}[ref][j]$ or $\Theta[i][j] != \Theta[ref][j]$ then
6          $(\Delta_x, \Delta_y) = \mathbf{C}[i][j] - \mathbf{C}[ref][j]$;
7          $\Delta_\theta = \Theta[i][j] - \Theta[ref][j]$;
8          for $k = 1, \cdots, N_c$ do
9              $\mathbf{C}[i][k] = \mathbf{C}[i][k] - (\Delta_x, \Delta_y)$;
10             $\Theta[i][k] = \Theta[i][k] - \Delta_\theta$;
11         end
12     end
13 end
14 for $i = 1, \cdots, N_c$ do
15     cover $\mathbf{Map}[i]$ on $\mathbf{Map}_{all}$ with center $\mathbf{C}[i]$ after rotating
         $\Theta[i]$;
16 end
17 Apply Canny filter to $\mathbf{Map}_{all}$ + post processing for
     denoising to obtain the boundary identifying the floor map
     (e.g., Figure 6(d)).
```

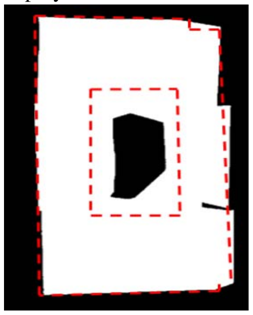
angular room structures with straight floor-wall boundaries; though this is not true for all environments, many buildings exhibit these characteristics.

Figure 4 shows the logic flow for generating the floor map. First, at each localized camera, SoFIT *detects* all straight lines in the its FOV using OpenCV's line segment detection (LSD) function. Next, SoFIT *filters* out all lines that do not correspond to the floor map.

There are two types of straight lines commonly observed that do not correspond to the floor map. First, are the lines that are *vertical* in the physical world because these correspond to lines on the wall or on furniture off the ground. As mentioned in Section III-A, all segments that are vertical in the physical world will tend towards the vanishing point or the center of the camera's FOV. Using this observation, SoFIT can easily filter out these types of lines. An example of this process is shown in Figure 5. The second type of straight lines are *ground patterns*, due to tiles, designs, or reflections on the ground. To filter out these lines, we use the observation a person's trajectory would never cross a line that is the floor map boundary (people generally do not walk on walls). This means if a person's trajectory intersects with a line, then that line is most likely not part of the floor map. Using this observation, SoFIT filters out all lines that intersect a person's trajectory, meaning a person walked through the pattern or tiles on the ground. Finally, we apply the alpha-shape algorithm [17] to interpolate between noisy and disjoint line segments to create the floor map or *alpha-shape* visible at each camera.


(a) Ground truth floor plan with deployed SoFIT cameras.


(b) White lines: Alpha-shapes generated by each edge device.


(c) White area: After stitching together alpha-shapes.

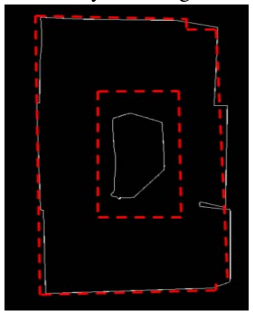

(d) White lines: The overall floor map after post-filtering.

Fig. 6: An example process of generating the overall floor map of an environment. Here, six cameras are placed throughout the environment, with their locations marked by blue stars in (a). The red lines indicate the ground truth floor map.

Next, only the alpha-shapes, which represent the portion of the floor map observed at each camera, are sent to a cloud server, where SoFIT stitches together the alpha-shapes generated from different cameras to generate the integrated floor map for the entire environment, display it to users, and use it to track occupants. The coordinates of the alpha-shape's vertices for each camera, i, are denoted as map_i.

To generate the floor map on the server, the first step is to create the camera location matrix ($\mathbf{C}$), by integrating locations from each camera and their neighbors:

$$\mathbf{C} = \bigcup_{i=1}^{N_c} \vec{\mathbf{X}}_{ij} \tag{1}$$

where j indicates all the cameras that overlap with camera i, N_c is the total number of cameras, and $\vec{\mathbf{X}}_{ij}$ is the location of camera i. We generate the orientation matrix as follows:

$$\Theta = \bigcup_{i=1}^{N_c} \phi_{ij} \tag{2}$$

ϕ_{ij} is the orientation for camera i. In addition, the server also computes the alpha-shape's vertices for each camera (map_i) into the overall floor map alpha-shape matrix ($\mathbf{Map}$). The overall camera location matrix ($\mathbf{C}$), the overall camera orientation matrix (Θ), and the overall floor map alpha-shape

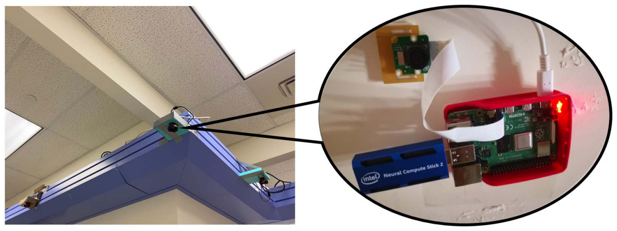


Fig. 7: Left: An example of a deployment of SoFIT with three cameras. Right: One Raspberry Pi-based camera module with a wide angle lens, and the NCS2 acceleration hardware.

matrix (**Map**) are the inputs for Algorithm 1 to stitch together the alpha-shapes floor maps from each camera to generated the entire integrated floor map.

Figure 6(b) shows the alpha-shapes generated from each camera overlaid onto an image of the floor map of the environment. Figure 6(c) shows the overall shape generated after stitching together all alpha-shape boundaries generated at each camera in white. SoFIT then applies Canny filtering to remove artifact noise and generate the final floor map, as shown in green in Figure 6(d).

C. Real Time Tracking

Once the floor map has been generated and stitched together for localized cameras, SoFIT begins tracking occupants. The entire floor map does not need to be generated for SoFIT to begin tracking occupants; SoFIT will begin tracking areas where cameras have been localized and the floor map is available. SoFIT will begin to track more areas as more cameras become localized.

To track individuals, each camera in SoFIT utilizes the same YOLO person detection DNN and SSIM metric to confirm the same person in two frames, from Section III-A. Again, we find the location of each person by detecting the location of their feet and mapping this location onto the floor map.

D. Finding Real-World Measurements

Every method presented in this section generates measurements, locations, and floor maps in pixels rather than absolute distances; this is because SoFIT has no real-world length reference. To allow SoFIT to map all measurements into absolute distances, we decided to estimate the height of each camera from the floor, using the FRCN deep neural network (DNN) [18]. This network takes as input one image and attempts to estimate the depth by generating a second image that would likely be observed if a second lens was nearby to enable stereo vision. The generated image and the input image are then used to estimate depth and height of the camera from the ground. Since we assume that all cameras are at the same height, SoFIT averages all of the height estimates from all cameras in the network. This height measurement is then used to map all relative pixel-based measurements, locations, and floor maps into real-world and absolute measurements.

TABLE I: Price breakdown for one camera module.

Module	**Unit Price** [U$]
Raspberry Pi 4	55
Pi Camera with Wide Angle Lens	20
Intel Neural Compute Stick 2	70
total	145

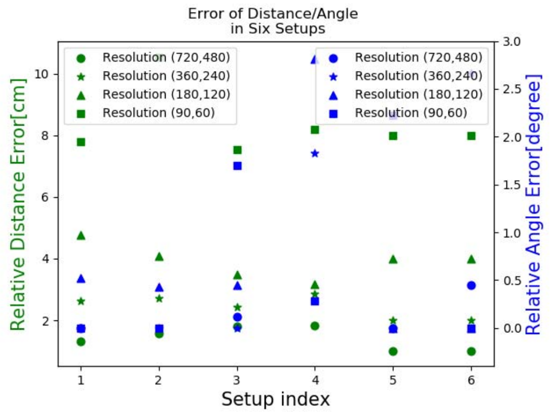


Fig. 8: Average camera localization error across six configurations across all three environments (Green: error of camera location (cm), Blue: error of orientation (degree)).

IV. Implementation

Hardware Implementation: As shown in Figure 7, we implement the SoFIT camera modules on the Raspberry Pi. To enable the use of the *YOLO* object detection DNN, we incorporate an Intel Neural Compute Stick 2 (NCS2) onto each module. However, we found that SoFIT could only run at 2.5 fps even after incorporating an NCS2. As such, we adopted a lighter version, *YOLO v3 tiny* [19], that enabled SoFIT to run at 18 fps. 18 fps is close to the frame rate of TV shows and movies (24 fps) and was fast enough to enable good performance in all aspects of SoFIT, as we show in Section V.

Expanding the Field of View: We added a wide-angle lens to each camera module to increase the FOV. This introduces distortion, especially at the edge of each frame. To correct this, we integrate an algorithm adapted from the MATLAB toolbox SWARD_Toolbox_2.2 [20], [21]. This algorithm has a one-time calibration procedure that can be performed during production, well before installing the cameras on site.

Cost: Table I shows the price breakdown of each camera module in SoFIT, which is inexpensive and under 150 USD.

V. Evaluation

We deployed a network of five camera modules in three different environments. The environments we deployed SoFIT could be inscribed in a rectangular prism with dimensions (width by length by height) 4W by 2L by 2.5H, 6.5W by 4.5L by 3H, and 8W by 4L by 3.5H (all in meters). Two of the environments are highlighted in Figure 11. In each deployment, we deployed the cameras in six different semi-random configurations. We had one person move around the environment in each configuration, simulating movements in

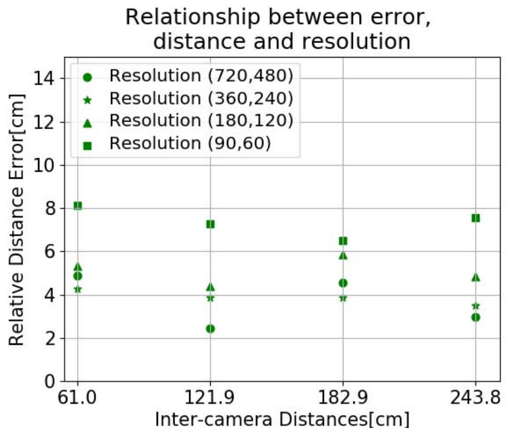


Fig. 9: Camera localization error vs. the average inter-camera distance. We see that even as the average distance between cameras increases, the localization error remains consistent.

daily life, until all cameras were localized and the floor map was fully generated. For all configurations, we also evaluated SoFIT under four different camera resolutions (90 by 60, 180 by 120, 360 by 240, and 720 by 480) to see how well SoFIT performs if we reduce the resolution to improve privacy.

SoFIT utilizes several open-source models to perform parts of its pipeline. First, it uses the *YOLO v3 tiny* object detection neural network for person detection; we retrain this network specifically for person detection using a dataset we collected containing over 1,500 images in various environments, achieving a 89.0% precision and 91.0% recall.

Second, SoFIT utilizes the *FCRN* to estimate the height of the cameras and uses it as the reference to map all pixel and relative measurements to real-world measurements. We trained *FCRN* using the KITTI odometry dataset [22]. Throughout our deployments, SoFIT estimates the height of the environment with an error of 6.3cm on average and a standard deviation of 1.1cm, which is small enough compared to the heights of the environments we deployed SoFIT.

Third, SoFIT utilizes the *SSIM* metric to confirm whether people detected in two camera views correspond to the same person. Using this metric, we created a classifier with a 89% accuracy on a 400 sample dataset extracted from images from our deployment. Using these three models, we evaluate the performance of SoFIT's camera localization, floor map generation, occupant tracking performance.

A. *Camera Network Self Localization*

Figure 8 shows the average estimated error in camera location and orientation in each of SoFIT in our six configurations and across three environments with differing camera resolutions. We see that as resolution decreases, the error increases, as expected. However, even when cameras are only 90 by 60 pixels, the average localization error is at most 8cm, which is small compared to our deployment area. Using the highest resolution (720 by 480), SoFIT achieves just 4.6cm camera localization error. Thus, we select 90 by 60 as the camera resolution used in the rest of this subsection.

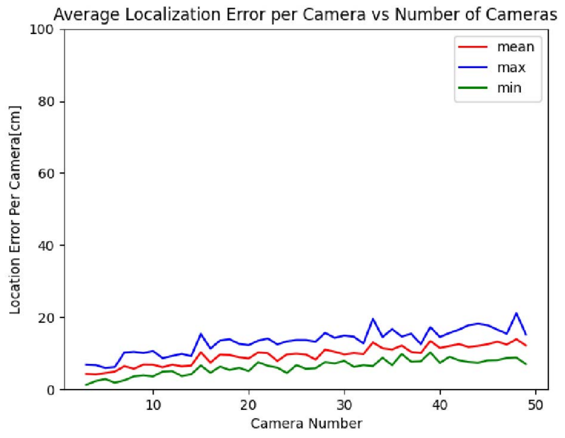


Fig. 10: Max, average, and min localization error per camera vs. number of cameras. We see that the error remains relatively constant even as the number of cameras in the network increases, showing that SoFIT scales gracefully.

To better understand how well SoFIT localizes cameras within its network, we looked at how the average error in estimated camera locations changes as the average distance between cameras increased (error vs. average camera distance) and conducted a simulation to see how well SoFIT could localize cameras *at scale*.

Localization Error vs. Average Camera Distance: Figure 9 shows the average localization error across all our environments with respect to the average distance between cameras. As the distance between cameras increases, the overlapping FOV between cameras decreases, which could lead to decreased localization accuracy. However, we see that as long as SoFIT fully converges, the localization error remains consistent across inter-camera distances.

At Scale: To quantify how well SoFIT runs at scale, we ran a simulation to see if errors in camera localization will accumulate at a high rate if the number of cameras in the network increases. We simulate a 4m by 2m area with 2.5m height and place N semi-randomly in the area such that each camera's FOV overlaps with at least one other camera in the network. We simulate errors in camera localization by adding a random offset. This offset is drawn from a Gaussian distribution centered around the average and variance in inter-camera errors observed between two cameras across all of our deployments, which is 4.8cm. Figure 10 plots the average error in localization of each camera in the network as the number of cameras, 50, increases. We see that even though the average error increases slightly as the number of cameras increases, the average error in camera localization is less than 20cm even when there are up to 50 cameras. This shows that the effect of accumulating errors across a large number of cameras is very small and that SoFIT scales well to large camera deployments.

B. *Floor Map Estimation*

Figure 11 shows examples of floor maps generated from two of the environments at two resolutions (720 by 480 and 360

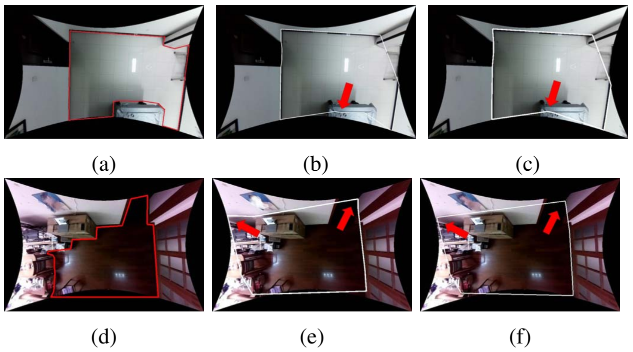


Fig. 11: Example of floor maps generated by SoFIT for two environments. (a)&(d) are the ground truth floor maps, (b)&(e) are the floor maps generated by SoFIT using higher resolution cameras (720 by 480), and (c)&(f) were floor maps generated using lower resolution cameras (360 by 240). Areas highlighted in red refer to areas of the generated floor maps that saw a reduction in accuracy after reducing camera resolution due to higher impact of lighting and occlusions.

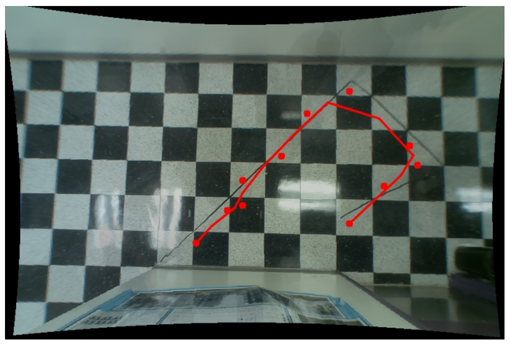
by 240) where we deployed SoFIT, highlighting areas where higher resolution cameras make a difference. This is because higher resolution allows SoFIT to better account for lighting and occlusions that are caused from furniture lying throughout the environments. Overall, SoFIT produced floor maps with an average of 85% intersection over union. This means that 85% of the floor maps generated across all scenarios overlapped in area and is consistent with the ground truth floor map.

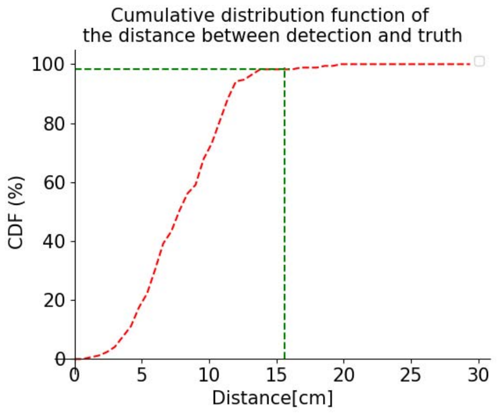
C. Indoor Tracking

After the floor map is generated for each scenario, we evaluate how well SoFIT can localize occupants. In each environment and configuration, we instructed five participants to walk a predetermined path. Figure 12(a) shows one of these paths in green and the path that SoFIT estimated in red. Figure 12(b) shows the cumulative distribution function (CDF) of occupant localization across all environments and configurations. We see that 98.25% of errors are less than 15.6cm, showing that SoFIT can accurately track occupants after self-localizing cameras and generating a floor map.

D. Convergence

One concern about SoFIT is how fast it can localize cameras, generate a floor map, and begin tracking occupants. This depends primarily on whether or not a person walks into the FOV of the cameras, which depends on the traffic patterns of the environment and which areas people are in throughout the day. Instead, we measure how long it takes for SoFIT to localize one camera after one person comes into the FOV of that camera since if no one moves within the FOV of the camera, that camera will never be localized. Figure 13 shows the CDF distribution of these convergence times. We see that 98% of the time, SoFIT takes less than 4.5 seconds to localize a camera after a person walks into the FOV. It is not unreasonable for a person to be moving in the FOV of a camera for this much time, meaning SoFIT can successfully localize a camera within the network with a high success rate as long as there is traffic in that area.


(a)



(b)

Fig. 12: (a) An example of one trajectory of a person estimated by SoFIT (red) compared with the ground truth trajectory (green). (b) CDF of occupant location errors across all configurations and all three environments we deployed SoFIT.

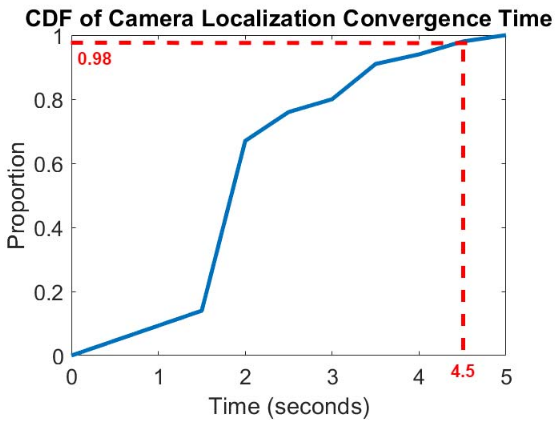


Fig. 13: CDF of the time it takes for SoFIT to localize one camera in the network once one person enters its FOV.

VI. Discussion and Future Work

Camera placement restrictions. Currently, SoFIT requires all cameras to be at the same height and facing down. We aim to generalize our methods to allow cameras to be placed on any surface with any orientation.

Floor map shape restrictions. SoFIT assumes that the floor map of indoor environments are mainly composed of straight lines. Though this is the case for many buildings, buildings with curved floor maps (e.g., a circle or dome) would pose a significant challenge for SoFIT in its current state. We plan to expand SoFIT's floor map generation pipeline to accommodate buildings with floor plans with arbitrary shapes and characteristics.

Beyond moving occupants. There are many objects and changes, besides a moving occupant, that we can potentially leverage to improve camera localization or floor map generation. For instance, we can potentially leverage the changes in shadows due to changes in lighting throughout the day to help localize cameras in areas with little human traffic. We plan to explore more avenues in the future.

Reducing energy consumption. Currently, we implement our camera modules and algorithms on a Raspberry Pi, which consumes a total of 11 Watts of power. In future work, we aim to reduce power consumption by moving to lower energy processing units (e.g., microcontrollers) and utilizing lower resolution cameras, as we have shown that SoFIT can still perform robustly even if we reduce camera resolution to improve privacy.

VII. Conclusion

We present SoFIT, an easily-deployed and privacy-aware camera network system for indoor tracking. Compared to existing camera network-based tracking systems, SoFIT does not require a user to calibrate or provide any real-world references, allowing SoFIT to be easily and quickly installed in almost any indoor setting. SoFIT automatically localizes cameras, generates a floor map of the environment and begins tracking all using the everyday movements of occupants in the environment. We demonstrate in real deployments that SoFIT can localize cameras within its network with less than 4.8cm error, generate floor maps with 85% accuracy, and track occupants with less than 7.8cm error.

VIII. Acknowledgements

This research was partially supported by the National Science Foundation under Grant Numbers CNS-1704899, CNS-1815274, CNS-11943396, and CNS-1837022. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Columbia University, NSF, or the U.S. Government or any of its agencies.

References

[1] S. Xia, D. de Godoy Peixoto, B. Islam, M. T. Islam, S. Nirjon, P. R. Kinget, and X. Jiang, "Improving pedestrian safety in cities using intelligent wearable systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7497–7514, 2019.

[2] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A survey of enabling technologies for network localization, tracking, and navigation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3607–3644, 2018.

[3] S. Xia, J. Nie, and X. Jiang, "Csafe: An intelligent audio wearable platform for improving construction worker safety in urban environments," in *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, ser. IPSN '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 207–221. [Online]. Available: https://doi.org/10.1145/3412382.3458267

[4] F. Viani, P. Rocca, G. Oliveri, D. Trinchero, and A. Massa, "Localization, tracking, and imaging of targets in wireless sensor networks: An invited review," *Radio Science*, vol. 46, no. 05, pp. 1–12, 2011.

[5] P. Wei, H. Shi, J. Yang, J. Qian, Y. Ji, and X. Jiang, "City-scale vehicle tracking and traffic flow estimation using low frame-rate traffic cameras," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, ser. UbiComp/ISWC '19 Adjunct. New York, NY, USA: Association for Computing Machinery, 2019, p. 602–610. [Online]. Available: https://doi.org/10.1145/3341162.3349336

[6] P. Wei, X. Chen, J. Vega, S. Xia, R. Chandrasekaran, and X. Jiang, "eprints: a real-time and scalable system for fair apportionment and tracking of personal energy footprints in commercial buildings," in *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, 2017, pp. 1–10.

[7] H. Zou, H. Jiang, J. Yang, L. Xie, and C. Spanos, "Non-intrusive occupancy sensing in commercial buildings," *Energy and Buildings*, vol. 154, pp. 633–643, 2017.

[8] P. Wei, S. Xia, R. Chen, J. Qian, C. Li, and X. Jiang, "A deep-reinforcement-learning-based recommender system for occupant-driven energy optimization in commercial buildings," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6402–6413, 2020.

[9] A. N. Parks, A. P. Sample, Y. Zhao, and J. R. Smith, "A wireless sensing platform utilizing ambient rf energy," in *2013 IEEE Topical Conference on Biomedical Wireless Technologies, Networks, and Sensing Systems*. IEEE, 2013, pp. 154–156.

[10] D. de Godoy, B. Islam, S. Xia, M. T. Islam, R. Chandrasekaran, Y.-C. Chen, S. Nirjon, P. R. Kinget, and X. Jiang, "Paws: A wearable acoustic system for pedestrian safety," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 237–248.

[11] Q. Sun and D. Xu, "Self-calibration of multi-camera networks without feature correspondence between different cameras," *Optik*, vol. 125, no. 13, pp. 3331–3336, 2014.

[12] J. Liu, T. Wark, S. Martin, P. Corke, and M. D'Souza, "Distributed object tracking with robot and disjoint camera networks," pp. 380–383, 2011.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] R. Ayyalasomayajula, A. Arun, C. Wu, S. Sharma, A. R. Sethi, D. Vasisht, and D. Bharadia, "Deep learning based wireless localization for indoor navigation," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[15] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, "Layoutnet: Reconstructing the 3d room layout from a single rgb image," pp. 2051–2059, 2018.

[16] J. Nilsson and T. Akenine-Möller, "Understanding ssim," 2020.

[17] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983.

[18] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 239–248.

[19] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[20] X. Ying, X. Mei, S. Yang, G. Wang, and H. Zha, "Radial distortion correction from a single image of a planar calibration pattern using convex optimization," pp. 3440–3443, 2014.

[21] X. Ying, X. Mei, S. Yang, G. Wang, J. Rong, and H. Zha, "Imposing differential constraints on radial distortion correction," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 384–398.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.