Preprint – to be published in ICCHP 2020

An Indoor Navigation App using Computer Vision and Sign Recognition

Giovanni Fusco[1[0000-0001-9437-6954]] , Seyed Ali Cheraghi[1[0000-0001-8720-1291]] , Leo Neat[2] and James M. Coughlan[1[0000-0003-2775-4083]]

[1] The Smith-Kettlewell Eye Research Institute, San Francisco, CA
{giofusco, ali.cheraghi, coughlan}@ski.org
[2] Department of Computer Science and Engineering, University of California, Santa Cruz, CA
leosneat@gmail.com

Abstract. Indoor navigation is a major challenge for people with visual impairments, who often lack access to visual cues such as informational signs, landmarks and structural features that people with normal vision rely on for wayfinding. Building on our recent work on a computer vision-based localization approach that runs in real time on a smartphone, we describe an accessible wayfinding iOS app we have created that provides turn-by-turn directions to a desired destination. The localization approach combines dead reckoning obtained using visual-inertial odometry (VIO) with information about the user's location in the environment from informational sign detections and map constraints. We explain how we estimate the user's distance from Exit signs appearing in the image, describe new improvements in the sign detection and range estimation algorithms, and outline our algorithm for determining appropriate turn-by-turn directions.

Keywords: Navigation, Wayfinding, Accessibility, Visual Impairment, Blindness, Low Vision.

1 State of the Art and Related Technology

The key to wayfinding tools is localization – a means of estimating and tracking a person's location as they travel in an environment. The most widespread localization approach is GPS, which enables a variety of wayfinding tools such as Google Maps and BlindSquare, but it is only accurate outdoors. There are a range of indoor localization approaches, including Bluetooth beacons [1], Wi-Fi triangulation[1] and RFIDs [7]. However, all of these approaches incur the cost of installing and maintaining physical infrastructure, or of updating the system as the existing infrastructure changes (e.g., whenever Wi-Fi access points change). Dead reckoning approaches such as step counting using inertial navigation [3] can estimate relative movements without any physical infrastructure, but this tracking estimate drifts over time unless it is augmented by absolute location estimates.

[1] https://techcrunch.com/2017/12/14/apple-maps-gets-indoor-mapping-for-more-than-30-airports/

Computer vision is a promising localization approach, but most past work in this area has either required special hardware [9] or the use of detailed 3D models of the environment [8] that are time-consuming to generate and make the approach vulnerable to superficial environmental changes (e.g., new carpeting or moved shelves). The iOS app Clew [15] uses visual-inertial odometry (VIO) [10], a function built into modern smartphones, to perform dead reckoning, which requires no model of the environment. However, while dead reckoning allows a blind user to retrace their steps from a destination they have already reached back to their starting point, on its own it doesn't provide guidance to a new destination, and does not provide absolute localization. To overcome these limitations we developed an indoor localization system [4] that combines computer vision-based recognition of barcodes posted on walls of the environment for absolute location information with VIO to track movements between barcode detections. The new version of our localization system [5] is a self-contained iOS app that recognizes standard Exit signs, eliminating the need for barcodes. In this paper we describe enhancements to this system, including the development of an accessible navigation function that provides turn-by-turn directions to guide the user to a desired destination.

2 Overall Approach

This section summarizes our localization approach. The sections that follow it describe the specific contributions of this paper: improved sign recognition, improved range estimation from a sign detection, and turn-by-turn navigation directions.

Our localization approach is described in detail in [5]. It combines three main ingredients to estimate the user's location in an indoor environment and track it over time: (a) A 2D floor plan (map), annotated with the locations of walls and other impassable barriers, locations of interest such as rooms, elevators and stairwells, and the locations of informational signs such as Exit signs. (b) A sign recognition algorithm, such as the one we have implemented [6] for standard Exit signs, combined with an algorithm that estimates the distance to the sign from its appearance in the image and its known physical size. (c) A dead reckoning algorithm that estimates the user's relative movements, even when no signs are visible. We use the iOS ARKit's[2] built-in visual-inertial odometry (VIO) function, which combines computer vision and inertial sensing, to perform dead reckoning.

Our localization algorithm combines the three ingredients as follows. It uses a particle filter (a standard tool used in robotics [14]) to maintain multiple hypotheses ("particles") of location and bearing (i.e., the direction the smartphone camera is facing relative to the map) over time. The particle filter integrates multiple sources of information, and after some time it converges to an estimate of the location and bearing – even if the algorithm is initialized with no knowledge other than the specific floor the user is on. These information sources include distance estimates obtained from sign detections, VIO estimates of the user's movements (available even when no signs are

[2] https://developer.apple.com/augmented-reality/

visible), and two constraints: traversability (the fact that the smartphone can't move through walls) and visibility (the smartphone camera can't see a sign through a wall).

We implemented our localization algorithm as a real-time app running on an iPhone 8. Studies with blind participants [5] show that the app is accessible to blind users, who need only hold the smartphone camera straight ahead while walking, rather than having to aim the camera at specific signs (which would be challenging for people with low or no vision). Alternatively, the user can "wear" the smartphone to point the camera straight ahead, using a lanyard, shirt pocket or other means. The median localization error in our studies was shown to be approximately 1 meter or less, which is more than adequate for typical wayfinding applications.

3 Sign Recognition and Range Estimation Algorithms

The localization algorithm [5] currently used in our iOS app uses an Exit sign recognition algorithm that we developed previously [6]. This algorithm is a fast AdaBoost [12] (short for "Adaptive Boosting", a machine learning technique for combining multiple forms of evidence into a single more reliable decision) cascade-based approach that processes a single VGA frame on the smartphone in about 7 msec, and returns an approximate bounding box (Fig. 1a) for each detected Exit sign. While our previous algorithm for estimating the distance to the Exit sign (see below) requires only knowledge of the Exit sign centroid in the image (which can be obtained from even an approximate bounding box), our new algorithm depends on having at least a rough segmentation of the Exit sign – in other words, knowledge of the Exit sign boundaries in the image.

Accordingly, we have explored other sign recognition algorithms that return detailed segmentations instead of a rough bounding box. The algorithm that has proved most promising so far is a deep learning one called U-Net [11], using MobileNet 2 as a backbone to facilitate a mobile implementation[3,4]. Since the standard U-Net implementation processes small images of resolution 224 x 224, we have devised a multi-scale procedure that first detects any Exit signs at coarse scale, zooms in on the part of the image centered on the detected Exit sign and then applies U-Net again on that part of the image to segment it accurately. We are in the process of integrating the U-Net algorithm with our iOS app, which will require efforts to optimize its speed for real-time iOS performance.

Previously [5] we estimated the distance to the Exit sign using a simple calculation that assumes knowledge of camera height above the ground and the height of the Exit sign above the ground. The range (distance to the Exit sign) was estimated using the apparent elevation of the detected sign in the image: the closer the sign appears to the horizon, the farther away it is. While this approach is effective, it is inconvenient to have to obtain the camera and Exit sign heights above the ground. The camera height depends not only on the user's height but on how they hold the smartphone, which also

[3] https://medium.com/vitalify-asia/real-time-deep-learning-in-mobile-application-25cf601a8976
[4] https://github.com/akirasosa/mobile-semantic-segmentation

varies over time; moreover, the Exit sign heights may vary from sign to sign, even within the same floor of a building.

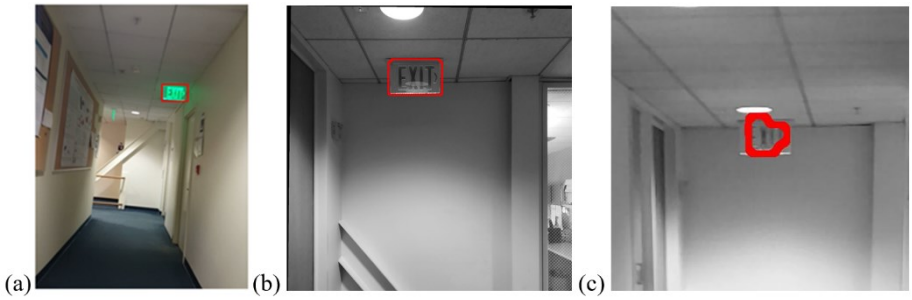

Fig. 1. Sign recognition algorithms. (a) Original Adaboost approach [6] returns a rough bounding box around the Exit sign. (b) U-Net algorithm shows an accurate segmentation border (red pixels). (c) More distant view of Exit sign, zoomed in for clarity, shows a noisy segmentation by U-Net.

Instead we decided to estimate the range using the apparent size of the sign in the image, which eliminates the need for knowledge of the camera or sign height above the ground. A standard computer vision approach is to use a pose estimation algorithm such as PnP [13]. For a rectangular sign, this requires knowledge of the physical height and width of the sign, and the pixel locations of all four sign corners in the image. However, getting accurate location estimates of the sign corners is difficult without a precise segmentation (Fig. 1b) – and segmentations are often noisy in images acquired under real-world conditions, such as when the sign is viewed at a distance (Fig. 1c) or when the image quality is degraded by motion blur.

Thus, we have devised a range estimation algorithm that is effective even when the segmentation is approximate. Our approach relies on three key assumptions:

1. The sign is rectangular, with a known physical height (e.g., in cm).
2. It is mounted so that the sign lies in a vertical plane, with the borders of the sign horizontal or vertical with respect to gravity.
3. The camera pitch (angle that the camera line of sight makes with respect to the horizontal plane) and roll (the angle the camera is rotated about its line of sight, with 0° and 90° corresponding to portrait and landscape orientations, respectively) are known. This enables us to estimate the horizon line and the apparent angle of the sign above the horizon, which is a key measurement in our range estimate.

Fortunately, these assumptions are satisfied for our application. Exit signs are rectangular, with a standard size, and they are almost always mounted in a way that satisfies assumption 2. Moreover, the camera pitch and roll are estimated in real time on modern smartphones using the built-in inertial measurement unit (IMU).

We have derived a simple formula for calculating the sign range in terms of known and measured quantities. Briefly, instead of requiring the pixel coordinates of the four corners of the sign in the image, the formula requires only an estimate of the apparent

height of the sign in pixels and its location in the image. This is an important advantage over the PnP approach, since it is possible to estimate the apparent height even when the segmentation is too blurry to clearly identify the four sign corners.

Next we describe this formula in detail. Given our assumptions, we can transform the raw image acquired by the camera into an *unrolled* image, which is an image that is rotated to undo the effects of any non-zero roll (i.e., rotation about the camera line of sight). This means that the horizon line appears horizontal in the unrolled image. Then we can analyze the scene geometry of the sign relative to the camera in 2D (Fig. 2), in which the Y axis represents the vertical axis (with respect to gravity) and the Z axis represents the horizontal axis. Given a column of pixels in the image plane, this sweeps out a vertical plane (shown in the figure) that intersects the camera center and a vertical slice of the sign. (Different image columns give rise to different slices of the sign, and thus slightly different ranges Z, but this discrepancy is minimal at typical sign viewing distances.)

We denote the camera pitch by τ, the angle between the bottom pixel of the sign and the center row of the image (the camera line of sight direction corresponds to the pixel in the center of the image) by α, and the angle subtended by the sign (from its bottom pixel to top pixel) by δ. (It is straightforward to estimate α and δ from the pixel locations of the corresponding image features using the camera focal length and a simple pinhole model.) The bottom of the sign is height H above the ground (assumed unknown), and the physical height of the sign is h_0 (known). From trigonometry we have:

$$\tan(\tau + \alpha) = H/Z \tag{1}$$

and

$$\tan(\tau + \alpha + \delta) = (H+h_0)/Z \tag{2}$$

Combining both equations allows us to solve directly for Z and H in terms of known quantities, leading to this range estimate equation:

$$Z = h_0/[\tan(\tau + \alpha + \delta) - \tan(\tau + \alpha)] \tag{3}$$

Given a segmented sign (e.g., Fig. 1c), which is obtained for an unrolled image, we analyze multiple vertical slices of the segmented sign to arrive at a single robust estimate of α and another of δ. This is done by calculating the height (top row – bottom row) of each slice, and calculating the median height over all the slices; calculating the median of the bottom row over all the slices; and then converting the median height and bottom values into an overall estimate of α and δ.

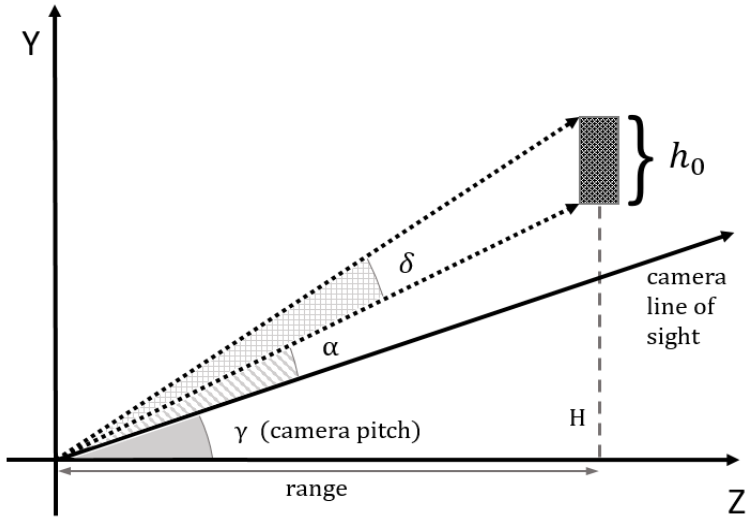

Fig. 2. We can estimate the range Z to the sign in terms of easily measured quantities: the camera pitch ɤ, the angle α between the bottom pixel of the sign and the center of the image (which corresponds to the camera line of sight), the angle δ subtended by the sign (from its bottom pixel to top pixel) and the physical height h_0 of the sign. (The height of the sign H above the ground is assumed unknown.)

Table 1 presents experimental results describing the accuracy of various range estimation algorithms, obtained using 852 Exit sign images taken in the Smith-Kettlewell building. The top row indicates the actual range of the Exit sign (measured with a tape measure) and the next three rows show results for each of three methods for detecting and segmenting the sign: VGA uses the approximate bounding box reported in [5], HR refers to the same approach applied to a higher resolution image (1920x1440) and U-Net refers to the new multi-scale approach detailed above. The accuracy is reported as the median percent range estimation error, where the median is taken across all detected signs in a distance category, and the percent range estimation error is defined as $|e\text{-}a|/a$ (expressed as a percentage), where e = estimated distance and a = actual distance. N/A indicates that no Exit signs were detectable for that algorithm and distance. (Of the 852 Exit sign images, there are 74 false negatives, in which the U-Net recognition algorithm fails to detect an Exit sign and for which no range estimate is available; these cases, most of which occur at distances of 7 m and greater, are excluded from this error analysis.) Note that the U-Net approach almost always yields the smallest median errors, even at distances as far as 9 m.

We expect that our new range estimation algorithm will improve our localization results, both in terms of accuracy and the time needed to arrive at an accurate localization after the wayfinding app is first launched. These improvements will be augmented in the future as we continue to improve the underlying Exit sign recognition algorithm, and extend this algorithm to other types of signs.

Table 1. Median percent range estimation error, categorized by actual distance to sign. See text for details.

Actual	2 m	3 m	4 m	5 m	6 m	7 m	8 m	9 m
VGA	9.1%	14.7%	20.8%	11.0%	11.8%	25.0%	31.5%	N/A
HR	9.2%	13.8%	14.1%	11.4%	10.7%	15.5%	17.7%	13.3%
U-Net	9.2%	7.7%	9.3%	8.9%	5.6%	8.0%	8.5%	9.1%

4 Turn-by-Turn Navigation Directions

This section describes the turn-by-turn navigation directions that build on our localization algorithm, which are verbal directions that guide a visually impaired traveler in real time to a desired destination. Navigation directions are necessary for transforming a localization algorithm into a fully accessible wayfinding app.

These turn-by-turn directions assume that the walkable area of an indoor environment is described by a graph (Fig. 3) whose nodes are either (a) points of interest (POIs) or (b) *control points* where the traveler either must turn or has multiple turning directions to choose from. The graph, including nodes and edges connecting neighboring nodes, is embedded in the (x,y) coordinate system used by the localization algorithm to identify locations on the map. Given a destination node and current location estimate (x,y), we first "snap" the location (x,y) to the closest point (x',y') on the graph; note that this point (x',y') lies either on a graph edge or (in rare cases) a graph node. Next we use a standard Dijkstra shortest-path algorithm 2] to determine the shortest path in the graph to the destination. Our app issues directions such as "turn left" shortly before the suggested action should be executed, for instance, when the snapped location (x',y') is 0.75 meters before the control nodes where a left turn should be taken.

The app allows the user to specify their starting location from a pull-down menu, or to indicate the current floor if the starting location is unknown. The desired destination is selected from a second pull-down menu. Note that the app issues no directions other than "start walking" until it is able to estimate the user's location. If the user begins heading in the wrong direction (e.g., the wrong direction down a corridor), the app signals this path deviation with haptic feedback. The app continuously updates the shortest path to the destination, which means that re-routing is automatically performed as needed. Guidance is provided until the destination is reached, including directions announcing where the destination is relative to the user as they approach within 1 meter and also announcing whether the desired destination is behind a door.

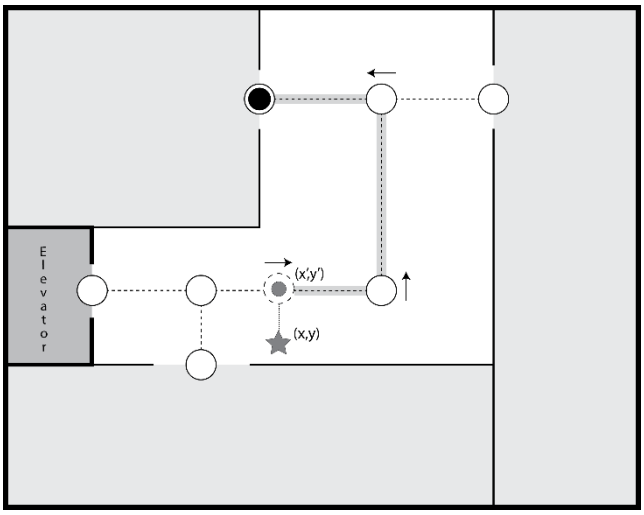

Fig. 3. 2D floor plan (map) of indoor area showing a graph of the walkable area defined by nodes and vertices. In this example, the destination node is filled in black, and the user's current location (x,y), indicated by a star icon, is snapped to the nearest location (x',y') along the graph. The shortest path to the destination is shaded in gray, with arrows indicating the directions to turn at each node on the path.

The graph representation we are currently using is optimized for environments dominated by corridors and other narrow paths; in the future we will explore appropriate ways of representing walkable areas in large open spaces, such as shopping centers or airports, and of communicating appropriate verbal or non-verbal directions in these spaces.

5 Conclusions and Future Work

We have described enhancements to our previous indoor localization approach, including improved Exit sign segmentation, a more effective method for estimating the range of a sign, and a navigation app that provides turn-by-turn directions to a desired destination. We have begun informal testing on the navigation app by two visually impaired participants, which demonstrates the app's accessibility.

We will conduct systematic tests of the app with more visually impaired participants as soon as this is possible, and will expand our testing to include multiple buildings. Future work will focus on optimizing the user interface to communicate information in a timely fashion without overwhelming the user with unnecessary feedback. We will also test and refine an algorithm that uses the built-in smartphone barometer to estimate floor changes, allowing the app to provide guidance in an entire building. Our sign recognition algorithm will need to be optimized for real-time use in the app, and we will explore ways of recognizing multiple sign types (such as restroom signs and room number signs) of any shape, size or appearance. Finally, we will experiment with

the persistent Augmented Reality capabilities that have recently been added to ARKit[5] and ARCore, which allow an app to create and save a 3D model of an environment and use it later for localizing the camera (effectively a SLAM-based approach), and which may be useful for handling wide open spaces.

6 Acknowledgments

GF and JMC were supported by NIH grant 1R01EY029033 and NIDILRR grant 90RE5024-01-00. LN was supported by NIH grant 1R01EY029033 and SAC was supported by Smith-Kettlewell's CV Starr Fellowship. We thank Dr. Roberto Manduchi at UC Santa Cruz for helpful discussions about sign recognition.

References

1. D. Ahmetovic, C. Gleason, K. Kitani, H. Takagi & C. Asakawa. (2016). NavCog: turn-by-turn smartphone navigation assistant for people with visual impairments or blindness. Web for All Conference. ACM.
2. E.W. Dijkstra. (1959). A note on two problems in connexion with graphs. Numerische Mathematik. 1 (1959), 269-271.
3. G. Flores & R. Manduchi. (2018). Easy Return: An App for Indoor Backtracking Assistance. CHI 2018.
4. G. Fusco & J. Coughlan. (2018). "Indoor Localization using Computer Vision and Visual-Inertial Odometry." 16th Int'l Conf. on Computers Helping People with Special Needs (ICCHP '18). Linz, Austria. July 2018.
5. G. Fusco & J. Coughlan. "Indoor Localization for Visually Impaired Travelers Using Computer Vision on a Smartphone." 17th International Web for All Conference (2020). Taipei, Taiwan.
6. G. Fusco, E. Tekin, & J. Coughlan. (2016). Sign Finder Application -Technical Report. https://www.ski.org/sign-finder-application-technical-report
7. A. Ganz, S. R. Gandhi, C. Wilson & G. Mullett. (2010). INSIGHT: RFID and Bluetooth enabled automated space for the blind and visually impaired. 2010 Int'l Conf. of the IEEE Engineering in Medicine and Biology.
8. C. Gleason, A. Guo, G. Laput, K. Kitani & J.P. Bigham. (2016). VizMap: Accessible visual information through crowdsourced map reconstruction. ASSETS 2016.
9. F. Hu, Z. Zhu, and J. Zhang, (2014). Mobile Panoramic Vision for Assisting the Blind via Indexing and Localization. Second Workshop on Assistive Computer Vision and Robotics, in conjunction with ECCV 2014.
10. J. Kelly & G.S. Sukhatme. (2011). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. The International Journal of Robotics Research, 30(1), 56-79.
11. O. Ronneberger, P. Fischer, & T. Brox. (2015). U-net: Convolutional networks for biomedical image segmentation. Int'l Conf. on Medical image computing and computer-assisted intervention. Springer, Cham.

[5] https://developer.apple.com/documentation/arkit/saving_and_loading_world_data

12. Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. Machine learning, 37(3), 297-336.
13. R. Szeliski. (2010).Computer vision: algorithms and applications. Springer Science & Business Media.
14. S. Thrun, W. Burgard, & D. Fox. Probabilistic Robotics. (2005). Massachusetts Institute of Technology.
15. C. Yoon, R. Louie, J. Ryan, M.K. Vu, H. Bang, W. Derksen, & Paul Ruvolo. (2019). Leveraging Augmented Reality to Create Apps for People with Visual Disabilities: A Case Study in Indoor Navigation. ASSETS 2019.