An Indoor Positioning System Facilitated by Computer Vision

Avi Cooper
Torah Academy of Bergen County (HS)
Wireless Information Network Laboratory (WINLAB)
Rutgers University
New Brunswick, New Jersey
Coopera2017@tabc.org

Poojit Hegde
Middlesex County Academy for Science, Mathematics, and Engineering Technologies (HS)
Wireless Information Network Laboratory (WINLAB)
Rutgers University
New Brunswick, New Jersey
phalpha1729@gmail.com

Abstract—The purpose of this paper is to present a revolutionary method which allows for high accuracy location determination of users inside an indoor. Global positioning has changed the way in which we interact with our specific locations on a real time basis, as can be seen most prominently in mapping applications. However, global positioning is severely limited indoors where location is equally important and requires greater reliability and pinpoint accuracy. While there have been attempts to create new technologies to allow for indoor positioning, they are still lacking. Thus we developed an easily implementable system able to achieve previously unattainable performance metrics. We use low cost webcams and a series of algorithms to detect people in a video frame, and then identify and position them. Accuracy for identification is upwards of 95% and positioning accuracy is within a foot for the majority of the frame of view, all while running in real time on mobile CPUs. Such a system can be implemented on large scales to allow for exciting new applications; indoor directions in malls and public transportation hubs, new forms of human-robot interactions and consumer habit analysis in stores are all now possible.

I. INTRODUCTION

The Global Positioning System, (GPS) developed first by the United States Navy and later scaled up by the Department of Defense in the 1970s [1], allows for positioning of a GPS receiver anywhere in the world. This positioning is, in the worst case scenario, accurate to 7.8 meters with 95% accuracy, but is typically much more accurate, on the scale of less than five meters [2]. However, due to limitations on the physical properties of the radio waves transmitted by GPS satellites, the signal can become too noisy indoors. At times the signal may even be nonexistent. Even disregarding such adverse circumstances, the region of interest indoors is much smaller. Five-meter accuracy may be sufficient to assume that a user is on one roadway or another, but indoors, there may be several hallways and turns inside a five meter squared area. Therefore, an Indoor Positioning System (IPS) must remain accurate within distances a full order of magnitude less.

Previous attempts to implement IPS have utilized various positioning technologies including WiFi [3], as well as Bluetooth Low Energy (BLE) [4]. The WiFi solution involves multiple access points as well as a mobile application. For each WiFi access point, the received signal strength indication (RSSI) and the media access control (MAC) address are analyzed, and based on this data, an algorithm calculates current position. This method has limitations, as it requires a sustainable and extensive WiFi infrastructure, while accuracy still remains quite low, ranging from five to fifteen meters. The Bluetooth solution, on the other hand, trades WiFi access points for Bluetooth beacons. These beacons send signals that are either analyzed through a phone-based application or through server-based tracking for position determination. While BLE has higher accuracy than the WiFi method, range is limited and it requires additional hardware. Though both these existing methods, along with other solutions, contain their benefits, scalability and accessibility are severely compromised. Moreover, previous attempts at indoor positioning have almost always relied on user cooperation, with the use of an application on a hand-held device for client based solutions. They do not allow the tracking of a person who is not publishing their location data. Finally, they often require extensive setup procedures, such as requiring a person to walk around with a cart to observe signal strengths in each room.

Our project aims to overcome these shortcomings and offer a practical, simplistic method to not only position users, but to also identify them and monitor their location over time. Our goal is to offer a solution to IPS that is both reliable and easily scalable for many different scenarios.

The major motivation for improvements in indoor positioning stems from the potential of possible applications. These applications are wide ranging, extending from the original use of GPS for real time mapping applications, now possible indoors, to human robot interactions. New visitors to public common areas, such as large malls and public transportation hubs, often have difficulty locating specific destinations, whether they are certain stores or gates. Maps, if available at all, are far and few between, and often times quite confusing. IPS will do for this scenario what GPS did for outdoor directions; create a vastly simpler user experience, while increasing efficiency and reducing total transportation time. In contrast, a totally different user may use the exact
same infrastructure in a very different way. While the previous application requires that the system know the person’s identity to find them in a database, if each user is to remain anonymous then the operator of the implemented IPS can still track users as they move through a public space. For example, a mall owner may analyze the behaviors of the users of the mall, and identify which stairways, entrances and exits are being used inefficiently, as well as identify which stores receive the most traffic, and at what times. One final example application is a new way for humans and robots to interact. As personal robots become more ubiquitous, pinpoint positioning accuracy of users will become ever more important. For example, delivery robots will be able to bring packages directly to their destinations without any person taking the time to tell them where to go. They simply need the command to bring package ‘x’ to person ‘y’, much in the same way that one would tell a person to do a similar task.

II. DESIGN OVERVIEW

In an attempt to create a system that was built for scalability, we took hardware and cost to implement limitations into careful consideration. While there are ways to detect people in a video stream using very specialized hardware, such as the Xbox 360 Kinect, we determined that this method would be infeasible for large-scale usage, as each unit can cost upwards of several hundred dollars. In addition, proprietary devices could create limited functionality and limit room for development. For example, while the Kinect does have the ability to identify users in its view and even segment their bodies into crude joint- skeletons, it is limited in the number of users it can identify. Closed source solutions pose limitations such as this, along with the very real possibility of discontinuation, as has proven true with the Kinect. Thus we aimed to create an open source system that is limited simply by the end user’s hardware capabilities. A simple stereoscopic camera and computer, along with the software we have developed, are sufficient for implementing IPS. Accuracy and capability can be improved with higher resolution cameras and faster CPUs and GPUs.

For our experimental design, we used two Axis M1054 Network Cameras which allowed us to access and record video remotely. We assigned local IP addresses so that they could be accessed anywhere from inside WINLAB (Rutgers University.) The cameras have a maximum resolution of 1280x800 pixels with a horizontal angle of view of 84 degrees, and video- frame rates of 25 frames per second [5]. Two cameras are needed for stereocopy, allowing us to calculate depth, and thus eventually calculate distance and position of people in an image. This will be further discussed later in the paper.

Our software relies heavily on the capabilities of OpenCV, an open-source library rich in computer vision programming functions [6]. The development platform for this project was in Ubuntu, an open-source Linux operating system. The majority of the code for our software is written in the form of Python scripts, with the addition of some C++ executables, all of which can be found at a public GitHub repository here [12]. We also frequently use NumPy, an extension to the Python programming language that allows for large matrix manipulations and more versatile mathematical functionalities. To filter and represent the data collected from our identification and location algorithms, we use MATLAB.

Our high level design consists of multiple elements, which act dynamically in the overall functionality of our project. The design is depicted in a flowchart in Figure 1. Firstly, we access video streams from the cameras. For each frame, we process the image. This involves identifying persons in the frame through background subtraction, and optionally undistorting images using camera calibration. Then, we run the identification and location algorithms in parallel, so that the all the data is processed as quickly as possible. We then synthesize the identification and location data. Finally, we publish this data to a server to allow for user access.

III. IMAGE PROCESSING

In this step of the algorithm we take a captured frame from the video stream and process it to isolate areas of the image that contain people. This processing is based on the assumption that typically, the main objects moving within a frame will be people. Almost everything else will be stationary for the vast majority of the time. Thus we use an algorithm that is able to do background subtraction, which removes pixels that have remained constant over a set amount of frames [7]. After running this algorithm, we now have an image with changed pixels marked in white and constant pixels marked in black. Due to the imperfect nature of the cameras there is noise in this resulting image, so we run several iterations of erosion and then dilation on the clusters of white pixels. This removes almost all of the noise while maintaining the shape of the larger pixel clusters, indicating where there is a person in the frame. We then bind proximally close clusters with a contour, a bounding polygon indicating where in the image there is a detected person. This contour, along with a simpler bounding rectangle of that contour, are returned for use in the user identification and location calculation steps. For a pictorial representation, see Figure 2.

IV. USER IDENTIFICATION

With the bounding contour from the previous subprocess, we are now able to identify the people in the image. The first step in this process is to run a feature detection algorithm on the cropped image of the person. This type of algorithm detects changes in average pixel sub-crops, allowing for the detection of changes in brightness and color. This data is indicative of the edges and corners of certain regions, which, on a higher level, can describe the overall properties of an image, including where and how large certain shapes and similar areas are [13]. This data can be quite useful when trying to identify individuals. For example, a person with a checkered shirt will have a lot of specific data points, each marking the abrupt transition between colored areas. The
signature of these data points will be very different from another person who is wearing a mono-colored shirt. There are several algorithms supplied by OpenCV for this task, and we ended up using the Speeded Up Robust Features (SURF) algorithm because it consistently provided the most key points. Even though its developers own it, necessitating buying a license when using it commercially, it is free to use for researchers [14].

The data points from the feature detection must now be prepared for use as inputs to the neural network. One limitation of neural networks in general is that they have a predefined number of inputs. However, the data points from the feature detection can be quite variable, ranging from in the tens to several hundred or thousand. Thus, to create the necessary set number of inputs, we use the clustering algorithm K-Means Clustering, which has the ability to identify clusters of data and group them together. This is done by randomly placing a predetermined number of points and then finding the average distance between these test points with the random data. The test points are then moved to reduce average distance to all points near them, and after a few iterations, are able to migrate to the center of the data clusters [8]. At the conclusion of this step, we have descriptions of the average of the data points in each of the set number of clusters, allowing for input into a neural network.

We finally use a neural network to map these inputs to outputs, which are in turn correlated to users of the system. We used a standard model for this last step, implementing it by calling the provided neural network-machine-learning library provided by OpenCV [9].

V. CALCULATING LOCATION

Once we have located where a person is in a two dimensional image, we then calculate their position in the three dimensional room. To do so, we utilized stereoscopy, a method that allows for the perception of three dimensions using two cameras side by side [10]. In our design, we designated one of the two Axis M1054 cameras as the left camera and the other as the right camera. Then we positioned them six feet apart on the ceiling, ensuring that they were parallel to one another and perpendicular to the segment connecting them. We also measured the angle of declination between the ceiling and the cameras as 21°.

Our algorithm to calculate location involves the use of the angles between the focal axes of the cameras and the location of the person on the two dimensional image. Since we know the resolution of our image and the horizontal angle of view, we can then find the number of degrees per horizontal pixel in the image [11]. The horizontal angle between the focal axes (represented by the center of the image), and the actual person can be calculated by determining the number of horizontal pixels between the center of the camera and the center of the bounding box containing the person (derived during the image processing step.) Using these angles and the distance between the two cameras, it is possible to determine the rest of the triangle's measurements with the angle side angle (ASA) method. We then calculate the length of the altitude drawn from the person to the segment connecting the two cameras. However, these distances are incorrect as they are on an arbitrary plane not parallel to the ground due to the downward angle of the camera lenses. The desired triangle with the distances from the cameras to the person is the previously calculated triangle projected onto the floor. Again, using some trigonometry, we are able to determine the measurements of this new triangle and are finally able to calculate the coordinate of the person.

Figure 3A and Figure 3B both display a view of the person being located, and indicate the coordinate location of the person at a specific frame. Figure 2A is the image from the left camera, and Figure 2B is the image from the right camera. The origin, (0,0), is the midpoint of the segment connecting the two cameras projected on to the floor. The first number in the coordinate pair represents the distance in feet that the person is to the right of the origin, and the second number represents the distance in feet that the person is forwards from the origin. Figure 4A shows the coordinate location of a person over time, and Figure 4B shows the filtered version of this graph. This type of plot of the information serves as metadata to give a better overall representation of the position of the person over time.

VI. API DESIGN

In creating the Application Programming Interface (API) for this project, we designed it to be scalable, easily implementable and modular. Each component was designed to be completely independent, save for one data structure, referred to as a Region Of Interest (ROI), which is shared in between functions. This modularity allows for the testing of each core function separately, or for the stringing together of all the core functions to create a customizable implementation of the full IPS. This modularity lends itself to scalability and easy implementation. Each function was written to work on an arbitrary number of inputs, which in turn allows for an arbitrary scale factor. Refer to the GitHub Repository here [12].

VII. LIMITATIONS AND FUTURE WORK

Though the project was not only successful but exceeded expectations, there are limitations of our design and of our results that must be put into consideration. With more work and improvements, many of these limitations can be reduced or even eliminated.

Firstly, no neural network is perfect, limiting identification accuracy. That being said, perfection is not necessary, as an accuracy around 95% (which was achieved) is more than adequate. However, we have not tested the network in extreme conditions, such as low lighting, objects being moved around, or a large number of people in the area or in the database. However, even considering faultiness in the
identification process, large errors can easily be removed with filtering. We can assume that a person will not jump more than a certain number of feet between frames, so if the computer incorrectly identifies an individual, we can check to see if this position makes sense based on previous located positions. To improve accuracy in the future, more factors may be fed into the neural network input, aside from those derived in feature detection like clothing colors and patterns.

Camera properties also present difficulties. If the identified person is near the middle of the frame, the calculated distance is very accurate, to within a foot. However, as the person moves towards the edges of the frame, the accuracy decreases due to radial distortion from the cameras. Camera calibration for the individual cameras helps to get rid of some of the distortion, but better stereo calibration would help improve the calculations towards the edges. Also, considering other methods for accounting for barrel and pincushion distortion may prove to be more reliable. Perhaps purposefully barrel-distorting an image with pincushion distortion would negate the distortion altogether, or vice versa. Another limitation inherent to the cameras is poor camera resolution. As an individual moves away from the camera, they take up fewer of the pixels, limiting accuracy. The number of features decreases, as well as the number of pixels available to differentiate between the angles created between the person and the cameras.

In the future we may also consider the inclusion of more than two cameras. This may allow for more precise triangulation. The position of the cameras could drastically change position calculation accuracy, so an experiment to find the optimal spacing and positioning of the cameras would be ideal.

VIII. CONCLUSION

Overall, this project was successful. We were able to design a new and near-complete indoor positioning system. The methodology involved is very different from preexisting ones to solve the problem of indoor positioning. Clearly, there are challenges and limitations that will have to be overcome before the implementation described in this paper can be put into practical use. However, there is great promise for the use of machine learning and computer vision for achieving indoor positioning. Hopefully in the future, IPS can become a term as ubiquitous as GPS.

ACKNOWLEDGMENTS

We would like to thank the Liberty Science Center Partners in Science Program, and its coordinators Ruben Rosario and Deepa Shah, for giving us new opportunities to parake in academic research as well as finding mentors for us to learn from. We would also like to thank the summer internship program at WINLAB for creating an environment that allowed us to succeed in this project. Finally, we would like to specifically thank Professors Ivan Seskar and Richard Martin of WINLAB for their guidance thoughout the project.

REFERENCES
