Activity-Based Indoor Localization with Smartphones

Richard Teammco
The University of Texas at Austin
teammco@cs.utexas.edu

William Xie
The University of Texas at Austin
wxie@cs.utexas.edu

ABSTRACT

Global positioning system is inaccurate for indoor localization, and the growing need for a reliable indoor positioning system (IPS) has yet to be filled. We propose a new technique for indoor localization using activity recognition from motion data. Using only a standard smartphone app, we collect the user's motion data from the phone's onboard accelerometer, gyroscope, and compass. This data is then sent to a server for processing and classification. The classifier decides what activity the person is engaged in (e.g. walking, sitting, etc.) and this information is used as a sensation for a particle filter based localization system. Combined with a simple odometry and turning detector, we are able to correctly localize users inside a university building. This can be useful for many applications, but we are specifically interested in exploring its effectiveness in an office setting and working towards an intelligent building.
A demo of this work in simulation can be found here:
https://youtu.be/gFB34H4qNEM.

1. INTRODUCTION

Robots are becoming increasingly capable with recent advancements in artificial intelligence. They are now able to provide services to and cohabit in the complex environment people live in. In order best serve people's needs, it is important for robots to be aware of where people are. For outdoor environments, global positioning system (GPS) is an adequate solution. However, as soon as a person steps inside a building, GPS can no longer correctly locate the person due to the lack of line of sight communication with the satellites. Interference from building materials and electromagnetic waves from competing devices also further deteriorate its effectiveness. In an indoor environment, there is yet to exist a localization technique that could allow easy positioning without incurring infrastructural overhead or require the use of specialized hardware.

We investigate a localization method through activity recognition using accelerometer, gyroscope, and compass data. Since smartphones have become ubiquitous, most people always carry one around in their pocket. This placement of the phone when not under active use allows onboard motion sensors to capture a rich representation of the person's movements. When the person is indoor, activities such as walking, climbing the stairs, opening doors, and riding the elevator are a few landmarks that could greatly help to localize a person. When paired with a traditional localization algorithm such as a particle filter, a system could probabilistically narrow down some most likely locations. To our knowledge, this is the first such attempt to derive a person's location solely on based on their activity.

For this project, we implemented the following:

1. A smartphone app that can collect and export accelerometer, gyroscope, compass, and GPS data.
2. A convolutional neural network based activity classifier that categorizes segmented accelerometer data into one of six classes.
3. A localization tool that uses a particle filter to predict the user's location.

2. RELATED WORKS

2.1 Indoor Localization

The goal of indoor localization, or an indoor positioning system (IPS), is to accurately and precisely localize a person inside a building where GPS is unavailable. Much research has been done in recent years in an attempt to reliably solve this problem; however, all of these methods have practical limitations that must be addressed if this technology is to be applied to real-world situations.

The most common type of indoor localization uses WiFi signal strength to estimate the location of the user's device. These techniques use the known location of WiFi access points and their transmission power to compute an estimate of the user's position. While these methods can be highly accurate [3], they rely on predetermined awareness of access point locations and their respective signal strengths. This data is typically collected and updated by a time-consuming process called *war driving* [4]. However, WiFi localization is not always reliable. Radio signals are affected by noise and

interference, and access points can be moved, go offline, or change their signal strength. Also, some methods require specialized hardware, whereas our approach only uses commercially available smartphones. Furthermore, many platforms, including Apple's iOS, do not allow applications to access link-layer network information due to security concerns. This makes WiFi localization limiting on consumer devices without extensive modifications.

Person re-identification can also be used for indoor localization. This approach uses computer vision in order to detect and re-identify the same people with a set of cameras scattered throughout the environment [2]. There is an explicit fixed cost behind the premise of this problem: the indoor environment must be equipped with enough cameras to fully cover the entire building. Due to the cost and infrastructure constraints, person re-identification for localization can only be used in a small subset of domains. In addition, there are inherent challenges in vision systems such as object detection, recognition, and handling occlusions. These are still open research problems. Furthermore, the performance for state of the art methods for re-identification can only achieve a rank 1 result of below 0.7 on commonly available evaluation datasets [1]. This is still far cry from the precision needed for good localization.

The most promising results have been demonstrated by recent work in magnetic positioning, which can be accurate up to 1 meter [5]. A user's location can be determined by mapping distortions in the Earth's magnetic field caused by metals building structures. However, these methods require special hardware. Also, alterations of the environment over time (e.g., repositioning metallic furniture) degrade the system's performance and substantially lower the accuracy.

2.2 Motion-Based Activity Classification

Accelerometer data has been used extensively for activity recognition [11, 12, 7]. Previous approaches have shown promising results to categorize human activities based on the common, three-axis accelerometer alone, especially when paired with deep neural network feature extraction techniques [15]. However, this research field often focuses on human condition and sensing rather than localization. [6, 8] incorporated an accelerometer in localization, but their approaches are catered towards approximating the speed of the agent. They did not take into account the possibility of different types of movements other than walking. The closest work to our approach is described in [10]. They combined both WiFi signal strength and motion sensor data from commercial smartphones to provide the most likely estimate of a robot's position using a particle filter. However, they only estimate movement speed and heading to increase the accuracy of WiFi localization, and they only cover a single floor scenario. Our approach does not rely on WiFi. Instead, we use activity classification as a major component to identify landmark locations in the building in order to localize a person across multiple floors.

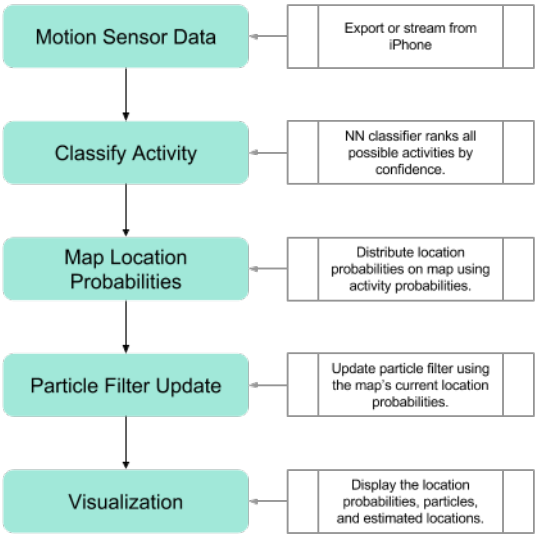

Figure 1: The data flow of our method. Motion data is first collected on a smartphone and sent to the localization server. This data is then classified by a convolutional neural network, which outputs a set of activity probabilities. These activities are mapped to locations on the building map (e.g., the activity of walking up stairs would be mapped to a staircase). A particle filter then uses these new location probabilities to update the particles for that time step.

3. METHODS

In this section, we break down the steps we took to achieve accurate localization of a user with only a smartphone's motion sensors. An overview of our method is shown in Figure 1.

3.1 Collecting Motion Data

We collect motion data from a user's smartphone when it is placed in their pocket. Our app periodically samples the device's accelerometer, gyroscope, and compass sensors, and streams the data over WiFi or a cellular data connection to a localization server where the data is processedand classified to determine the user's current activity. It is also possible to collect motion data from the phone while it is in the user's hands or in different pose configurations. We could further exploit the surge in popularity of wearable devices such as smart watches, which also have motion sensors built in. However, this is beyond the scope of our project.

3.2 Activity Classification

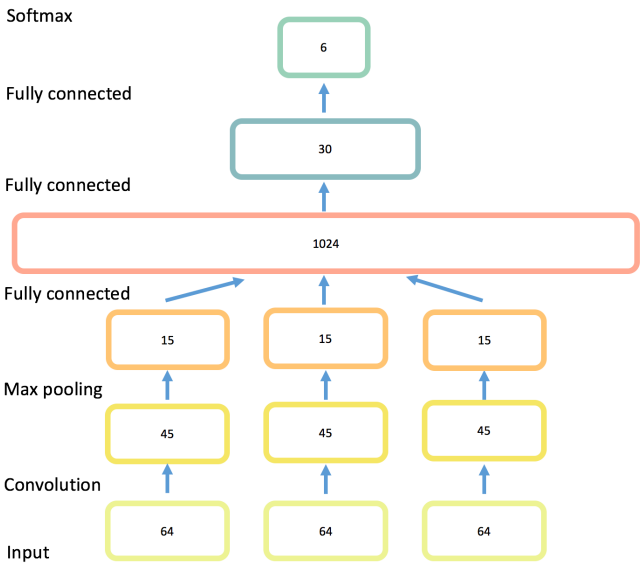

Figure 2: Our CNN architecture consists of convolution, max pooling, fully connected, and softmax layers. The output dimention of each layer is specified in each box.

We use a convolutional neural network (CNN) for feature extraction and classification. The input data is divided into samples consist of N adjacent data points (with each data point containing the x, y, z sensory information from the accelerometer). We allow for overlap or shared data points between neighboring samples.

Exploiting the consistent relative values of the data points for each activity, the convolution layer of the CNN performs a 1-D convolution of the sample. A filter of size m, with $m < N$, slides across the sample. At each position, the filter is convolved with the input producing an output. This captures the local dependency of the data. The resulting output size is $N-m+1$. The output of the convolution layer is then fed to the max-pooling layer. The main purpose of this layer is to reduce the sensitivity of the output making it scale invariant as well as adding nonlinearity. As the name suggests, max-pooling divides the input into non-overlapping bins and selects the maximum value from each. The max-pooling outputs from each of the three accelerometer channels are concatenated and followed by two fully connected layers. These layers strips the temporal relationship of the data and focus on producing a classification. Every output node is connected to every input. Lastly, the top softmax layer outputs a probability distribution of the classes. The architecture can be found in Figure 2. This is similar to the one used in [15].

In summary, the convolution and max-pooling layers are used as feature extractors. Using the local, time-series property of the data, the input is transformed into more general representations in a lower number of dimensions. The fully connected layers are used for classification. The output of the network is a confidence score associated with each of the activities. The sum of all confidence scores are normalized to one. We also incorporated common regulation techniques such as dropout, momentum, and weight decay to ensure that our model does not overfit to the training data.

3.3 Odometry

Although our project does not specifically address methods for estimating the distance a person travels using motion data alone, this topic has been well studied in the literature [14]. One simple approach to achieve this is to compute the Fourier transform for each sample. Since data for walking, running, or going up the stairs produces a periodic motion, we can extract the frequency of the walk by setting an activation threshold and search for the maximum frequency. By either collecting data or using publically available sources to determine the average step length as well as the variance, we could estimate the person's displacement from this information.

3.4 Turn detection

Another key aspect of odometry is the ability to detect when a user makes a turn. Since we focus on indoor office environments where the map is usually structured with defined corners, we can use detect changes in the average compass sensor readings to determine if and by how much the user has turned.

3.5 Virtual Building Map

In order to apply our method, we need to have an accurate map of the building with annotated locations that reflect different activities. In particular, we need a map with labeled hallways, staircases, sitting areas, and standing areas, onto which we map the probabilities output by the classifier.

We use floor maps of the *Bill and Melinda Gates Computer Science Complex* on the University of Texas at Austin campus. These maps are created using self-navigating robots and a laser mapping package from the Robot Operating System. These robots roam around the building autonomously and uses SLAM (Simultaneous Localization and Mapping) techniques to create seperate maps for each floor they explore. The files are stored as bitmap images, one per floor. We converted each floor map into a PNG image and colored in each region with a distinct RGB value (e.g., red for stairs, blue for hallways, etc.). We then convert the pixels into a 2D matrix where each color is represented by a digit (e.g., hallway = 1, stairs = 2, etc.). This 2D matrix is then used by our localization program to assign weights to different regions according to their numeric ID.

The convolutional neural network provides a classifi-

cation probability for each activity. Each time we get a new output from the classifier, we update the respective locations on the map by setting those region weights to the classification result. As an example, suppose the classifier outputs walking with a confidence of 0.6 and going up stairs with a confidence of 0.4. Then we set the weights of all hallways to 0.6 and the weights of all staircases to 0.4. All inaccessible parts of the map, such as walls or exterior space, receive a weight of 0.

Although we hand-label the regions on our map, this assumes that there can only be one activity per region. However, clearly a person may be, for example, in a hallway but not be walking. We can instead collect user motion data along with ground-truth location information (with the help of the BWI robots) and learn an activity probability distribution across the entire map. In this case, each pixel (i, j) of the map could be represented not by a single activity, but instead as a vector v_{ij} of l activity probabilities where each dimension represents the observed likelihood of a person performing a particular activity in that area (e.g., walking, standing, etc.). Since the classifier also produces an output vector c with the confidence of each of the l activities, the probability that a user is at location (i, j) at time t would simply be the dot product $p_{ij}^t = v_{ij} \cdot c^t$. This would produce a more accurate representation of the activities that people perform. However, for the purpose of our work, hand labeling is sufficient.

3.6 Particle Filter

The final step in our pipeline is to estimate the user's location with a particle filter. The particle filter is particularly applicable for our approach due to its ability to incorporate restrictions imposed by the environment, as well as its natural ability to simultaneously track multiple location hypotheses [10]. It has also proven to be very useful for localizing robots indoors [13].

Initially, all particles (x, y, θ) are randomly placed over the map with a weight of 1.0 and a random orientation. Each time the map probabilities are updated, we update the particle filter as follows:

1. We reduce the weights of all particles proportionally to the probability of the region that they fall over. The weights of all particles are then normalized with respect to the maximum particle weight.
2. Next, the particles are updated based on the movement speed (odometry) and the turning rate of the user. All particles are turned by an amount equivalent to the estimated turning rate and then moved forward proportionally to the estimated walking speed of the person being localized.
3. We perform a *random walk* by offsetting the position and angle of all particles by a small random amount. This has the effect of promoting "exploration" of the environment so that all particles don't end up converging to a single point, and allows the algorithm to recover from false predictions.
4. All particles are then resampled from the existing particles based on their weights. Particles residing over higher-probability regions will have larger weights, thus making them more likely to be resampled.
5. Finally, the user's position is estimated by computing the weighted average of all particle locations and orientations. We can get multiple weighted position hypotheses by computing separate averages for spatially separated particle clusters.

Figure 3 shows our particle filter implementation over the third floor of the building. It is clear to see that particles which are initialized or get randomly displayed into inaccessible areas of the map will not be resampled since their weight will quickly decrease to 0. By the nature of its design, the particle filter converges to new areas slowly. Hence, if the classifier makes a false prediction, it will not cause the particles to immediately deviate from their existing location and trajectories, making our approach very robust to noise in the raw sensor inputs as well as classifier failures. We can control the rate of convergence of the particle filter with a parameter that specifies the rate at which particle weights get updated.

To account for multiple floors, we can use a single set of particles across the disjoint floor maps. Only elevators and staircases would be shared across the maps, meaning that if a particle is in one of these regions and moves out of it, it would end up on one of the floors at random. It is also possible to incorporate odometry information, and other sensors such as the barometer (available on many modern smartphones), to estimate the user's current floor.

4. EXPERIMENTS

4.1 Training the Classifier

To train the CNN classifier, we used an evaluation dataset called Actitracker [11]. This dataset is consisting of 1,098,207 examples from 3-axis accelerometer data sampled at 20 Hz. It contains six activity classes (walking, jogging, upstairs, downstairs, sitting, and standing) from various participants. The distribution of the data points shows a bias toward walking and jogging. This dataset was selected and helpful in achieving our goal because of the large amount of data points it has and the diversity of gaits. Most of the activities were relavent to the localization scenario our method is trying to address. The data was also collected via

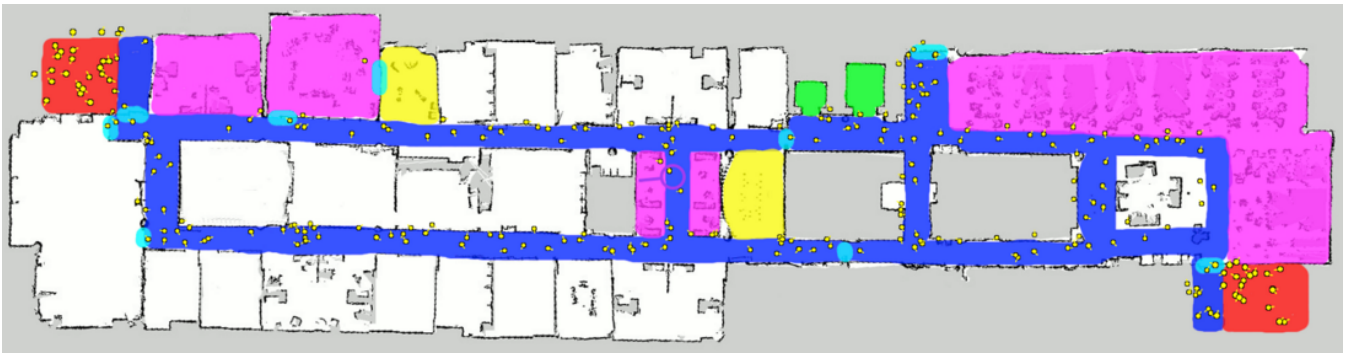

Figure 3: A visualization of our particle filter over the third floor of the *Bill and Melinda Gates Computer Science Complex*. The blue, red, cyan, yellow, purple, and green colors represent hallways, stairs, doors, standing areas, sitting areas, and elevators, respectively. The highest probability regions on this map were stairs and hallways.

smartphones placed in the users' pants pockets, similar to how we expect our system to collect data.

We implemented the neural network in Caffe [9]. Caffe is a deep learning framework written in C++. It is one of the fastest open source frameworks currently available, making it ideal for use in a real time system. We processed the Actitracker data into HDF5 format. HDF5 is one of the few data input formats that Caffe supports for fast batch sequential reading for training and evaluation. The data points were grouped into samples of length $N = 64$ (approximately 3 seconds). Sequential samples have 50% overlap in data points, ensuring that the network also learns from the time shifted data. The samples were then shuffled so that once we start learning the weights in batches, each batch has more evenly distributed classes. Each channel (x, y, z) is saved into an independent file.

Following the architecture described in Figure 2, each layer propagates its output onto the next. The input layers were batched into sizes of 200. We used a filter size $m = 20$ and shifted it one data point at a time with no additional zero padding. Three filters were used to improve the richness of the representation. The window for max-pooling was set to 3 with no overlap (shifting at 3 data points). The outputs from all pooling layers werefed to two fully connected (FC) layers with 1024 and 30 outputs respectively. After each FC layers, rectified nonlinear unit (ramp function) was used to make the gradients more stable during backpropagation computation. Additionally, dropout layers were also used after on the output of each FC layer with 50% dropout rate. This helped to prevent overfitting by using only a randomly selected 50% of the connections each time. And lastly, we used a softmax layer with loss for gradient computation during backpropagation. The weights at each layer were learned through the process.

For training, our base learning rate was 0.05, momentum 0.9 with weight decay of 0.0005. The starting weights were randomly initialized. We trained the network in GPU mode over 100,000 iterations. The plot of the loss over iterations can be found in Figure 4.

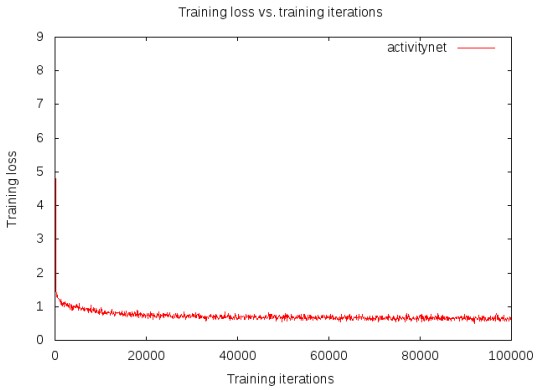

Figure 4: The training loss over 100,000 iterations we trained on.

4.2 Collecting Test Data

To collect user data, we developed an iPhone app that samples raw sensor information from the phone's accelerometer, gyroscope, and compass. The app can also track the user's GPS location, which can be used to initialize the particle filter when a user first enters the building from the outside.

When the iPhone was placed in a user's pocket, our app sampled the sensor readings at a rate of 20 times per second, to be consistent with the Actitracker data set. We exported the data as a file through an email interface. For each sample, we got 9 data points: the x, y, z axes for each of the three sensors. These values could be stored as 32-bit floats. Hence, we got $32 \times 9 \times 20 = 5760$ bits per second of data. This came out to less than 2.4 megabytes per hour. The small data footprint is essential for being able to incorporate real-time localization in a practical application of our system. For experimental purposes, we stored the data locally on the device,

Predicted class:	Walking	Jogging	Sitting	Standing	Upstairs	Downstairs
Walking	0.75	0.01	0.0	0.0	0.24	0.0
Standing	0.0	0.0	0.62	0.37	0.0	0.0
Upstairs	0.93	0.01	0.0	0.0	0.06	0.01
Downstairs	0.76	0.11	0.0	0.0	0.06	0.05

Table 1: The probability distribution for each of activities using our data training on Actitracker. Each row is a distribution of classifier outputs for that activity. Each activity contains data gathered by two different people from least two different locations. Due to the rounding error, the probability distributions might not add up to one.

and then exported them in batches. This allowed us to create a more controlled environment to run experiments.

The testing data was collected from two different adult males in their twenties. The iPhone was placed in the users left, front jean pocket with the camera tip of the phone pointing down, and the screen facing toward the user. All the activities were done in stairs and hallways of the second and third floors of the *Bill and Melinda Gates Computer Science Complex* (GDC). Since our localization module used maps from the same building, we found this data best represents our testing environment.

4.3 Particle Filter Setup

We found experimentally that the particle filter works best when we use approximately one particle for every two squared meters of floor space that we are localizing over. For our experiment in the north-east section of the third floor of the GDC building, we used 2000 particles. We moved the particles on the map forward proportionally to the observed odometry of the user. The number of pixels per meter depended on resolution of the map. Our map had a resolution of 20 pixels per meter; thus for every meter the user moves, we moved all particles 20 pixels forward. Whenever the user turned, we turned all particles by the same amount. To account for uncertainty in the turning direction, if the user moved by α radians, we randomly added or subtracted α to the current orientation of all particles. Those particles which were turned in the wrong direction will likely bump into a wall or wander off into low-probability areas and would not get resampled.

For the random walk, all particles were moved a random distance between 0 and 1.5 meters (or 0 to 30 pixels), and each particle's orientation was randomly shifted between $-\frac{\pi}{6}$ and $+\frac{\pi}{6}$. We also set the weight decay to 0.8. This parameter controled the rate at which particles adapt to the probability of their position on the map. A higher weight decay made the particles converge quickly, but also makes the system more susceptible to false convergence due to noise. The particle filter was updated each time the classifier produces a new output, which was approximately once every 1.5 seconds.

The particle filter could produce multiple hypotheses based on a spatial clustering of the particles. We considered the "best" cluster to be the cluster that has the highest overall particle weight. For evaluation purposes, we also considered a "closest" cluster, which is the cluster whose location is closest to the ground truth at any given iteration. Ideally, the closest cluster to the ground truth would also be the cluster with the highest probability, but this is not always the case.

4.4 Evaluation

We have trained the neural network with the Actitracker dataset and tested it with the data we gathered using our app. We used the class with the highest output probability as the classifier output. Table 1 shows the probability distribution of the results for each of the activities we have performed. The classifier was able to distinguish between the motion and motionless samples with near perfect accuracy. However, between the different motion classes, it was not able to provide meaningful results. The classification currently is too noisy to be useful in localization.

There are a few reasons why our classification results are suboptimal. We are trying to use a classifier trained on a separate dataset and this naturally leads to several problems. The hardware, setup, and data collection methodologies cannot be exactly matched. For example, the Actitracker data is scaled differently than our data. Additionally, the alignment of the axes might not be the same either. We preprocessed our data to the best of our abilities to make it as coherent as possible. Secondly, parameter tuning remained a time consuming and non-standardize process. We were only able to explore a subset of all possible combinations of parameters given the time we had. Caffe's high setup cost of restructuring data further impeded the progress.

Additionally, because GDC lacks a long stretch continuous stairs, the staircase data we collected contains a combination of stairs and walking akin the zigzag structure of the staircases. We currently cannot distinguish the independent parts of this data with separate labels, so we simply used "up stairs" or "down stairs" labels. This is mainly due to our inability to gather high fidelity ground truth localization data inside the building

to finely label our data.

Nevertheless, other research has shown good performance with a similar architecture [15]. This shows that proficient activity recognition is possible with good tuning and better data processing and augmentation. For this project, we tested the rest of the modules independently.

We tested our particle filter by manually controlling a simulation of a person moving around the map. The simulation can generate artificial classifier ouputs, odometry, and turning rate data based on the ground-truth location of where the simulated person is on the map and how they are moving. We then added noise to this simulated data and fed it as input to the particle filter.

Noise was added to the ground truth data as follows. For motion, we specified a noise value $\mathcal{N} \in [0, 1]$. For the amount the user moved m, we added or subtracted from m a random value between 0 and $\mathcal{N}m$. For the amount the user had turned θ, we added or subtracted from θ a random value between 0 and $\frac{\mathcal{N}\pi}{4}$. Next, for the activity classifier noise, we specified separate noise value $\sigma \in [0, 1]$. We then added noise to each region probability i in the classifier output vector c by setting $c_i = \text{abs}(c_i + g(0, \sigma))$ where g is a Gaussian distribution. Vector c is then normalized.

Figure 5 shows our results with particles initialized at random and with particles initialized at the ground truth starting location. When the initialization was random, our performance was poor. However, if the particles were initialized in a good starting location, we achieved very good accuracy, with an error of only 2.175 meters for the best cluster, and 1.265 if we always consider the closest cluster to the ground truth. These results are close to [5] who use magnetic positioning to get an accuracy of under 1 meter 88% of the time. However, their approach requires expensive and specialized hardware.

We also compared our approach to baseline performance of using odometry and turn information (without activity classification) to update the particle filter. The results (seen in figure 6) show that our approach provides a significant performance boost to localization.

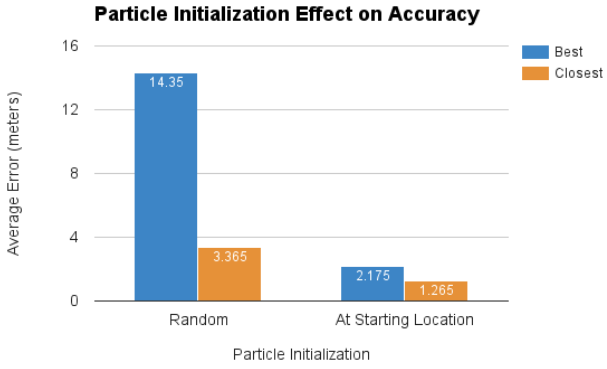

Figure 5: The average localization error when the particles were initialized randomly vs. at the ground truth starting location. The noise was fixed at 0.2 for both the motion data and classifier. The "best" columns show the prediction error of the particle cluster with the highest probability weight. The "closest" columns show the average error of the cluster that's closest to the ground truth. Ideally, the closest cluster is also the best cluster. This data shows that our method's accuracy is very high when the particles were initialized in a good starting location. In practice, such a good initialization may be available when a person first enters the building, since outdoor GPS readings will narrow their location down to a single point of entree before indoor localization begins.

5. CONCLUSION

We introduced a new method for indoor localization that uses only the motion data collected from a person's smartphone to determine their location in a building. Our method has several advantages over other previous methods. Most notibly, it does not require any special hardware (both for the client and infrastructure) or complex software modifications. A user needs only to download an app on their mobile device. Since we do not rely on WiFi localization, our method is unaffected by radio interference or noise that typical localization methods suffer from. Although we show that we can provide reasonably accurate location estimates using motion data alone, our particle filter can easily be extended to support additional inputs from other localization techniques. These characteristics of our approach make it an encompassing, highly portable, and modular platform that can be easily deployed.

There are several limitations of our method. First, our method only works if we have an accurately annotated map of the building. Although annotations can be derived from training data, a digital floor plan of the building is still required for our approach to work. We also assume that we can get reasonably accurate activity classifications from the user's motion data. This may not always be the case. People have a large variety of gaits for performing all these activities depending on lifestyle, fitness, and other reasons. It is challenging to gather enough data and train a classifier robustly and in a general enough way to capture the wide spectrum. Lastly, due to the need of classifying the user's activity and updating the particle filter, our method is significantly more computationally expensive than other localization methods, and must be run on a dedicated localization server rather than directly on the mobile device.

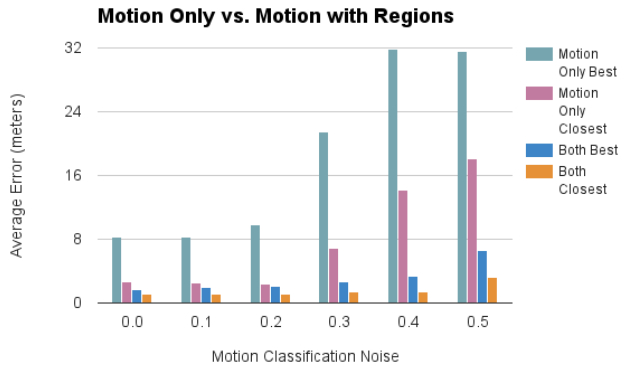

Figure 6: This figure shows the average localization error over ten trials for varying noise levels. All particles were initialized at the ground truth starting location. The "motion only" values shows the particle filter's performance when only considering odometry and turning information. "Both" includes the region probabilities from the activity classifier along with odometry and turning information. The x-axis shows varying levels of motion noise. The activity classifier noise was fixed at 0.2. Clearly, our addition of region probabilities improved performance substantially.

5.1 Future Work

We would like to experiment with incorporating other localization techniques into our our particle filter and test it against the current accuracy of our method. This would require carefully weighting the map probabilities that we input into the particle filter based on the reliability of each new probability signal. We would also like to evaluate our system by testing an end-to-end prototype of the entire pipeline of our approach. This means streaming data directly from one or more users and getting real-time location estimates.

Beyond improving the accuracy of the localization, we would like to further integrate our approach into the *Building-Wide Intelligence* (BWI) project at our department. Our localization technique can help the autonomous BWI robots track the locations of people in the building to better allocate resource and dispatch more units to highly populated areas. Furthermore, we can use the activity classification results to detect interesting events in the building. For example, if we detect a group of people dancing in a particular spot, a robot can go there and play some funky music or join them in the dancing. Or more realistically, if we detect a group of students playing table tennis in the graduate lounge, a robot can go there to offer some snacks or beverages. Or, if someone falls or gets injured, the BWI system can dispatch a rescure robot to their location. Because our apporach doesn't require special hardware, this can be realized with very little overhead.

Our system can also be used to monitor human group behavior. Since it can provide both the location and current activity of everyone in the building, we can use this data to better understand the environment and provide improvements accordingly. For example, if we can detect that people often stand around a particular area for extended periods of time, we can put some chairs there so they can have a place to sit.

Finally, we can further extend this project to annotate the environment using activity recognition. First, we can use GPS to determine the starting location of the user before they enter the building. Then, given enough data, we could map out a distribution of where people commonly perform different types of activities. This information could be valuable to further improve the performance of our system and also be used in other types of human behavior studies.

6. REFERENCES

[1] E. Ahmed, M. Jones, and T. K. Marks. An improved deep learning architecture for person re-identification. *Differences*, 5:25, 2015.

[2] A. Bedagkar-Gala and S. K. Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, 2014.

[3] J. Biswas and M. Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4379–4384. IEEE, 2010.

[4] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245. ACM, 2005.

[5] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor location sensing using geo-magnetism. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 141–154. ACM, 2011.

[6] I. Constandache, R. R. Choudhury, and I. Rhee. Compacc: Using mobile phone compasses and accelerometers for localization. In *IEEE INFOCOM*, 2010.

[7] P. Gupta and T. Dallas. Feature selection and activity recognition system using a single triaxial accelerometer. *Biomedical Engineering, IEEE Transactions on*, 61(6):1780–1786, 2014.

[8] C.-H. Hsu and C.-H. Yu. An accelerometer based approach for indoor localization. In *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on*, pages

223–227. IEEE, 2009.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[10] N. Kothari, B. Kannan, E. D. Glasgwow, and M. B. Dias. Robust indoor localization on a commercial smart phone. *Procedia Computer Science*, 10:1114–1120, 2012.

[11] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[12] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *AAAI*, volume 5, pages 1541–1546, 2005.

[13] S. Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[14] S. Yang and Q. Li. Inertial sensor-based methods in walking speed estimation: A systematic review. *Sensors*, 12(5):6102–6116, 2012.

[15] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 197–205. IEEE, 2014.

APPENDIX

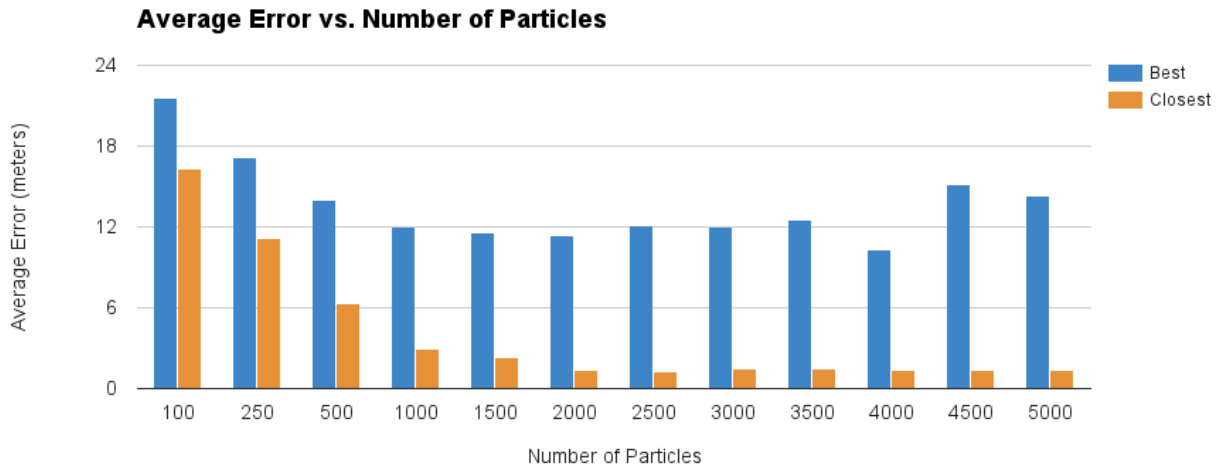

Figure 7: This graph shows the performance of our particle filter (initialized randomly) with an increasing number of particles used. This experiment was done using the third floor north-east (3ne) map of GDC. We get optimal performance at around 4000 particles; however, due to the computational power required to run the particle filter, we use 2000 particles, since the accuracy was similar.

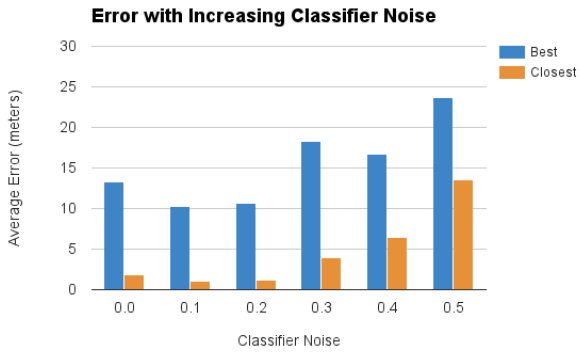

Figure 8: The accuracy of our method with increasing classifier noise. All particles were initialized randomly and motion noise was fixed at 0.2.

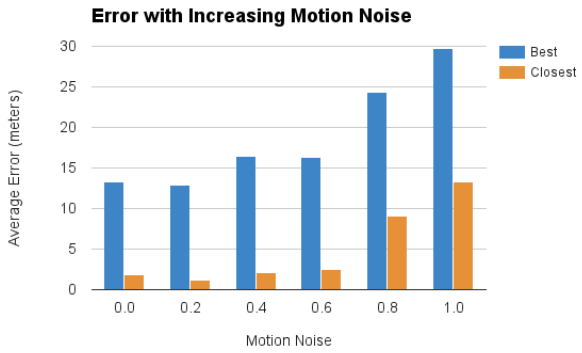

Figure 9: The accuracy of our method with increasing motion noise. All particles were initialized randomly and classifier noise was fixed at 0.2.

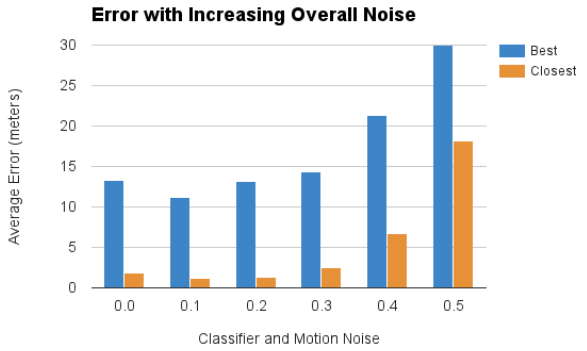

Figure 10: The accuracy of our method with increasing overall noise. The x-axis shows the noise set for both the classifier and the motion signals. All particles were initialized randomly.

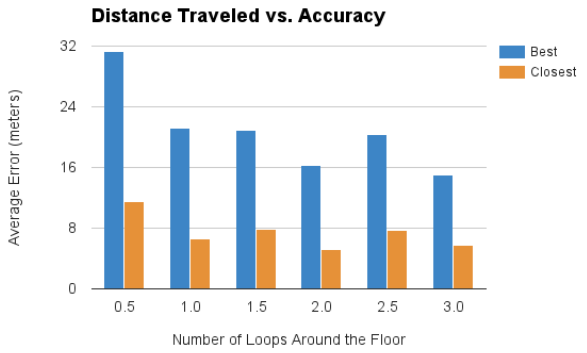

Figure 11: Over time, the accuracy of our method increases as the particles have longer to converge. In this experiment, classifier noise was fixed at 0.25 and motion noise was fixed at 0.35. The simulated user was moved around a single floor in a loop. The number of loops around that floor is indicated by the x-axis. As the user walks for an extended period of time, the error decreases since the particles begin to converge.