Exploration for Range Based Indoor Localization Technologies and Algorithms

Casey Mackin

August 9, 2018

Acknowledgement

Claire Tomlin, PhD (Primary Advisor)
Murat Arcak, PhD (Second Reader)
Divyesh Jadav, PhD (Advising, funding)
Tom Griffin, PhD (Experimental Infrastructure, advising)
German Flores, PhD (Experimental Infrastructure, advising)

Exploration for Range Based Indoor Localization Technologies and Algorithms

by Casey Mackin

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Claire J. Tomlin
Research Advisor

08/01/2018

* * * * * * *

Professor Murat Arcak
Second Reader

08/01/2018

ABSTRACT

Indoor localization is becoming an increasingly relevant topic as the world rapidly becomes more connected through existing and emerging wireless technologies. The use cases of location based services (LBS) are endless; including locating people and objects inside of hospitals, corporate headquarters, subway stations, manufacturing facilities, and more.

Due to the wide range of applications for indoor localization, many solutions have been explored with important trade-offs in energy efficiency, accuracy, range, cost, and availability. Technologies including Wi-Fi, Radio Frequency Identification (RFID), and Ultra-Wideband (UWB) perform well in some metrics but poorly in others. Bluetooth low energy (BLE) technology performs well in most metrics except accuracy. Range-based solutions often use received signal strength (RSS) as it a free metric with compatible technologies (Bluetooth and Wi-Fi). However, RSS is prone to large error when converting to distance estimates due to multipath propagation and interference from obstructions in the environment. Still, the low cost of BLE technology allows for many nodes to be deployed and filtering and other machine learning techniques can be applied to achieve improved localization.

Several techniques for performing localization based on RSS of BLE nodes are explored. We explore algorithms based on nearest neighbors, extended Kalman filtering, and several supervised learning methods. A simulated environment is developed to quickly test the robustness of algorithms under varying conditions and assumptions. A discussion of trade-offs associated with using each technology and algorithm is presented.

ABBREVIATIONS

AoA	Angle of Arrival
BLE	Bluetooth Low Energy
BRNN	Bidirectional Recurrent Neural Network
EKF	Extended Kalman Filtering
IoT	Internet of Things
IPS	Indoor Positioning System
KF	Kalman Filter
LBS	Location Based Services
LSTM	Long-Short Term Memory
NN	(Artificial) Neural Network
PBS	Proximity Based Services
RNN	Recurrent Neural Network
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
ToF	Time of Flight
TDoA	Time Difference of Arrival

TABLE OF CONTENTS

1 INTRODUCTION

1.1 BACKGROUND & MOTIVATION

The global positioning system (GPS) has become widely adopted as the core technology behind many existing outdoor location based services today. However, GPS is not a viable technology for location based services in heavily obstructed environments like buildings and subways. GPS accuracy is also inadequate for many services that require precise positioning. Several alternative technologies are being developed to address the limitations of GPS especially for indoor environments.

In addition, wide-scale use of smart mobile devices and the growth of internet of things (IoT) is spurring motivation to create new location based services (LBS) for indoor and outdoor environments. There is tremendous value in the data generated from widespread use of these services. Indoor positioning systems (IPS) have been used in environments including hospitals, shopping malls, subways, factories, stadiums, and much more [1]. Consider the following as a few motivating examples that illustrate the large-scale potential impact of these technologies. Indoor positioning systems can be used to monitor the locations of occupants within a building to enable/disable heating and cooling in subsections. This can translate to immense energy reduction while still maintaining or improving the comfort of occupants. Building maintenance workers can also improve their efficiency by knowing which subsections have low usage and require less maintenance and which subsections have high usage and require constant care. In addition, occupants can report issues in the building along with their current location. IPS applications are becoming more and more complex with the introduction of smart homes, offices, vehicles, etc.

In this report, we aim to provide several contributions in the following sections:

- **Overview of important performance metrics**: We discuss important performance metrics in designing indoor positioning systems.
- **Overview of leading technologies**: We discuss several technologies currently being used for indoor localization.
- **Overview of common techniques**: We discuss several range-based techniques and the pros and cons of each.
- **Overview of challenges**: We discuss several challenges that designers must consider in creating an indoor positioning system.
- **Overview of simulated environment**: We discuss a simulated environment developed to quickly test various indoor localization techniques.
- **Overview of experimental environment**: We describe the real-world experimental setup with details of the utilized technologies.
- **Overview of covered methods**: We briefly go through the details of each localization method that is applied in this paper.
- **Simulated results and comparison of methods:** We compare the results for each method using the simulated environment.
- **Experimental results:** We review the results for several methods used in the experimental lab environment.

- **Conclusions and future work:** We discuss the implications of the simulated and experimental results and discuss future work for this project.

1.2 DESCRIPTION OF PERFORMANCE METRICS

To make this technology viable, designers must consider trade-offs in several categories. More complete and detailed metrics important for indoor positioning systems are described in [1] [2]. We consider a subset of these performance metrics that are vital for use cases we are developing:

- **Availability**: This metric considers how common the technology is in everyday devices today including cell phones, tablets, and laptops. This metric is important as it allows the technology to scale easier as many users already own compatible devices and can utilize the technology with minimal effort or additional cost.
- **Cost**: This metric considers hardware, maintenance, and setup cost for the IPS. Low cost is essential in making the technology available to as many consumers as possible.
- **Energy efficiency**: This metric considers the amount of energy consumed by the hardware and software needed to run the IPS. This is determined mainly by the transmission power and computational cost of algorithms.
- **Range**: This metric considers the reception range of the hardware. Reception range is closely tied with cost and energy efficiency. Range limitations are often countered by adding nodes to the system. This affects equipment and maintenance cost directly and can have a drastic negative effect on the energy consumption of complex algorithms as computational complexity increases.
- **Accuracy**: This metric considers jointly the accuracy of the hardware and algorithms applied for localization. Some applications only require proximity while others require micro-location. The accuracy of the underlying hardware often has a large impact on how well the algorithms perform. More complex algorithms often help remedy lack of accuracy in hardware.
- **Scalability**: This metric considers the system's ability to track large numbers of users or devices. Building constraints (i.e. outlets or power supplies) can limit the scalability of hardware solutions. Computational constraints can limit the scalability of algorithms (i.e. the number of nodes that can be added).

1.3 OVERVIEW OF TECHNOLOGIES

In this section, we will provide a summary of existing technologies currently used in indoor localization. We will also briefly discuss some of the strengths and weaknesses of each technology and explain the factors that lead to the experimental setup for this project.

Wi-Fi is one of the most commonly available technologies for localization inside of buildings. The main benefits of Wi-Fi are its availability, range, and scalability. Many large buildings where localization applications are highly motivated are already equipped with several Wi-Fi access points. In addition, most modern mobile devices are equipped with Wi-Fi capability. The range of Wi-Fi can be from 50-100m depending on the environment and hardware [2]. The main drawbacks of Wi-Fi are the cost of adding additional nodes, the high-energy consumption, and the low accuracy of the RSS based distance estimates.

Bluetooth is another widely available technology in terms of user devices. Bluetooth has several benefits including availability, low energy, low cost, and range. The range on Bluetooth is typically shorter than Wi-Fi ranging from 10-15m [2]. Most modern mobile devices are equipped with Bluetooth technology. Bluetooth low energy (BLE) solutions are preferred to Wi-Fi due to low energy consumption, configurability, and reported better performance even with identically placed nodes [3]. BLE beacon devices are also cheap compared to Wi-Fi and other technologies when considering the need to add additional nodes. Some previous work even adds BLE technology to Wi-Fi to create additional nodes for RSS based localization systems [3]. Like Wi-Fi, the main downside to BLE is the low accuracy of distance estimates based on the RSS. Due to all the other benefits Bluetooth technology is popular in localization literature [3] [4] [5] [6] [7].

Ultra-wideband (UWB) technology is not available in common mobile devices. It is also more expensive than other technologies. This makes this an overall poor overall solution for many applications that need to reach many consumers or need to run continuously. The major benefit of UWB is its high level of accuracy. UWB works by transmitting short pulses over a large bandwidth [2]. UWB has several significant advantages over Bluetooth and Wi-Fi: 1) It is more robust to interference from other signals 2) it is robust to disturbance from many materials (accurate through walls) 3) it is robust to multipath effects [2] [6]. However, UWB is sensitive to interference from metals and liquids which can be problematic in some indoor environments [2]. For more details on UWB technology refer to [2] [6] [8]. Sources report ranging accuracy as higher than 20 cm [2] and some higher than 10cm [8]. This makes UWB a great solution for collecting accurate training data for indoor localization and for real-time control applications.

ZigBee, RFID, Visible light, Acoustics, and Ultrasound are all less commonly used technologies that can also be found in the literature. For this paper, we focus on developing IPS based on BLE technology as the availability, low cost, and low energy consumption make it ideal for wide-spread use. We discuss several techniques that mitigate the sacrifice in accuracy with BLE. In addition, we discuss utilizing UWB technology for collecting "ground truth" training data given its high level of accuracy relative to more available technologies.

A popular emerging technology for BLE devices is Apple's iBeacon protocol which utilizes beacon transmitters and Bluetooth technology to provide proximity based services [3] [9]. These devices are highly energy efficient. More details on the iBeacon protocol can be found in Chapter 2 where we describe the experimental setup utilizing the iBeacon protocol. For detailed documentation on the iBeacon protocol refer to [3] [9].

1.4 OVERVIEW OF RANGE BASED METRICS

In this section, we will provide a summary of existing sensor metrics and techniques used for indoor localization. The most common signal metrics are received signal strength (RSS), Angle of Arrival (AoA), time of flight (ToF), and time difference of arrival (TDoA).

RSS is a relatively simple metric that is commonly used for localization. It is the measurement of the transmitted signal power seen at the receiver. In general, the signal is strongest near the transmitter and decays logarithmically as the distance increases [6]. Path loss models are popular for modelling this logarithmic decay and are covered in much of the literature [6] [10]. RSS works particularly well for proximity services as the user can confidently indicate that an agent is near a node when the RSS for that node nears the known transmission power for the transmitter or exceeds a specified threshold. RSS has been heavily used in the literature for localization [11].

Time of flight (ToF) or time of arrival (ToA) is a measurement of the time it takes for the signal to propagate from transmitter to receiver [5]. Because the speed of the signal is the known, all that is needed is the time of transmission and the time of reception to estimate the distance between transmitter and receiver. The distance is simply the difference in timestamps multiplied by the speed of the signal. However, ToF requires highly accurate synchronization between transmitters and receivers [1]. This makes implementation with most commonly available platforms difficult. UWB systems using ToF have been developed that include both transmitters and receivers that are synchronized.

TDoA uses multiple receivers to measure the difference in arrival time of the transmitted signal. In TDoA the time of transmission does not need to be known. Using only the difference in arrival time at the receivers and the known speed of the signal, we can compute the difference in receiver distance from the agent [6] [5]. Therefore, for this metric only the receivers need to be synchronized and not the transmitters as in ToF [1].

AoA is a more complicated metric that utilizes an array of receivers and estimates the angle the signal is arriving from the time difference of arrival in the array [6] [5]. However, this requires costlier equipment and requires precise calibration making it a difficult solution to incorporate into readily available platforms [1] [5]. The computational cost of this metric is also relatively heavy when compared to simple metrics like RSS [1].

Due to the heavy additional incurred costs of equipment or computational energy for utilizing many of these signal metrics, the RSS metric is ideal for large scale applications. However, for many applications that are less cost constrained and/or require more accurate results with less nodes, these other metrics can provide significant advantages.

1.5 OVERVIEW OF RANGE BASED TECHNIQUES

There are several techniques for utilizing received signal strength (RSS) metrics for localization. The most basic techniques are based on fingerprinting. The designer collects fingerprints, or sensor readings for a state, and then maps future readings to the state corresponding to the "nearest" stored fingerprint or a function of the set of "nearest" fingerprints. The most common method for mapping readings to fingerprints is k-nearest neighbors. In k-nearest neighbor regression, the predicted state is the average of the k-nearest neighbors to the current sensor reading. The limitations of k-nearest neighbors arise when new sensor readings deviate widely from existing fingerprints.

Kalman filtering is a popular solution in many applications for predicting the state of a system based on the system's dynamics model, known control inputs, and sequential measurements with a known sensor model. The Kalman filter is a recursive filter and thus only requires the most recent prediction, control input, and set of measurements to make the next prediction. This makes it ideal for real-time applications. The Kalman filter weights each sensor measurement based on the modeled uncertainty and thus can be easily extended to combine measurements from multiple types of sensors ("sensor fusion"). Basic Kalman filtering is limited to linear systems, but adaptations including the extended Kalman and unscented Kalman filter can be applied to non-linear systems as well.

Artificial neural networks are universal function approximators that have gained popularity for many applications as computational power has increased dramatically. They can be used to learn highly non-linear functions as needed for mapping RSS measurements to position. Generally, as the neural network's size increases, so does its expressive power. However, the amount of training data required to effectively train the network also increases. Neural networks have been trained for indoor localization application using labeled training data to infer position from sensors measurements. The challenge in this approach arises in collecting labeled training data and designing networks with enough expressive power. In addition, a basic feed-forward neural network does not maintain information from previous states which is highly beneficial for applications using sequential information. Neural networks have been used in the previous literature to perform localization based on RSS and other metrics [12].

Recurrent neural networks allow for learning functions to perform inference on sequential data and can provide sequential output. The internal state is maintained and passed on through sequential RNN cells. For indoor localization applications with known control inputs, the controls can also be included as inputs to the network and the sequential effect of control inputs on the internal state can be learned. Memory of an internal state helps to deal with issues that arise with fingerprinting and basic feed-forward networks when sensor measurements do not match well with previous fingerprints (difficult to interpolate) or when noisy sensor measurement look nearly identical to fingerprints with drastically different states (causing large jumps in state).

A related technique to RNNs are Bidirectional Recurrent Neural Networks (BRNN). BRNNs allow for the internal state to be shared both forwards and backwards in time. They also allow for depth to be added in the form of layers of interconnected forward and backward recurrent neural networks.

1.6 OVERVIEW OF CHALLENGES

In this section, we briefly describe several challenges that designers must consider in creating an indoor positioning system. This list is not exhaustive but includes common major challenges that arise in much of the literature:

- **Multipath effects and Noise**: Wireless signals can reach a receiver via many paths, and each path can have a different phase shift. This is especially a challenge in narrowband technology because it is unclear which if any of the received signals is accurate. Wide band technology reduces multipath effect by transmitting at various frequencies and using all the received signals to determine the true underlying value.

- **Radio environment**: Indoor environments often have several sources of radio signals. Cell phones, Wi-Fi, Bluetooth, FM/AM, and many more can make indoor environments extremely noisy when trying to measure RSS. Placing transmitters or receivers close to other transmitters can amplify this problem.
- **Cost**: The cost of the equipment, setup, maintenance, and energy is an important factor in making a viable indoor positioning system. In choosing equipment, configurations, and algorithms as well as other factors the designer always need to consider the overall cost of the system.
- **Negative impact on used technology**:
    - Wi-Fi heavy methods: Utilizing Wi-Fi for indoor positioning systems can take away from its availability for other applications, using up bandwidth. This is especially important for applications where many objects are being tracked in a dense environment.
    - Bluetooth heavy methods: Bluetooth heavy methods can run into issues of signal interference with other devices.

1.7 NOVELTY OF PROPOSED INDOOR LOCALIZATION SYSTEM

The hardware (beacons, receivers, network equipment) and data collection software (RESTful web service, receiver software) was already setup in an indoor lab environment from a previous indoor localization project describe in [13]. The previous project employed K-nearest neighbor methods. The K-nearest neighbor algorithm was applied to manually recorded fingerprints. Several difficulties arise in this project; motivating the alternative approaches discussed in this report.

One difficulty comes with collecting the fingerprints manually. This collection of fingerprint data is extremely tedious and difficult to do accurately. On top of this, continuous sequential fingerprints with accurate time values are even more difficult to collect. This was accomplished by creating an app that allows the user to tilt the screen to move a marker through the map as the user attempts to manually relocate the receiver to the marker at the correct time [13]. Another difficulty comes with collecting adequate data. Due to the significant random noise in the signal, it is difficult to guarantee there is high enough granularity to distinguish between states. Also, various states may have similar fingerprints that are made indistinguishable by noise in the environment. This can cause the nearest neighbor algorithm to produce dramatic state changes that do not align with the agents underlying dynamics (i.e. large instantaneous jumps in position). To address these concerns, we explore several solutions; some of which are novel to the best of our knowledge.

An existing solution is extended Kalman filtering which does not require large sets of training data and instead only uses the most recent measurement data. This method predicts the state over time based not only on sensor readings but also on a designer specified dynamics model of the underlying agent. This helps remedy extreme changes in the state due to noise in the sensor readings. The difficulty with extended Kalman filter is that it requires an accurate dynamics model with noise covariance model and an accurate sensor model with noise covariance model. An accurate dynamics model is often obtainable for robotic systems.

However, an accurate sensor noise model for indoor environments is difficult due to the previously mentioned challenges.

To investigate more novel solutions, we explore several supervised learning tools. We implement artificial neural network architectures to do inference based on the RSS values. The first network is a feed-forward network. With adequate training, we intend for this network to produce accurate estimates even when a closely matching data point was not provided in the training data; addressing one concern for nearest neighbor based methods. To address the final concern, we implement two recurrent neural network architectures; One forward recurrent and one bidirectional. These allow us to train models for sequential inference. This architecture propagates an internal state through several recurrent cells, and learns how the state changes over time based on sensor measurements and sequential control inputs. We intend that this architecture remedy the instantaneous state changes as seen with nearest neighbor based methods and with basic feed-forward neural networks. The use of such an architecture, to our knowledge, is novel in the space of range-based indoor localization.

1.8 RESEARCH QUESTION & PROBLEM STATEMENT

The goal of this project is to explore various tools for their practical use in performing indoor localization and in collecting training data for localization algorithms. We will start by posing our definition of the problem of state estimation. We start with an environment including n range sensors with known locations. The known sensor locations are represented as:

$$P = [p_1, p_2, \ldots, p_N]$$

$$p_i = [x_p^i, y_p^i]$$

We also have a system with a known dynamics model and a receiver with known sensor model. The dynamics and sensor models are represented as:

$$x_{k+1} = f(x_k, u_k, v_k)$$

$$z_k = h(x_k, w_k)$$

We assume the process noise and sensor model noise are additive and sampled from Gaussian distributions both with zero mean and known covariance. Hence the dynamics and sensor models assuming additive Gaussian noise can be rewritten:

$$x_{k+1} = f(x_k, u_k) + v_k \qquad v_k \sim N(0, Q_k) \tag{1}$$

$$z_k = h(x_k) + w_k \qquad w_k \sim N(0, R_k) \tag{2}$$

We represent the system with 4 states including the positions and velocities in the x and y directions:

$$\mathbf{x_k} = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}$$

The control input is assumed to be the acceleration in the x and y directions.

$$u_k = \begin{bmatrix} a_k^x \\ a_k^y \end{bmatrix}$$

For our computations, we need a linear representation of the dynamics model with additive Gaussian noise (if the dynamics are non-linear then a local linearization is needed as seen in extended Kalman filter section).

$$x_{k+1} = A\mathbf{x_k} + Bu_k + v_k \qquad v_k \sim N(0, Q_k) \tag{3}$$

If the control input is unknown, assume $u_k = 0$ and model possible controls as additional additive noise. This is often used when tracking people if IMU/accelerometer data is not available. For the purposes of this report we use a double integrator model so dynamics matrices are:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{5}$$

In our environment, the sensor model is non-linear and is a function of the state at the current time only. The non-linear sensor model is represented as**:**

$$z_k = h(x_k) + w_k \qquad w_k \sim N(0, R_k) \tag{6}$$

Our non-linear sensor model for the range sensors is distance from the known sensor positions to the agent position:

$$h(x_k) = \begin{bmatrix} h_1(x_k) \\ h_2(x_k) \\ \vdots \\ h_N(x_k) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_k - x_p^1)^2 + (y_k - y_p^1)^2} \\ \sqrt{(x_k - x_p^2)^2 + (y_k - y_p^2)^2} \\ \vdots \\ \sqrt{(x_k - x_p^N)^2 + (y_k - y_p^N)^2} \end{bmatrix} \tag{7}$$

The sensors return a noisy distance measurement of the agent's current position from each of the known sensor positions. In the real system, the distances are computed from the received signal strength indicator (RSSI) values obtained from the iBeacons. A description of RSSI and a summary of the various sources of the noise in the RSSI measurements is included in the section 2. In depth discussion of generating noisy measurements in simulation is covered in the section 3.

A linear sensor measurement model is useful for many applications including extended Kalman filtering. Determining this linear approximation will be discussed in detail in our description of the extended Kalman filter algorithm.

The challenge for this system lies in modeling the sensor noise covariance which is state dependent and is affected by the several factors in indoor environments. Several methods are explored for assigning the sensor noise covariance matrix and a discussion of tradeoffs/pitfalls of each method is included.

We will explore several techniques for performing indoor localization using this information. Each technique will be described in later sections and each is compared using simulated experimental results. The goal of this project is to explore methods for indoor localization using BLE technology. Furthermore, we explore tradeoffs between different methods.

Table 1: Description of variables

Variable	Description
x_k	State vector at time step k
u_k	Control input at time step k
z_k	Sensor measurement vector at time step k
$f(...)$	Dynamics model
$h(...)$	Sensor measurement model
v_k	Process noise at time step k
w_k	Sensor measurement noise at time step k
Q_k	Process noise covariance matrix
R_k	Sensor noise covariance matrix
x_k^p, y_k^p	Known x, y positions of sensor i
x_k, y_k	Actual x, y position of client

2 SIMULATED ENVIRONMENT

For this project, we also created a simulated environment for testing localization algorithms. The simulated environment allows us to rapidly test various algorithms and compare the results without needing to worry about issues that arise in integrating and debugging the various types of hardware seen in the actual system. We will start by outlining the components of the simulated environment that the designer specifies.

Table 2: Parameters for simulations

Value	Format	Dimension
Environment Size Dimensions	Tuple of x, y limits	2, 1
Beacon Locations	List of x, y coordinates	2, N
Dynamics Model, Noise Covariance Matrix	Any function, PSD	Any
Sensor Model, Noise Covariance Matrix	Any function, PSD	Any
Disturbance Model	Any function	N/A
State deviation (Q) and control input (R) cost matrices, and horizon T (for LQR control only)	PSD, PSD, $1 \leq T \leq \inf$	$[n_x, n_x]$, $[n_u, n_u]$
Trajectory	States over t	$[n_x, T]$

Node configurations (quantity and position) in the simulator can be generated randomly or at designer specified positions. The dynamics model can be any desired function. If the dynamics are non-linear, a locally linearized approximation is computed for use in the extended Kalman filter. The positive semi-definite noise covariance matrix needs to be specified (or a function for determining the sensor noise covariance matrix at run-time).

The sensor model can also be any desired function. For our experiments, it is the Euclidean-norm distance from agent to sensor and a locally linearized approximation is computed for use in the extended Kalman filter. Again, a positive semi-definite noise covariance matrix must be specified (or an algorithm for determining the noise covariance matrix at run-time). Some experiments gather samples of the signals and compute the variance of the sample set to construct the covariance matrix.

The disturbance model can be any desired function. For real-world scenarios, this function is typically unknown and difficult to determine. In simulation, the purpose of designing this function is to test the robustness of algorithms under various types of disturbances. The specified function can be state dependent (a function of the position in the environment). The numerical integrals from the position of each node to the agent are computed. The integrals are used to determine the parameters of the Gaussian distribution from which to generate the disturbances. The functions f and g can be chosen by the designer to create different effects in the simulated environment. Many models show received signal strength decreasing logarithmically with distance for indoor and outdoor environments as shown in popular path loss models [10]. These types of models can easily be incorporated into the simulations. In equation 9 below, z represents the numerical integral between a node and the agent.

$$d_k \sim N(\mu, \sigma^2) \qquad \mu = f(z) \qquad \sigma^2 = g(z) \tag{8}$$

Figure 6 below shows an example function for generating additive noise for the sensors. The function represents two perpendicular intersecting walls and a single point source of disturbance.

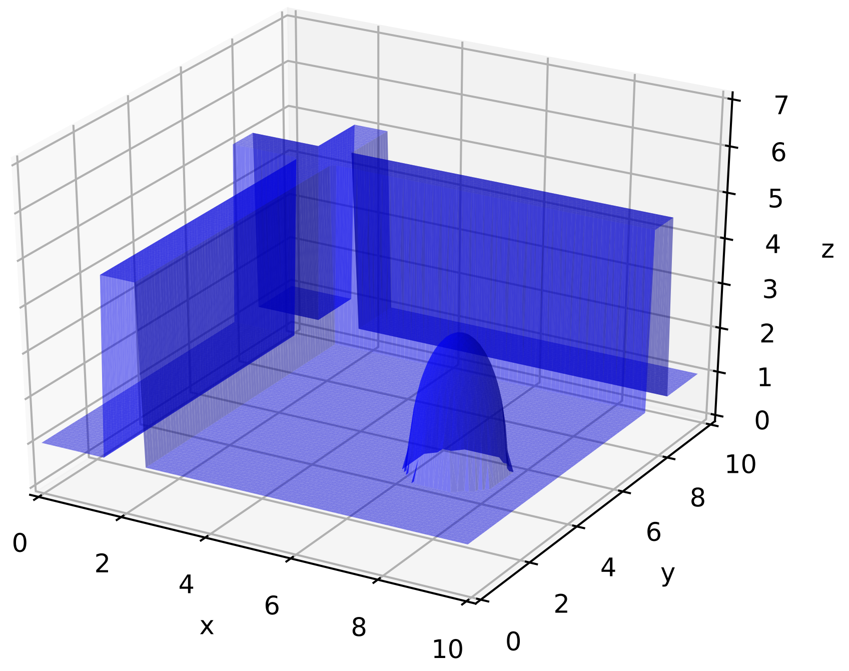

Figure 1: Visualization for simulating walls in an indoor environment

Figure 2 shows a visualization of a signal being sampled over a straight line while moving between the points (0,0) and (10,10) in the environment from Figure 1 above. We can see that as each wall is crossed, the mean (black line) of the signal as well as the variance (blue samples) increase.

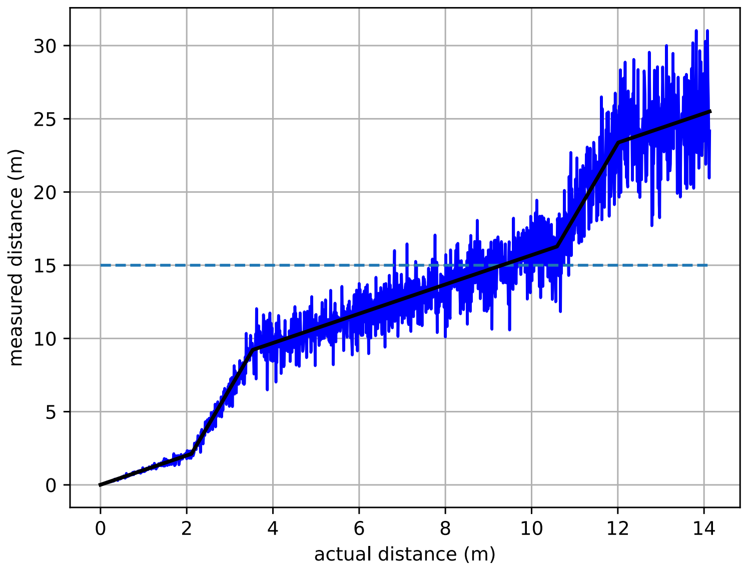

Figure 2: Visualization of noisy signals propagating over distance. Signals above the blue dashed line can be dropped.

The state deviation cost matrix (Q) and control input cost matrix (R) need to be specified when using LQR control. Q and R both need to be positive semi-definite matrices. Giving relatively large weight to the Q matrix causes the LQR controller to follow the trajectory tightly regardless of how much energy is needed. Giving relatively large weight to the R matrix causes the controller to limit energy at the expense of deviating more from the desired trajectory. The time horizon, T, also needs to be specified. Specifying an integer value for T calls the discrete time finite horizon LQR algorithm. Specifying 'inf' for the horizon calls the discrete time infinite horizon LQR algorithm. The trajectories are chosen by generating way-points between randomly selected points in the environment. It should be noted that it is the designer's responsibility to specify feasible trajectories.

Figure 3 shows a visualization of the simulator with 10 randomly placed beacons (large red dots) and an agent located inside of the filled sphere. The dotted circles in the figure represent the distance estimates from each transmitter at time *t*. The solid circles represent the expected measurements based on the previous belief state and the predictions based on the dynamics and control input of the system.

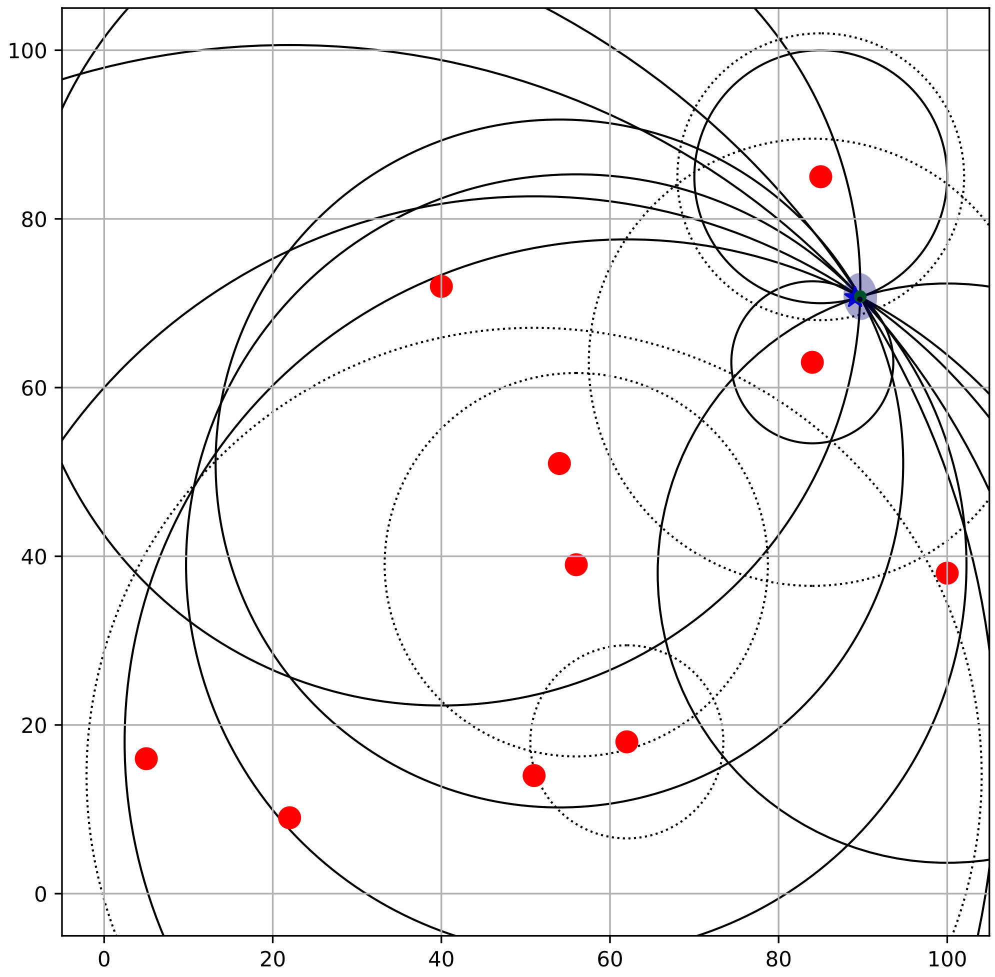

Figure 3: Visualization of simulated environment

To focus in on other key elements visualized in the simulator, we remove the display of the individual sensor measurements. Figure 4 shows another randomly generated configuration of nodes and we now notice a blue ellipse. The star represents the actual position of the agent. The green circle represents the estimated state via EKF. The filled sphere is a visualization of the error covariance matrix and gives us an intuition for the uncertainty in the state estimate based on the models. The small red circle represents the result of multilateration which only uses the last set of measurements to predict the state.

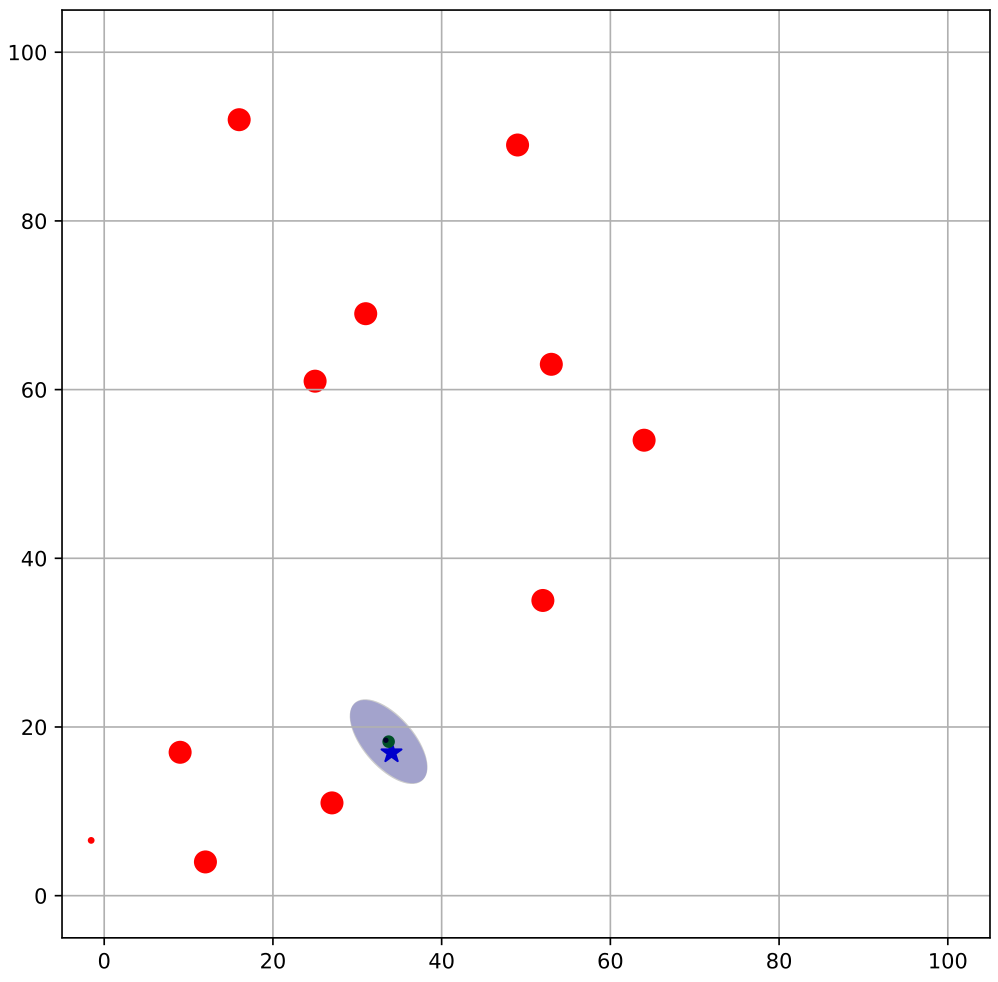

Figure 4: Visualization of simulation environment. Note the blue shaded error ellipse in the image above used to visualize the uncertainty matrix.

For testing the trained localization models, we move the agent through the simulated environment and feed the noisy beacon measurements (and control inputs when applicable) into the trained model. The prediction is compared against the known state and we compute the root mean squared error (RSME) to compare performance. Cumulative error histograms are also constructed to give a better visual intuition of how each algorithm performs.

3 INDOOR EXPERIMENTAL ENVIRONMENT

The testbed for this report is an indoor lab space and is built on the project in [13]. The lab consists of a network of transmitting beacons and receivers. The transmitters in these experiments use the iBeacon protocol and transmit the following information or have the information translated from the signal strength [3] [9]:

1) *Proximity UUID* or universally unique identifier which uniquely identifies one or more beacons as a certain type or from a certain organization
2) *major value* used to group related beacons that have the proximity UUID

3) *minor value* used to differentiate beacons with the same proximity UUID and major value
4) *RSSI:* The RSSI of the signal can be used to determine a distance measurement from the transmitter to the receiver.
5) *Proximity:* The signal is classified into one of the following ranges
    a. *Immediate – less than 1 meter*
    b. *Near – between 1 to 3 meters*
    c. *Far – greater than 3 meters*
    d. *Unknown – out of range or signal not received*

In the experimental lab scenario, we use Kontakt.io beacons. These beacons use Bluetooth Low Energy (BLE) wireless technology 2.4GHz RF. The beacons are stationary and fixed at known locations while the receivers are mobile. Each beacon has a range up to 70 meters. The battery life can last up to 9 months using the iBeacon profile with full power and 24-hour non-stop usage [14]. Also, the performance of each beacon varies and must be considered when converting RSSI values to distance estimates. There are several ways to do this conversion:

1) Individually sample each beacon at different ranges to build a conversion table from RSSI in decibels to distance in meters
2) Sample beacons of the same type and configuration and build a general conversion table like mentioned above
3) Apply path loss formula to convert the RSSI value to a distance estimate

A path loss model is described in [1] [6] that can be used to convert the RSSI value to distance estimates:

$$RSSI = -10nlog_{10}(d/d_0) + C \tag{9}$$

Table 3: Path-loss model variable descriptions

Variable	Description
n	Path-loss exponent
d	Distance from user to beacon
d_0	Reference distance (1 meter)
C	Average RSSI at 1 meter

Method 1 is the most tedious but provides the most accurate conversion. Method 2 sacrifices accuracy for individual beacons, but still aims to give a robust conversion based on the type of beacons and their configurations. Method 3 is dependent on the accuracy of the model being used to convert the RSSI values to distance estimates. These parameters for these models are often difficult to determine and only work well in open space. They also vary based on the configuration of the device. Much work has been done in developing path loss models for indoor environments but their accuracy in highly obstructed indoor environments can be problematic [10]. In our experiments, we found method 3 to be inconsistent and thus we avoided using it.

For most experiments in this project, we use manually constructed conversion tables for individual beacons when available, and resort to a generalized conversion table when a beacon does not have its own conversion table. The conversion tables are stored in JSON format. Each map contains a hash table of beacon major_minor keys and their corresponding conversion tables. The conversion tables are hash tables mapping RSSI values to distance in meters. The format is as follows:

```
{“map_name”: {“major_minor”: {"RSSI": distance, "RSSI": distance, ...},
              ...,
              “major_minor”: {"RSSI": distance, "RSSI": distance, ...}},
                             ...,
                             ...,
                             ...,
 “map_name”: {“major_minor”: {"RSSI": distance, "RSSI": distance, ...},
              ...,
              “major_minor”: {"RSSI": distance, "RSSI": distance, ...}}}
```

Figure 5: JSON posting REST service data format

The receivers collect visible iBeacon information and post in JSON format to a rest service. The localization algorithm then pulls the most recent information from the rest service. The available information to the solver for each received beacon signal then has the following items:

Table 4: REST service data formats

Label	Example Values
Timestamp	2018-04-06T20:42:40+00:00
UUID	F7826DA6-4FA2-4E98-8024-BC5B71E0893E
Major	3
Minor	8
RSSI	-93
Proximity	“Far”

The time stamp is used by the localization algorithm to verify that information is from the correct time frame. The major and minor values are used look up the beacon information (position, RSSI to distance conversion, map name). Finally, the RSSI value is used to come up with an estimate of the distance from the known location of the beacon.

Several types of receivers were used in the experiments and many more are available for use. The two primary receivers tested were Raspberry Pi’s and apple devices including iPhones and iPads [13]. Most Bluetooth 4.0+ enabled devices are compatible including modern laptops, tablets, and phones. Modern android devices are also compatible but were not used in these experiments.

Figure 6 below shows an overview of the system with data flow. The stationary nodes send iBeacon data to the mobile receivers. The mobile receivers relay this information to a

RESTful web service which the solver can access via URL. The solver requests the available information and posts the solutions to another RESTful web service which can be accessed by the users.

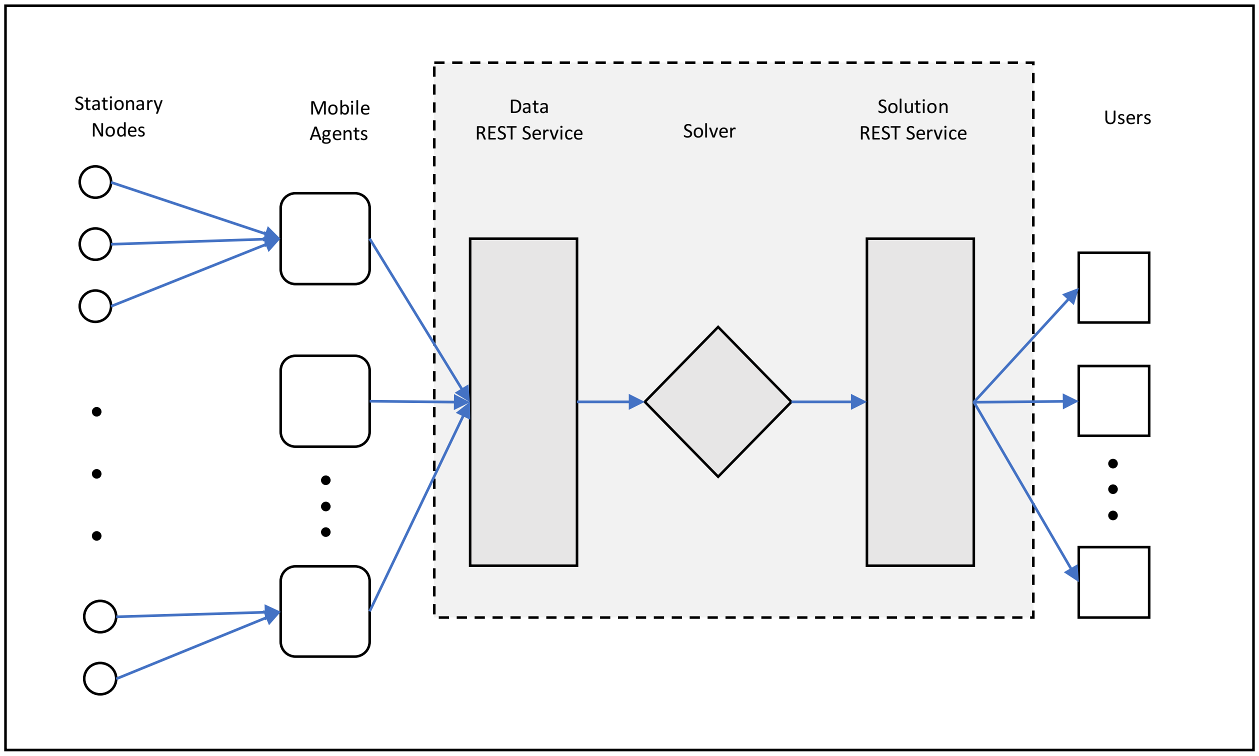

Figure 6: System diagram showing data flow

The main challenge in indoor localization is dealing with the many obstacles in the environment. There are several sources of interference or noise in the real-world setup. Some of these are due to the equipment itself and others are due to the environment. We will briefly list several sources of interference that are present in our indoor environment but this list is not exhaustive.

- Localization equipment: interference from other beacons
- Building structure and objects: multipath effect, walls, doors, bodies, etc [9]
- Other signals: cell phones, laptops, wireless routers, etc

The first important source of interference comes from the other beacons. This needs to be considered when deciding how to place the beacons as placing them too near to each other will cause signal interference. The second comes from the structure of the building itself and the objects within the building. Walls and other building infrastructure affect RSS readings and can even block signals completely. Even the human body can attenuate signals when standing between transmitter and receiver [9].This also needs to be considered when placing transmitters. The third disturbance that needs to be considered is interference from other electronics, especially other wireless devices. Placing transmitters and receivers near other devices with wireless hardware (i.e. Wi-Fi routers) can significantly affect accuracy.

These disturbances together create a highly complex environment for signal propagation making determining location via RSS challenging. And due to their dynamic nature, these complex indoor environments are constantly changing overtime. We provide two examples of real experimental environments that contain several of these obstructions. The first figure shows an open space environment in a larger building layout. The environment has several beacons spread out in range of the receiver. We can see that the solver is receiving data from each of the beacons inside the open space. There are also beacons outside of the open space in the adjacent hallways, but the signals are often obstructed by the walls and are not seen by the receiver.

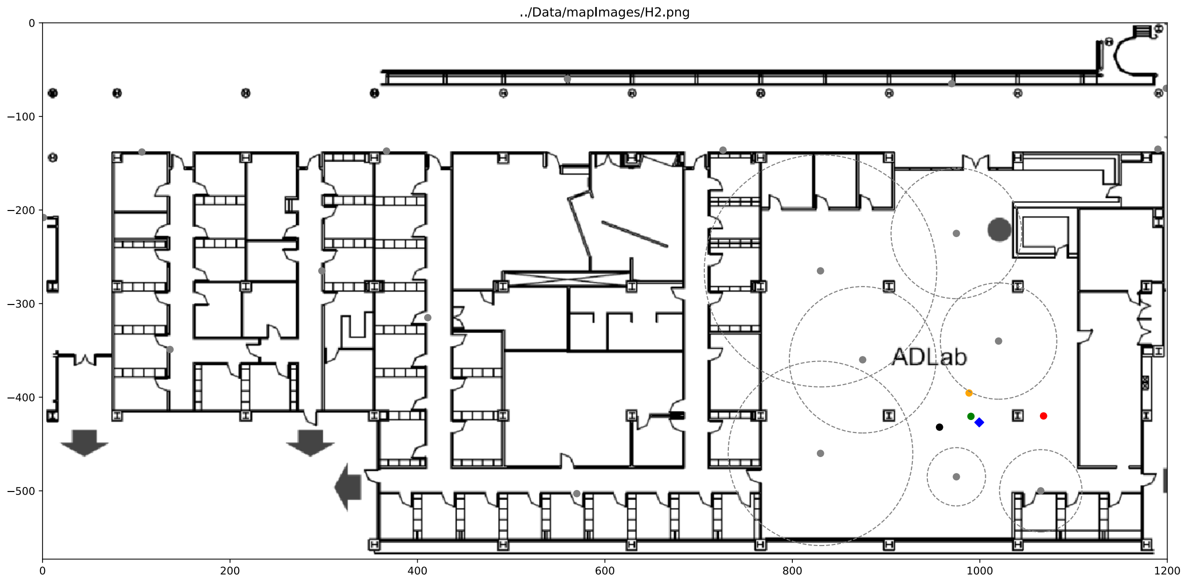

Figure 7: Experimental environment with open space and sparse obstruction of wireless signals [13]. The open region was used for the experiments. The other regions also have beacons but their signals are mostly obstructed.

The second figure shows a heavily obstructed hallway. Several beacons are spread out in each of the hallways, but only a few beacons are ever seen by the receiver.

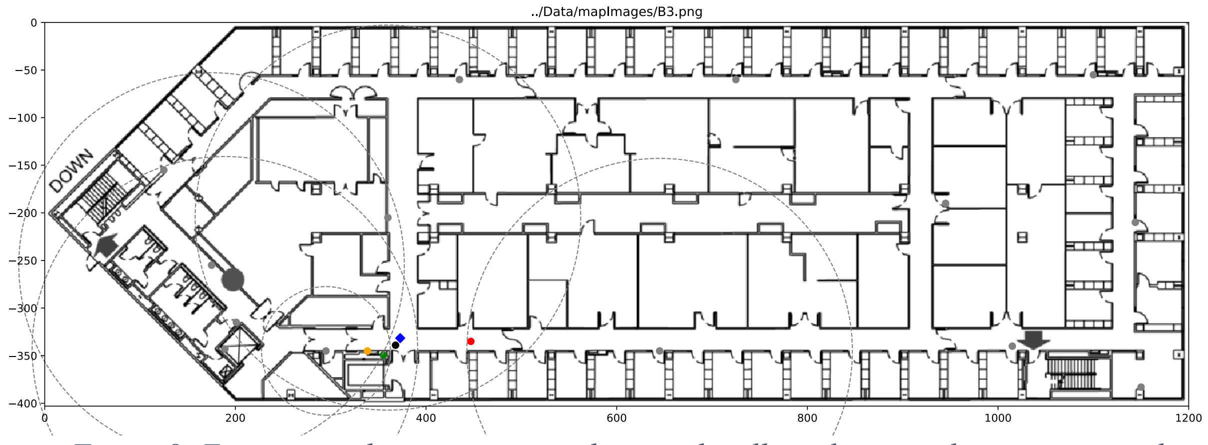

Figure 8: Experimental environment with several walls and rooms obstructing signal propagation [13]

These are two common scenarios for localization inside of buildings and we will compare the performance of our algorithms in both environments.

4 DESCRIPTION OF LOCALIZATION METHODS

4.1 MULTILATERATION LEAST SQUARES

Multilateration (MLAT) looks to determine the position of an agent based on distance estimates from 3 or more known locations. This approach gives the exact position when the distance measurements are exact. It also gives the optimal solution in a least squares sense using a single set of readings with additive Gaussian noise by minimizing the RMSE. However, in the presence of noise or interference MLAT methods can become highly inaccurate. Several adaptations of least squares exist that perform slightly better than ordinary least squares. An explanation of multilateration and the details for performing multilateration can also be found in [6].

Weighted least squares can be used to bias the optimization towards more reliable signals. The bias can put more emphasis on sensors that are inherently more accurate or sensors that are nearby the agent and encounter less interference or signal degradation. These multilateration based approaches use a single set of measurements to make a prediction and hence can provide a poor localization solutions for noisy environment. MLAT will be used as a nominal solution to compare against other algorithms.

MLAT begins with n sensors at known locations:

$$(x_i, y_i) \qquad i = 1, 2, ..., n$$

The distance from the agent to each sensor location can be denoted as:

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \qquad\qquad (i = 1, 2, ..., n)$$

Hence in vector notation:

$$\begin{bmatrix} (x_1 - x)^2 + (y_1 - y)^2 \\ (x_2 - x)^2 + (y_2 - y)^2 \\ \vdots \\ (x_n - x)^2 + (y_n - y)^2 \end{bmatrix} = \begin{bmatrix} r_1^2 \\ r_2^2 \\ \vdots \\ r_n^2 \end{bmatrix}$$

Expanding the terms on the left:

$$\begin{bmatrix} x_1^2 - 2x_1x + x^2 + y_1^2 - 2y_1x + y^2 \\ x_2^2 - 2x_2x + x^2 + y_2^2 - 2y_2x + y^2 \\ \vdots \\ x_n^2 - 2x_nx + x^2 + y_n^2 - 2y_nx + y^2 \end{bmatrix} = \begin{bmatrix} r_1^2 \\ r_2^2 \\ \vdots \\ r_n^2 \end{bmatrix}$$

And finally subtracting one row from all the others we can linearize the system (the subtracted row becomes the reference beacon):

$$2\begin{bmatrix} x_n - x_1 & y_n - y_1 \\ x_n - x_2 & y_n - y_2 \\ \vdots & \vdots \\ x_n - x_{n-1} & y_n - y_{n-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (r_1^2 - r_n^2) - d_{1,n}^2 \\ (r_2^2 - r_n^2) - d_{2,n}^2 \\ \vdots \\ (r_{n-1}^2 - r_n^2) - d_{n-1,n}^2 \end{bmatrix}$$

We can rewrite the system as:

$$As = b$$

We want to solve for s that minimizes the error in the above system:

$$s^* = arg \min_s ||As - b||_2$$

The solution is the ordinary least squares solution:

$$s = (A^T A)^{-1} A^T b \qquad (10)$$

$$s = A^\dagger b \qquad (11)$$

Note that multilateration requires computing the pseudo inverse. This computation is expensive and thus multilateration encounters scaling issues as the number of nodes becomes large. For some applications, pruning sensors from the computations can help reduce computation time. One example of implementing this is removing sensors with highly variable readings or sensors furthest from the previous state prediction.

4.2 EXTENDED KALMAN FILTERING

State estimation is a vital part of many applications in robotics, electronics, power systems, and countless more fields. Since most real-world applications don't provide direct access to the state, the state is generally estimated via the process model and sensor data. Sensors often do not have access to the true and full state of the system, but instead provide noisy and often incomplete measurements of the state. Still in some cases these measurements are enough to accurately determine the state of the system with a high level of accuracy.

Extended Kalman filtering (EKF) approach takes advantage of knowledge about the dynamics model, process noise, sensor model, and sensor noise. EKF is a recursive filter that estimates the internal state of a dynamical system. The filter uses a series of noisy measurements taken over time and Bayesian inference to make predictions that are often more accurate than predictions from a single measurement. The basic Kalman filter assumes linear dynamics and a linear sensor model with additive Gaussian noise.

$$x_{k+1} = Ax_k + Bu_k + v_k \qquad v_k \sim N(0, Q_k) \qquad (12)$$

$$z_k = h(x_k) + w_k \qquad w_k \sim N(0, R_k) \qquad (13)$$

The challenge in most applications of EKF is determining the covariance matrices accurately. When the system has linear dynamics and linear sensor model, EKF provides an optimal solution (ordinary Kalman filtering). EKF allows for non-linear dynamics and non-linear sensor models as seen in most real systems, but under these conditions no longer guarantees an optimal solution. Nevertheless, EKF is highly effective in many real applications. In this non-linear case, the dynamics and sensor models are locally linearized. For the purposes of this paper, the dynamics model is already linear and only the sensor model needs a local linear approximation.

$$H_k = \frac{\partial h(x)}{\partial x}|_{x=x^R}$$

Approximating using the Taylor expansion and replacing the first derivative term with the Jacobian of the sensor model we obtain:

$$z_k = h(x_k) + H_k(x_{k+1} - x_k) + w_k \qquad w_k \sim N(0, R_k) \tag{14}$$

If the noise in the process model is minute, our uncertainty will grow slowly as we move from our initial position. As we near a beacon location, the reading from the nearby beacon becomes more accurate and has lower variance. Hence, our known beacon locations can be used as landmarks to keep our uncertainty anchored. Therefore, in collecting data for training and testing our other algorithms, we sometimes randomly select beacon landmarks and move in linear trajectories to reach the next landmark. This requires beacon placement to be dense enough to maintain low uncertainty and keep the belief state from diverging from its true state.

The Kalman filter process has two stages: 1) the predict stage in which the state and covariance matrix are projected ahead and 2) the correction stage in which the Kalman gain is computed, the state estimate is updated with the sensor measurements, and the covariance matrix is updated.

Predict:

$$\bar{x}_{k+1} = f(x_k, u_k) \tag{15}$$

$$\bar{P}_{k+1} = A_k P_k A_k^T + Q_k$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$$

Correct:

$$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + V_k R_k V_k^T)^{-1}$$

$$x_k = \bar{x_k} + K_k(z_k - h(\bar{x}_k, 0)) \tag{16}$$

$$P_k = (I - K_k H_k)\bar{P}_k$$

4.3 ARTIFICIAL NEURAL NETWORK

Artificial neural networks can express complex non-linear functions as needed for accurate indoor localization. Neural networks can require large amounts of training data with increases in network volume. Their expressive power can be expanded by creating additional hidden layers and increasing the number of nodes in each hidden layer. For the purposes of indoor localization, neural networks can be trained directly on RSS measurements to do inference. In this project, the neural networks are implemented using google Tensorflow version 1.4.0. The basic structure of generic single and double layer neural networks is depicted in Figures 11 and 12 below:

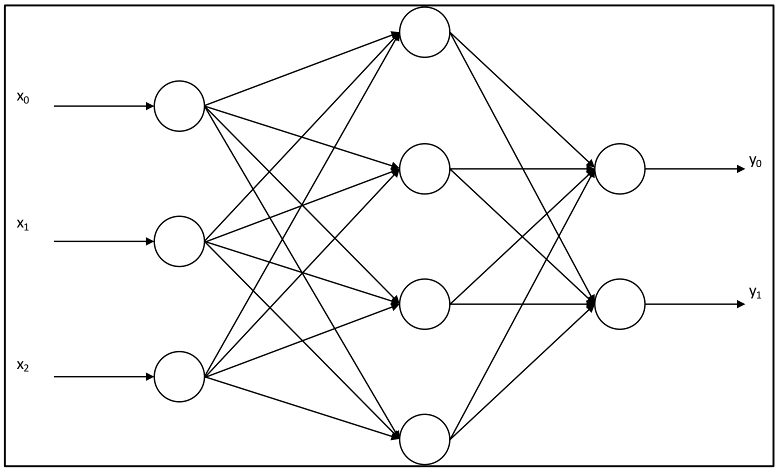

Figure 9: Generic Single Layer Feed-Forward Neural Network Architecture

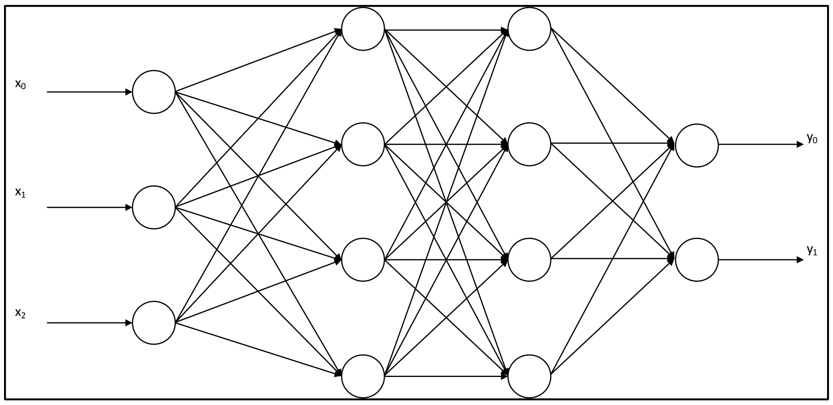

Figure 10: Generic Two Layer Feed-Forward Neural Network Architecture

The output of the first network is:

$$y = \sigma(Wx + b)$$

The outputs for the second network are:

$$h = \sigma(W_x x + b_x)$$
$$y = \sigma(W_h h + b_h)$$

The inputs to our network are the RSS values from each receiver. The labels to the training data inputs are the known (or estimated) locations. We train two networks. One with a single hidden layer of size 100 and another network with 2 hidden layers of size 100 and 10 respectively. Both layers apply sigmoid functions for the non-linearity. The output of both networks is the inferred position corresponding to the RSS measurements. The network is training with a momentum gradient descent optimizer and learning rate of .001 over 10000 iterations with batch size 100. Our network is constructed using Tensorflow [15].

The pitfall of this method is that a single set of measurements taken from all sensors can contain significant noise. Hence, it is often difficult to differentiate between multiple locations from a single measurement. On top of this, there is no memory of an internal state and thus no intuition of the internal dynamics. Hence, a small change in sensor readings can results in an unrealistically large change in inferred state from one prediction to the next. The next network architecture seeks to address these concerns.

4.4 RECURRENT NEURAL NETWORK

Recurrent neural networks (RNN), like feed-forward neural networks, can be used to learn complex non-linear functions for inference. RNNs have an added advantage that they have memory of an internal state and can be trained to do inference for sequential data. In our scenario, this allows us to not only do inference from RSS fingerprints to the state, but to also include controls/IMU data as inputs to infer how they affect the state of the system over time. Hence, RNNs can update their hidden state based on internal dynamics, control inputs, and measurements (like the EKF). They have the added advantage of having more flexibility in learning complex and highly non-linear models.

RNNs, as depicted in Figure 11 below, allow future cells to obtain information from previous cells in a recurrent chain. Each cell has its own set of inputs including the RSS values at the current time step and information passed from the internal state of the previous cell. Each cell makes its own prediction of the state at the current time and passes information of its internal state to the following recurrent cell.

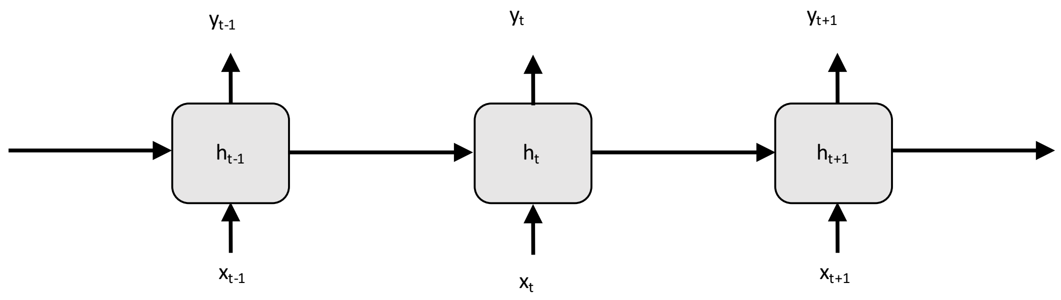

Figure 11: Visualization of Unfolded Recurrent Neural Network (RNN)

We implement the RNN using Tensorflow [15]. Our RNN implementation uses 10 recurrent cells with sigmoid non-linearity. The input to each cell $X_t \in R^{N,10}$ consists of a sequence of Nx10 RSS readings taken at consecutive time instances. The internal state of each cell is a function of the input and previous internal state:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{17}$$

The output of each cell is a function of the internal state:

$$y_t = \sigma_y(W_y h_t + b_y) \tag{18}$$

We optimize the network using a momentum optimizer (Tensorflow's RMSPropOptimizer) [15]. The cost function is the MSE between the actual states and the networks predictions. The outputs of the network are the inferred positions at times [t, t+9]. Basic RNN cells are limited to short term memory as they only pass information from a cell to its directly succeeding cell. Long-short Term Memory (LSTM) cells have added advantage over basic RNN cells as they also learn to weight information over time and have added structure to pass this information along to all future cells. The structure of these cells is much more complex and we will not cover the details in this report. For more details refer to [16].

Bidirectional recurrent neural networks (BRNN), as depicted in Figure 12 below, allow cells to share information between preceding and succeeding cells. These types of architectures along with LSTM cells are commonly used for natural language processing and other applications where past and future context are important for inferring the current context. BRNN also allow for an additional layer of depth by stacking recurrent layers (forward and backward) on top of each other. The use of these types of architectures in applications of indoor localization, to the best of my knowledge, has not been explored but seems promising in theory and in simulation.

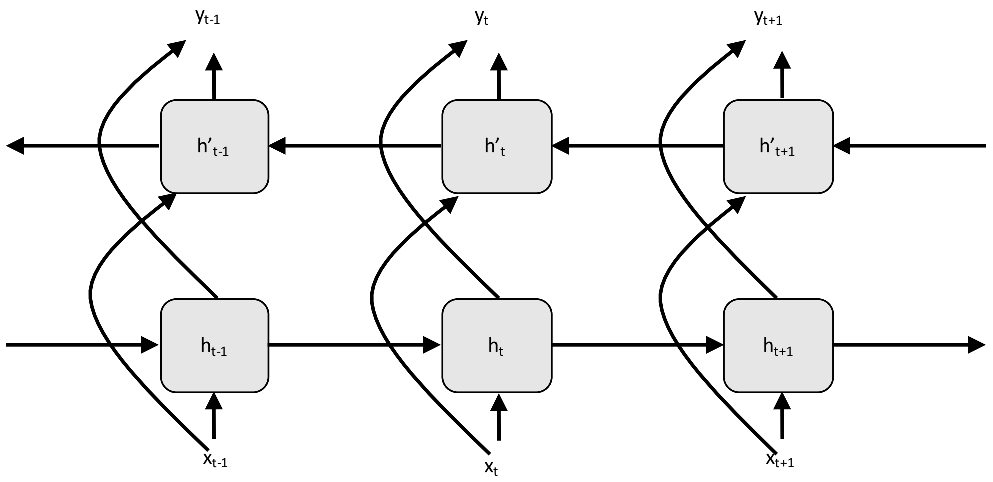

Figure 12: Visualization of unfolded bidirectional recurrent neural network (BRNN)

We implement our BRNN using Tensorflow [15]. In our architecture, the inputs to the BRNN are the RSS measurements at time t and the control inputs at time t for a sequence of length 10. We again optimize the network using the same momentum optimizer (Tensorflow's RMSPropOptimizer) [15]. The cost function is again the MSE between the actual states and the networks predictions. The outputs of the network are the predictions of the positions at time t. For details on the implementation of BRNN refer to [17].

5 RESULTS

5.1 SIMULATED RESULTS

The following results are derived by comparing the described algorithms in the simulated environment. The trajectories are generated by moving between randomly selected locations in the environment and colleting noisy measurements from the sensors along the trajectory. Each algorithm is then applied to the noisy measurements and the predicted state is compared against the actual state. The results are presented as cumulative histograms of the prediction error in meters.

The first set of results shown in Figure 13 compares the algorithms in the absence of noise. This gives us a baseline for how well we can expect the algorithms to perform. Multilateration produces perfect results as expected in the absence of noise.

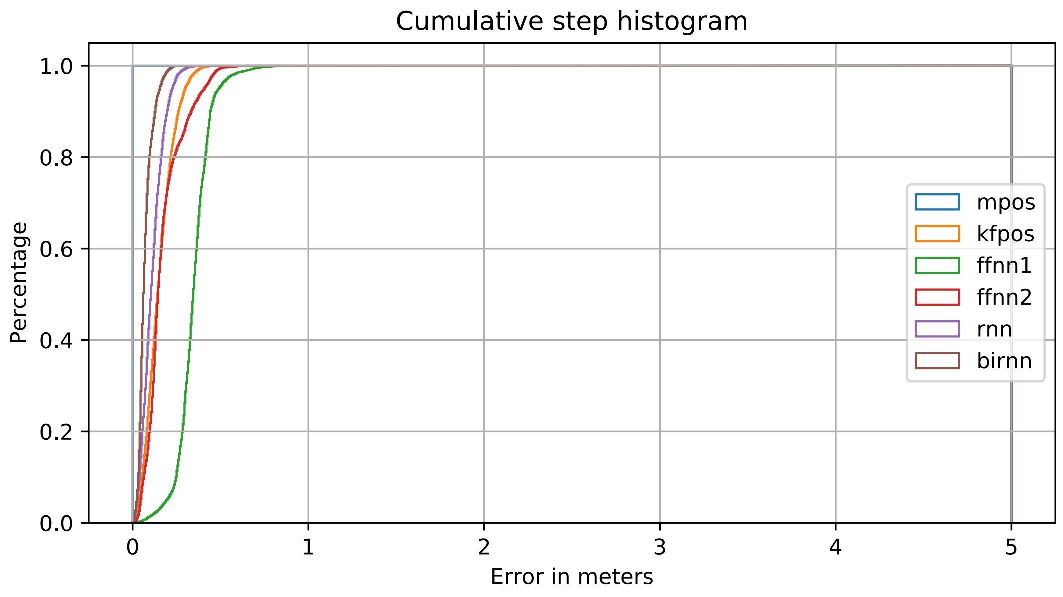

Figure 13: Performance comparison of noiseless simulation. Note: Multilateration produces exact solution in the absence of noise.

The second set of results shown in Figure 14 are taken from an environment with state independent noise. We now see that multilateration quickly becomes the lowest performing algorithm, with the feed-forward neural networks performing significantly better, and EKF and the recurrent neural networks performing the best.

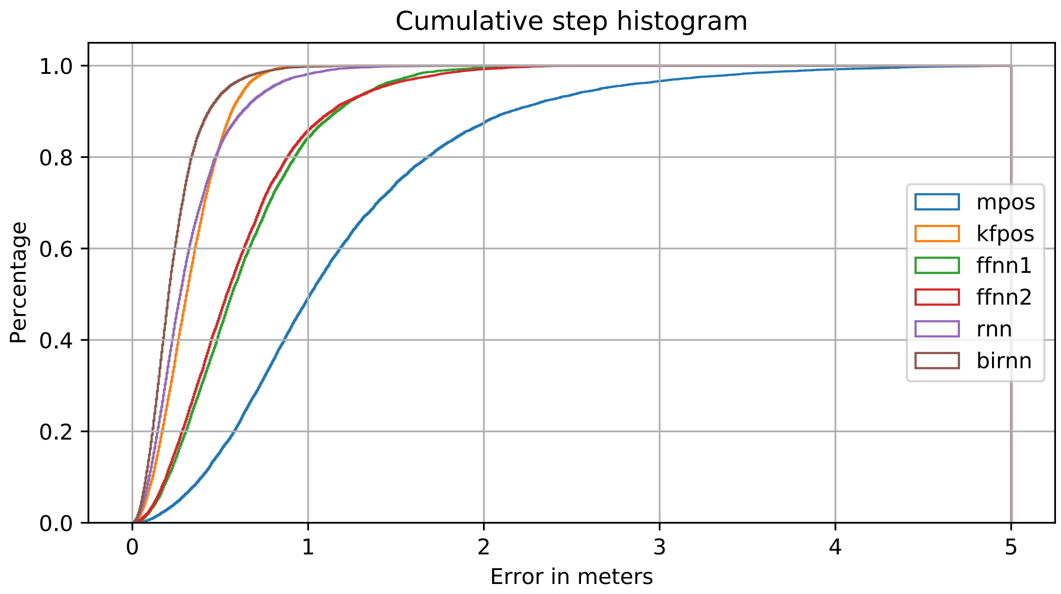

Figure 14: Performance comparison with state independent noise. Note: Multilateration performance significantly degrades with noise due to the discussed pitfalls.

The next set of results shown in Figure 15 are taken from an environment where sensor noise scales with distance. Performance for all the algorithms significantly degrades but the relative performance trend is the same.

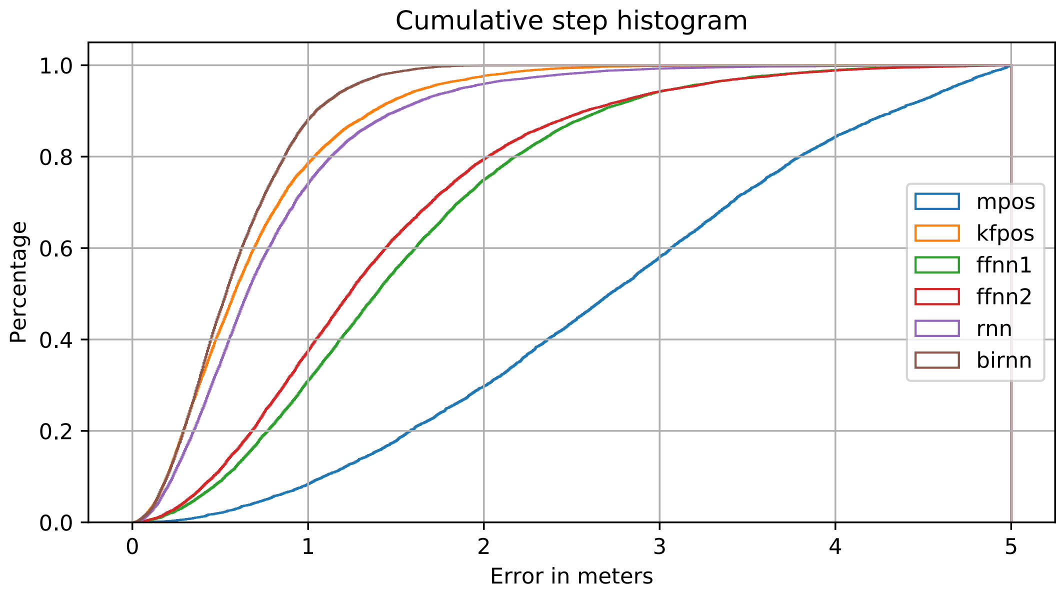

Figure 15: Performance with noise scaling with distance from node

The final two sets of results shown in Figure 16 and Figure 17 are taken from two environments each with several simulated objects affecting signal propagation. Again, the same relative performance is seen between algorithms.

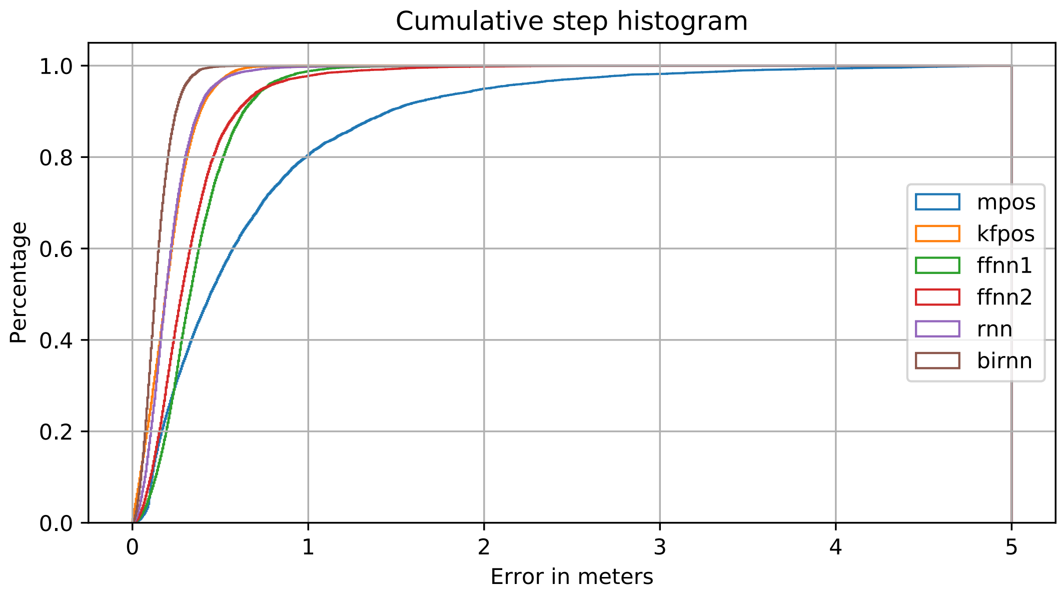

Figure 16: Performance in designed simulated environment 1

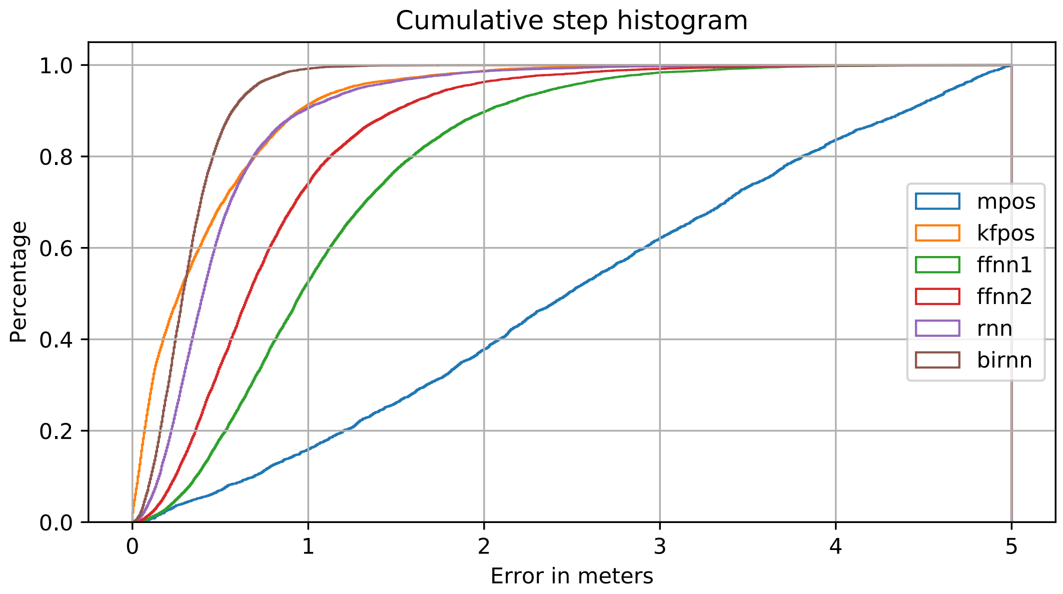

Figure 17: Performance in designed simulated environment 2

5.2 EXPERIMENTAL RESULTS

The experimental results do not include results for neural networks based methods. The experimental results pre-date the simulated results. They do include an additional algorithm that matches the EKF result to the nearest valid point on the map. For real environments, this ensures that we do not present users with solutions that are inside of walls, outside of buildings, or in other invalid locations. The experimental results are taken from two different indoor environments. The results for the first test environment are shown in Figure 18 and come from a heavily obstructed hallway environment. The results show that EKF significantly outperforms the previous nearest neighbor template matching method and multilateration. Adding nearest neighbor matching also reduces the error slightly.

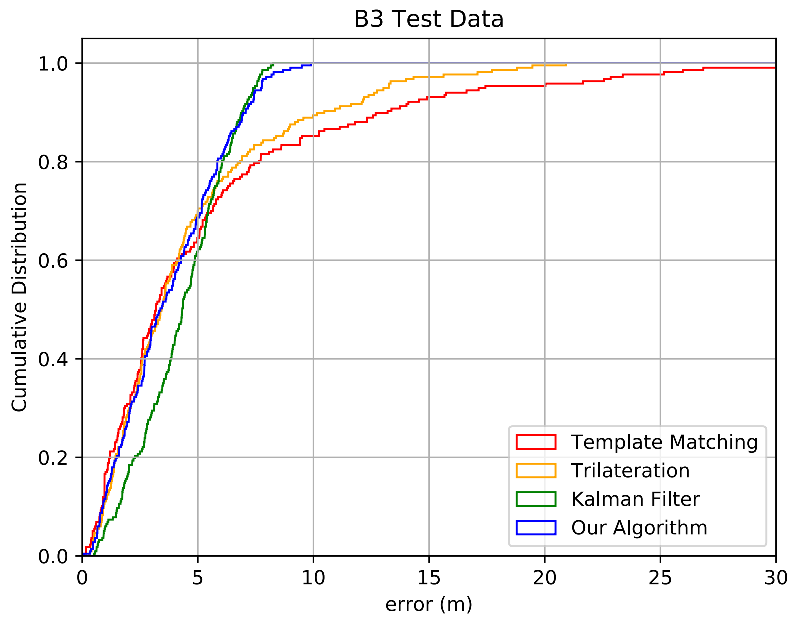

Figure 18: Results from hallway Environment

The second experimental environment has numerous sensors in an open space where the agent can freely move in all directions. The results are shown in Figure 19 with EKF significantly outperforming nearest neighbors template matching and multilateration.

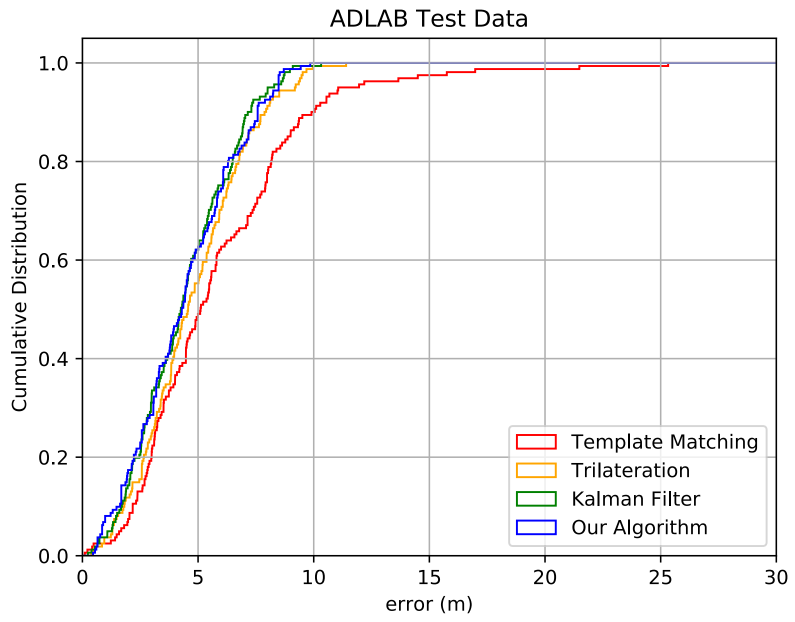

Figure 19: Results from open space environment

CONCLUSIONS & FUTURE WORK

The results of the simulations show promising improvements in localization accuracy when using RNN and BRNN for inference. From the results, we can see that the benefits of using more complex algorithms like EKF and RNN/BRNN become apparent in highly obstructed environments. Further real-world experiments need to be run in different environments to verify the robustness of these algorithms. The provided real-world experimental results have a significant amount of added error due to the challenge of manually collecting accurate ground truth at the time. Manual collection of ground truth is incredible tedious and error prone. We are currently working to automate the data collection process with highly accurate (<20cm error) ground truth data gathered using an UWB based localization system. This will allow us to re-run these experiments with much larger and more accurate datasets.

REFERENCES

[1] S. Kumar, R. Hegde, *A Review of Localization and Tracking Algorithms in Wireless Sensor Networks*.

[2] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” IEEE Transactions on Systems, Man, and Cybernetics. 2007.

[3] P. Kriz, F. Maly, and T. Kozel, *Improving Indoor Localization Using Bluetooth Low Energy Beacons*. Mobile Information Systems, 2016.

[4] J. J. Diaz, R. d. A. Maues, R. B. Soares, E. F. Nakamura, and C. M. Figueiredo, *Bluepass: An Indoor Bluetooth-based Localization System for Mobile Applications,* in Computers and Communications (ISCC), 2010 IEEE Symposium on, pp. 778–783, IEEE, 2010.

[5] Faheem Zafari, Athanasios Gkelias, Kin K. Leung, *A Survey of Indoor Localization Systems and Technologies.*

[6] Y. Liu, Z. Yang, X. Wang X, Et al, *Location, localization, and localizability*. Journal of Computer Science and Technology. Mar. 2010

[7] R. Bruno and F. Delmastro, “Design and Analysis of a Bluetooth-based Indoor Localization System,” in IFIP International Conference on Personal Wireless Communications, Springer, 2003.

[8] M. Malajner, P. Planinšič, D. Gleich, "UWB ranging accuracy," 2015 International Conference on Systems, Signals and Image Processing (IWSSIP), London, 2015, pp. 61-64.

[9] Getting Started with IBeacon version 1.0. 2014 [Online]. URL: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf. Accessed: 19-July- 2018

[10] Juan Manuel Castro-Arvizu1, Jordi Vila`-Valls2, Ana Moragrega2, Pau Closas3 and Juan A Fernandez-Rubio, *Received signal strength–based indoor localization using a robust interacting multiple model–extended Kalman filter algorithm*.

[11] P. Kumar, L. Reddy, and S. Varma, *Distance measurement and error estimation scheme for RSSI based localization in wireless sensor net- works*, in Wireless Communication and Sensor Networks (WCSN), 2009 Fifth IEEE Conference on. IEEE, 2009.

[12] L. Gogolak, P. Szilveszter, K. Dragan. *Neural Network-based Indoor Localization in WSN Environments*. 2013.

[13] G. Flores, T. Griffin, D. Jadav, *An Ibeacon Training App for Indoor Fingerprinting.* IEEE International Conference on Mobile Cloud Computing, Services, and Engineering 2017.

[14] Double Battery Beacon. Kontakt.io Store. URL: https://store.kontakt.io/our-products/30-double-battery-beacon.html

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[16] W. Zaremba, I. Sutskever, O. Vinyals, *Recurrent Neural Network Regularization*. CoRR 2014. http://arxiv.org/abs/1409.2329

[17] A. Graves, A. Mohamed, G. Hinton, *Speech Recognition with Deep Recurrent Neural Networks*. CoRR 2013. https://arxiv.org/abs/1303.5778

ACKNOWLEDGEMENTS
Claire Tomlin, PhD (Primary Advisor)
Murat Arcak, PhD (Second Reader)
Divyesh Jadav, PhD (Advising, funding)
Tom Griffin, PhD (Experimental Infrastructure, advising)
German Flores, PhD (Experimental Infrastructure, advising)