Proceedings of the 19th IASTED International Conference
Parallel and Distributed Computing and Systems
November 19–21, 2007, Cambridge, MA, USA
ISBN Hardcopy: 978-0-88986-703-1 / CD: 978-0-88986-704-8

COMPARISON OF MLP NEURAL NETWORK AND KALMAN FILTER FOR LOCALIZATION IN WIRELESS SENSOR NETWORKS

Ali Shareef, Yifeng Zhu*, Mohamad Musavi, and Bingxin Shen
Department of Electrical and Computer Engineering
University of Maine
Email: {ashareef, zhu*, musavi, bshen}@eece.maine.edu

ABSTRACT
Localization with noisy distance measurements is a critical problem in many applications of wireless sensor networks. Different localization algorithms offer different tradeoffs between accuracy and hardware resource requirements. In order to provide insight into selecting the best algorithm that optimizes this tradeoff, this paper evaluates the accuracy, memory, and computational requirements of two approaches that may be taken in localization: neural networks and Kalman filters. In this paper, we quantitatively compare the localization performance of a *Multi-Layer Perceptron* (MLP) neural network, PV, and PVA models of the *Extended Kalman filter*. Our experimental results show that the MLP neural network has weaker self-adaptivity than the Extended Kalman filters; however, the MLP can potentially achieve the highest localization accuracy and requires the least amount of computational and memory resources.

KEY WORDS
Localization, Sensor Networks, Neural Networks, and Kalman

1 Motivations

Localization arises repeatedly in many location-aware applications such as navigation, autonomous robotic movement, and asset tracking [1, 2]. Analytical localization methods include *triangulation* and *trilateration*. Triangulation uses angles, distances, and trigonometric relationships to locate the object. Trilateration, on the other hand, uses only distance measurements to identify the position of an object. Figure 1 gives a simple example of this method. Using three reference points S_i $(i = 1, 2, 3)$ with known coordinates and distances d_i $(i = 1, 2, 3)$ to the target object, the object can be uniquely located at the intersecting point of the three circles.

However, in the event that the distance measurements are noisy and fluctuate, the task of localizing becomes difficult. This can be seen in Figure 2. With fluctuating distances, regions within the circles become possible locations for the tracked object. In this case, rather than the object being located at a single point at the intersection of the circles as in Figure 1, the object can be located anywhere in the dark shaded region in Figure 2.

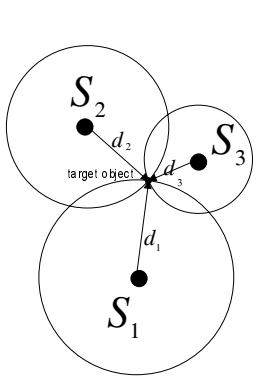

Figure 1. Trilateration

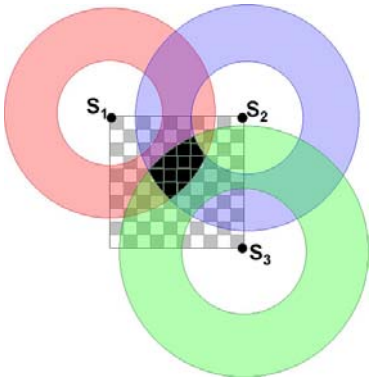

Figure 2. Trilateration with Noise

This uncertainty due to measurement noises renders analytical methods almost useless. Localization methods capable of accounting for and filtering out the measurement noises are desired. Which is why neural networks are very promising in this area.

The method by which the distance measurements are carried out determines the sources of noise in these measurements. Typically devices known as "beacons" are placed at known locations and emit either radio or acoustic signals or both. It is possible for a "mobile node" to determine the distance to a beacon by using properties of these signals such as the signal strength of the RF signal, Received Signal Strength (RSS) [3]. Other methods utilize both RF and acoustic signals by computing the time difference between an RF pulse and an acoustic pulse generated by a beacon [4, 5, 2, 6]. RF signals travel at the speed of light and the time it takes for a RF signal to get to a mobile node is almost instantaneous and can be considered zero while the time it takes for an acoustic signal such as ultrasound is much longer. In the case of the Crickets [2, 4], the RF and ultrasound signals are emitted simultaneously by the beacons. The mobile node computes the distance by using the time it takes for the first instance of the acoustic pulse to reach the sensor after the RF signal. However, wave reflection is common to both RF and acoustic signals and it is possible that the mobile node erroneously identifies a pulse due to the reflected wave of the original pulse as

a new pulse [5]. This, of course, results in skewed distance measurements.

Another consideration that must be made when selecting a localization method is the amount of computation that is required. It must be borne in mind that these methods will be implemented on embedded systems with limited computation and memory resources. Different methods with different levels of accuracy require varying amounts of resources. A trade-off must be made that balances the timely computation of the position and the desired accuracy necessary for the application. Disregard to this principal may result in the time required to compute the position exceeding the time when the information is needed.

In this paper, the accuracy, robustness, and efficiencies of localization utilizing Kalman filters and neural networks will be explored. The contributions of this paper are the utilization of a *Multi-Layer Perceptron* (MLP) neural network for localization, which to our knowledge has not been done before. We will also compare the performance of the neural network with two variants of the Kalman filter. In addition, we will also examine the computation complexity and memory usage of all of these methods.

The rest of the paper is organized as follows. Section 2 describes relevant localization methods proposed earlier. Section 3 introduces the principals of Kalman filters and neural networks. The experiment by which these methods will be compared is given in Section 4. The results will be discussed in Section 5, and concluding comments will be made in Section 6.

2 Related Work

The Active Badge Location System [6] is often credited as one of the earliest implementations of an indoor sensor network used to localize a mobile node [2]. Although this system, utilizing infrared, was only capable of localizing the room that the mobile node was located in, many other systems based on this concept have been proposed.

The Bat system [5, 7], much like the Active-Badge System, also utilizes a network of sensors. A central controller broadcasts an RF query to a mobile node and resets the serially linked receivers at the same time. The mobile node responds by emitting an ultrasonic pulse which are picked up by the receivers. The time it takes for the ultrasound pulse to reach individual receivers indicates the distance the mobile node is from the receivers and the position of the mobile node can then be trilaterated.

Researchers at MIT have utilized similar concepts from the Bat System in their Cricket sensors, albeit using a more decentralized structure. However, one draw back to the Crickets is the risk of collisions during the RF and Ultrasound transmissions between different beacons. The Cricket Location System [4] uses a hybrid approach involving the use of an Extended Kalman filter, Least Square Minimization to reset the Kalman filter during the Active state, and Outlier Rejection to eliminate bad distance readings.

Other researchers at MIT have proposed another method of localization utilizing the Cricket system exploiting properties of robust quadrilaterals to localize an ad-hoc collection of sensors without the use of beacons [8].

It is also possible to localize optically as in the HiBall head tracking system [9]. Arrays of LEDs flash synchronously, and cameras capture the position of these LEDs. The system utilizes information about the geometry of the system and computes the position.

Localization using signal strength of RF signals has been studied extensively, [10, 11, 12, 13] are all examples of methods that were devised using this approach. Neural networks, however, have not been used extensively in this area. There has been some research conducted by Chenna et al in [14]. However, Chenna et al, restricted themselves to comparing Recurrent Neural Networks (RNN) to the Kalman Filter. We would like to go further and compare the MLP neural network with the performance of the Kalman filters.

3 Introduction to Localization Algorithms

3.1 Kalman Filter

One technique of localization is with the use of the Kalman Filter (KF) [15]. The Kalman filter is an iterative approach that uses prior knowledge of noise characteristics to account for and filter out the noise. However, problems arise when attempting to model noise. Attempts at measuring noise are only approximations and do not indicate the real distribution of the noise. The Kalman filter can only be used for linear stochastic processes and for non-linear processes the *Extended Kalman Filter* (EKF) [16, 17] must be used. The assumption with these two methods is that the process and noise measurements are independent, white, and with normal probability.

There are different parameters that the EKF can use in modeling the trajectory of a moving object. It is possible to model the motion of an object using just the state of the X and Y position to obtain the *P Model*. The velocity can also be incorporated in the state in addition to the position to form the *PV model*. Of course, if acceleration is included also, this results in the *PVA model*.

The distances returned by the sensors can be related to the position of the object using the distance formula (1):

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \quad (1)$$

where $i = 1, 2, 3$

A way of modeling motion is by setting up a linear system composed of the kinematics equations for each dimension of tracked motion. The following example of a linear system describes an objects two-dimensional motion using the position, velocity, and acceleration (PVA).

State equation:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \ddot{x}_{k+1} \\ \ddot{y}_{k+1} \end{bmatrix} = A \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \\ \ddot{x}_k \\ \ddot{y}_k \end{bmatrix} + B \begin{bmatrix} u_{xk} \\ u_{y_k} \end{bmatrix} + Q \quad (2)$$

The state transition matrix A arises from the respective kinematics equations. For a PVA model, A becomes:

$$A = \begin{bmatrix} 1 & 0 & T & 0 & \frac{1}{2}T^2 & 0 \\ 0 & 1 & 0 & T & 0 & \frac{1}{2}T^2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

u_{xk} and u_{y_k} are the inputs to the system, and B is the input matrix. However, the input kinematics parameters of the moving object to be tracked are not known so the u_k terms and B can be dropped from the linear system. Also, the inputs to this system are distance measurements z_k. These distance measurement will be used to update the state of the object as given in step 4 of the Kalman Filter Equations as given in Algorithm (1).

Q is the process noise covariance matrix that accounts for the unmodeled factors of the system that will be treated as random noise. For example, in the systems of equations above, while the change of velocity is accounted for by acceleration the change in acceleration is not considered. The contribution of this effect to the state is accounted for as random noise. See [18] for a more in depth discussion. In this example, q is the standard deviation of the acceleration noise as given in [19]. Q is given as follows:

$$Q = \begin{bmatrix} \frac{\delta T^5}{20} & 0 & \frac{\delta T^4}{8} & 0 & \frac{\delta T^3}{6} & 0 \\ 0 & \frac{\delta T^5}{20} & 0 & \frac{\delta T^4}{8} & 0 & \frac{\delta T^3}{6} \\ \frac{\delta T^4}{8} & 0 & \frac{\delta T^3}{2} & 0 & \frac{\delta T^2}{2} & 0 \\ 0 & \frac{\delta T^4}{8} & 0 & \frac{\delta T^3}{2} & 0 & \frac{\delta T^2}{2} \\ \frac{\delta T^3}{6} & 0 & \frac{\delta T^2}{2} & 0 & q\delta T & 0 \\ 0 & \frac{\delta T^3}{6} & 0 & \frac{\delta T^2}{2} & 0 & q\delta T \end{bmatrix} \quad (4)$$

As described in [17], the three measured distances d_k where $k = 1, 2, 3$ given the locations of the three beacons (x_i,y_i) where $i = 1, 2, 3$ can be used to relate the location of the object (x_k,y_k) to the distances using the following equation:

$$d_{i,k} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} + \tilde{d}_{ik} \quad (5)$$

where $\tilde{d}_{ik}$ is the error in the measurement. The equation (5) can be expressed as (6).

$$\begin{bmatrix} d_{1k} \\ d_{2k} \\ d_{3k} \end{bmatrix} = H \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \\ \ddot{x}_k \\ \ddot{y}_k \end{bmatrix} + \begin{bmatrix} \tilde{d}_{1k} \\ \tilde{d}_{2k} \\ \tilde{d}_{3k} \end{bmatrix} \quad (6)$$

where H is the measurement matrix that relates the current state to the output. Since the output equations (5) are nonlinear, the Jacobian needs to be used.

$$H = \begin{bmatrix} \frac{\partial d_1}{\partial x} & \frac{\partial d_1}{\partial y} & 0 & 0 & 0 & 0 \\ \frac{\partial d_2}{\partial x} & \frac{\partial d_2}{\partial y} & 0 & 0 & 0 & 0 \\ \frac{\partial d_3}{\partial x} & \frac{\partial d_3}{\partial y} & 0 & 0 & 0 & 0 \end{bmatrix} \quad (7)$$

where

$$\frac{\partial d_i}{\partial x} = \frac{x - x_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} \quad (8)$$

$$\frac{\partial d_i}{\partial y} = \frac{y - y_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2}} \quad (9)$$

for $i = 1, 2, 3$

Using the formulation of the problem as described above, the following equations [15] can be evaluated iteratively to track an object. The distance measurements z_k are used in step 4 to update the state estimate.

Algorithm 1 Kalman Filter Equations

1: Project the state ahead: $X_k{}^- = AX_{k-1} + BU_k$
2: Project the error covariance ahead: $P_k{}^- = AP_{k-1}A^T + Q$
3: Compute the Kalman gain: $K_k = P_k{}^- H^T (HP_k{}^- H^T + R)^{-1}$
4: Update estimation with measurements: $X_k = X_k{}^- + K_k(z_k - HX_k{}^-)$
5: Update the error covariance: $P_k = (I - K_k H)P_k{}^-$
6: Repeat and go to Step 1.

In this paper, the performance of both the PV and PVA EKF will be compared. The benefit of one over the other depends upon the characteristics of the motion of the object. A system modeled using PV will tend to work when velocity is mostly constant. PVA on the other hand works better when the acceleration is mostly constant [18].

3.2 Neural Networks

Neural networks are modeled after biological nervous systems and are a network of interconnections between nodes called "neurons" with activation functions. Different classes of neural networks can be obtained by varying the activation functions and the structure of the weighted interconnections between the neurons. The MLP network is "trained" to approximate a function by repeatedly passing the input through the network. The weights of the interconnections are modified based on the difference between the desired output and the output of the neural network. The final weights of the MLP network is entirely dependent upon

the initial weights. Finding the set of weights that result in the best performance is ultimately through trial and error. Nevertheless, the power of both MLP neural networks lies in the fact that they can be used to model very complicated relationships easily. Detailed description is given in [20].

One major benefit of a neural network is that prior knowledge of the noise distribution is not required. Noisy distance measurements can be used directly to train the network with the actual coordinate locations. The neural network is capable of characterizing the noise and compensating for it to obtain the accurate position.

4 Experiment Design

We will explore the two approaches described using MIT's Cricket sensors [21]. However, as was discussed, the use of ultrasound introduces noise. The distance measurements returned by the sensors fluctuate often. During our experiment, the measured distances used for training and testing varied by as much as 32.5 *cm*. The distances from each of the beacons to the mobile node will be input to the MLP and RBF neural networks and the PV and PVA model of Kalman filters. Each will output positions that it estimates corresponds to the distance measurements. We will simulate the two dimensional motion of an object by collecting distance measurements of the mobile node while moving it in a network composed of Cricket sensors.

However, before any simulation can take place, training data must be collected to train the neural network. A tile floor provided a very regular grid of 30 *cm* in size upon which the sensors could be accurately located.

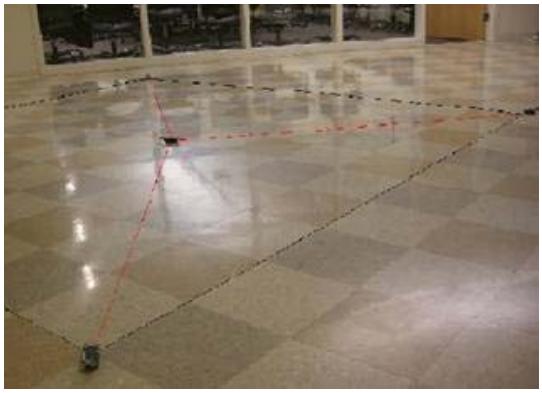

Figure 3. Experiment Test Bed

Using a grid of 300 $cm \times 300\ cm$, beacons were placed at positions $(0, 300)$, $(300, 300)$, and $(300, 0)$ of the grid as shown in Figure 4. The training data was collected by placing the mobile node at each intersection of the tiles and collecting distance measurements to each of the three beacons S_1, S_2, and S_3. The distance measurements to the beacons fluctuated constantly, and by collecting more than one set of distances to each of the beacons, we were able to capture the noise in the system. By training the

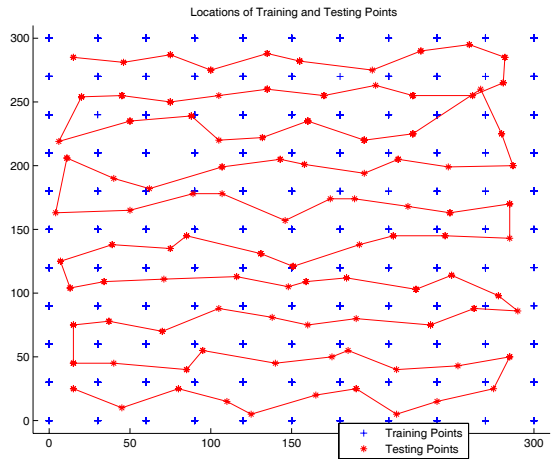

Figure 4. Locations of Training and Testing Data Collected

neural networks with these multiple sets of fluctuating distances to beacons for each position, the accuracy of the neural networks improved as it became capable of "filtering" out the noise in the distance measurements–just as the PVA Kalman filter utilizes the standard deviation of the acceleration noise to "correct" its estimates. The known locations for which the training distance measurement were collected are marked using '+' signs in Figure 4. For each of the 121 position, several sets of distances measurements were collected resulting in a total of 1521 sets of distance measurements for training the networks.

Once the training data was collected, the testing data was collected by moving the mobile node through the sensor network following a more random path. Again the distances, for each known positions were collected. This is indicated using the "$*$" sign in Figure 4. This data allows us to test the accuracy of the localization of the neural networks and the Kalman filters.

In the case of the MLP neural network, the distances will be input to the network and it will output the estimate of the X and Y coordinates location for those distances. In the case of the Kalman filters, the distances will be input as z_k in step 4 of the Kalman filter equations and used to update the state estimate as mentioned before.

The MLP neural network is a two-layer network composed of nine nodes in the first layer and three nodes in the output layer. The nodes in the first layer use the hyperbolic tangent sigmoid activation function, and the second layer uses a linear activation function. This network was trained in MATLAB for 200 epochs with a training goal of 0.001 error.

The training set was also used to compute the measurement and process noise required for the Kalman filters. The measurement noise was obtained by taking the variance of the difference between the actual and estimated positions that were obtained. The process noise values for the Kalman filters were obtained by calculating the minimum distance error between the estimated and actual positions

for a range of process noise components X and Y. In other words, the process noise values that resulted in the smallest distance errors between the estimated and actual positions were selected as the process noise parameters.

5 Results and Discussions

5.1 Performance Comparison

Figures 5, 8, and 12 show the localization accuracy of the MLP, PV, and PVA methods respectively. As Table 1 indicates, the MLP neural network has the least distance error per estimate and hence the best localization performance. This is followed by the PV model of the Kalman Filter, and finally the PVA model of the Kalman Filter.

Method	Distance Error per Estimate	RMSE (X, Y)	Net RMSE
MLP	5.726	(5.905, 4.693)	7.543
PV	8.771	(7.276, 7.537)	10.476
PVA	9.613	(8.159, 7.937)	11.382

Table 1. Comparison of Localization Errors (cm)

The use of the Root Mean Square Error (10) also reveals that the MLP neural network has the best performance.

$$RMSE = \sqrt{\frac{\sum (Actual - Estimated)^2}{Number\ of\ Samples}} \tag{10}$$

$$Net\ RMSE = \sqrt{{X_{RMSE}}^2 + {Y_{RMSE}}^2} \tag{11}$$

One characteristics of the RMSE is that it is biased towards large errors since large errors squared result in larger values resulting in a greater contribution of error. This interesting characteristic will become apparent in the analysis that follows. Figures 7, 10, and 14 reveal the distribution of the magnitude of error during the entire testing data. It is interesting to note that the neural networks have a higher percentage of errors of less than 10 cm. Whereas, the Extended Kalman filters have a lower percentage of errors, but most of these errors are greater than 10.

It seems that the neural network tends to make a lot of little mistakes, whereas the Kalman filters make fewer mistakes, but when they do, their mistakes are large. This may be due to the process noise of the simulated motion of the object not adhering to the assumptions of Gaussian characteristics as discussed in Section 3. In addition, Figures 6, 9, and 13 reveal the location and magnitude of errors in the testing area as shown in Figure 4. It is interesting to note that a good portion of the error for all of the methods seems to be along the edge of the testing boundary in the vicinity of the beacons. This may be due to the fact that the use of ultrasound on the Cricket sensors results in large interferences between signals close to the beacons. It is also interesting to note that the Kalman filters display relatively large errors than the neural networks at the edges of the other boundaries as well. This may be due to the fact that the Kalman filters iteratively close in on the localized position. At the boundaries, where the object's motion takes a sudden turn, the Kalman filter's estimates require a few iterations before it can "catch up" with the object, resulting in larger errors.

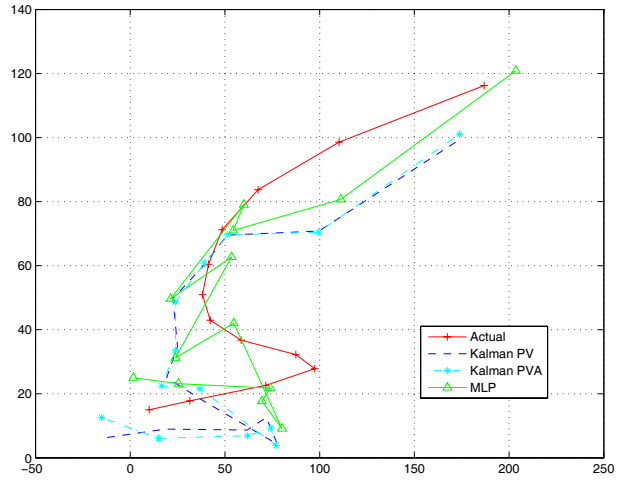

Figure 11. Simulated Path

In addition to the above analysis, the path of a moving object based on kinematics was simulated. Distances between beacons located at $(300, 0)$, $(300, 300)$, and $(0, 300)$ and a "moving" mobile node were computed. These distances were fed to each of the different localization methods and their performance was analyzed. Figure 11 depicts the estimated path of each of these localization methods. Table 2 reveals that the MLP neural network has the best performance, followed by the PV model Kalman Filter, and PVA model.

Method	Dist Error Per Estimate (cm)	RMSE	Net RMSE RMSE
MLP	16.568	(13.910, 11.594)	18.108
PV	23.359	(16.810, 17.059)	23.950
PVA	24.026	(17.349, 17.462)	24.615

Table 2. Error Per Estimate and RMSE for Simulated Path

5.2 Computation Requirement Comparison

Thus far, only the accuracy of the localization methods has been examined without any discussion of the computation requirements associated with them.

As mentioned before, these localization methods will be implemented on an embedded system with limited capa-

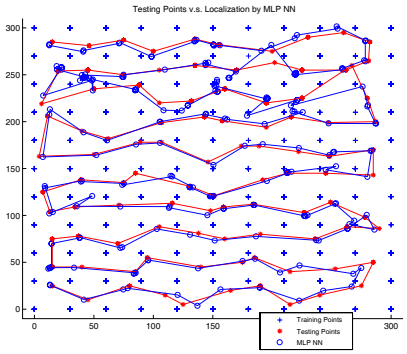

Figure 5. Tracking trajectory of MLP

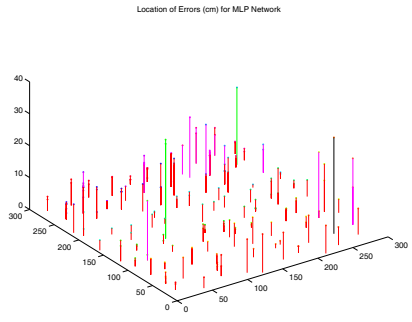

Figure 6. Localization errors of MLP

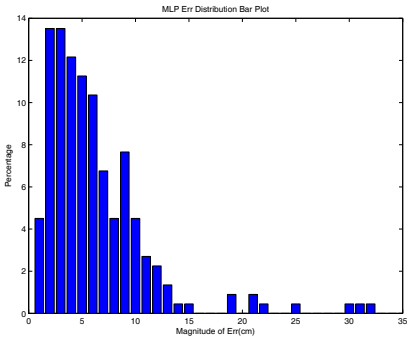

Figure 7. Error distribution of MLP

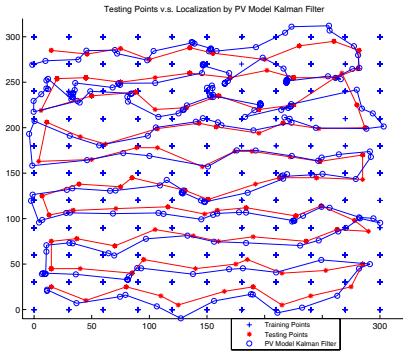

Figure 8. Tracking trajectory of PV

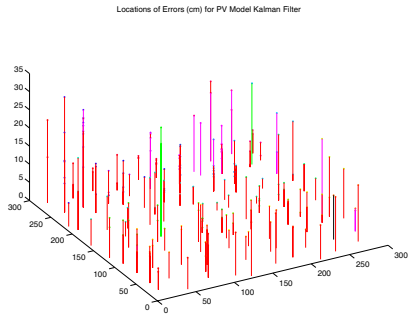

Figure 9. Localization errors of PV

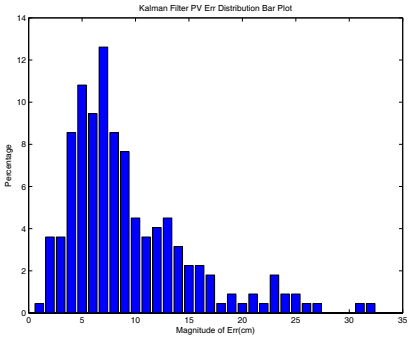

Figure 10. Error distribution of PV

bilities. Based on the application, an appropriate localization method must be used that balances accuracy with the capabilities of the system.

The following analysis utilizes the number of floating point operations as a metric to compare the different methods. For simplicity of presentation, this analysis assumes that the embedded system has native floating point capabilities and does not rely on integer computations to mimic floating point operations. Further, this analysis only accounts for steady-state computation, meaning the initialization and setup computations are not considered. Different localization applications may have different initialization and setup requirements. It is assumed that the neural networks are trained before-hand and the appropriate weights and biases are available. In the case of the Extended Kalman Filter, the listed operations are those that are associated with the set of iterative equations as described in Algorithm (1).

Method	Number of Floating Point Operations
MLP	153
PV	884
PVA	2220

Table 3. Comparison of Floating Point Operations of Localization Methods Per Estimate

As Table 3 reveals, the MLP is the least computationally intensive. It is followed by the PV and PVA Kalman Filters. These results as described in Table 3 provide an insight into the workings of these localization methods, however, it is very difficult to generalize these results.

A two layer MLP can be implemented using two sets of nested loops. The MLP has complexity $O(m^2)$ where m is the greatest number of nodes in the two layers. However, given the fact that the MLP in use in this experiment contains nine nodes in the first layer, and only two nodes in the second layer, $O(m^2)$ is an extreme overestimation of the complexity.

The Kalman filter equations as described in Algorithm (1) involve many matrix multiplications and an inverse operation for computing the Kalman gain K. These two operations have complexity $O(k^3)$ and as a result the Kalman filter is also of complexity $O(k^3)$ where k is the number of parameters in the state.

It is difficult to arrive at a generalized statement comparing the computational complexity of these methods. It is possible to compare the PV and PVA models. However, there are difficulties in trying to compare Kalman filters with the neural network. This is because there are no features that are shared between these two families of localizing methods. The Kalman filter utilizes a linear system to arrive at the localization estimates, whereas the neural networks localize by evaluating the inputs by tweaking it

Method	Order of Magnitude	Comment
MLP	O(m^2)	m is the greater number of nodes in the two layers.
PV	O(k^3)	k is the number of elements in the state variable.
PVA	O(k^3)	k is the number of elements in the state variable.

Table 4. Comparison of computational complexity between localization methods

using a set of activation functions and weights. Neural networks are a more amorphous method of modeling where, in the end, the arrival of the best network for the application is obtained through trial and error.

Another reason why it is difficult to arrive at a generalized statement comparing the Kalman filters and the neural networks is because, the scalability of the neural networks is not known. If the area of localization increased from the 300×300 cm grid as described above to a 400×400 cm grid, the noise characteristics of the distance measurements will also change. The ultrasound signals which are used to measure distances will attenuate differently over this larger distance. The noise characteristic of data for this new area of localization will be different and may require a much larger MLP than the one used above. Although unlikely, it is possible that this new neural network that is required to localize in this new or any other scenario may be more computationally expensive than the Kalman filters. The Kalman filter does not suffer from the same problem as the neural networks. If the size of the localization area changed, the computation complexity of the Kalman filter will not change. These are some of the difficulties in attempting to definitively compare neural networks with the Kalman filters.

5.3 Memory Requirement Comparison

Comparison of the memory requirements of these localization methods is as problematic as attempting to compare computational complexity. It is not clear at this time how the neural networks will scale compared to the Kalman filter for different applications. In the specific experiment carried out for this paper, the memory usage of the neural networks as compared to the Kalman filters is described in Table 5. It should be noted that the memory usage described here is the steady state memory, this does not take into account any initializations that may be required for different applications. It is also assumed that floats are four bytes long.

In the expressions for the two-layer neural networks, n is the number of nodes in the first layer, p is the number of nodes in the output layer, and m is the number of distance readings or the number of inputs.

In the expressions for the two Kalman filters, k is the number of elements in the state variable and m is the number of distance readings. The expressions for the memory requirements of the Kalman filters include an additional $k^2 + 2mk + m^2 + m + k$ bytes of memory for temporary variables.

It is interesting to note that the underlying memory characteristics for the Kalman filters are equivalent. The memory required by the MLP network is less than half of that required by the PV Kalman filter.

Method	Total Memory Usage	Number of Bytes
MLP	$nm + np + 2n + 2p + m$	280
PV	$6k^2 + 4k + 4km + 5m + m^2$	736
PVA	$6k^2 + 4k + 4km + 5m + m^2$	1344

Table 5. Comparison of memory requirements between Localization Methods

6 Conclusion

The experimental results indicate that among the three localization algorithms, the MLP neural network has the best performance in terms of accuracy, computation complexity, and memory usage. However, there are potential retraining or re-design costs associated with the use of neural networks that are not associated with the Kalman filter.

We have shown that the MLP neural network has a weaker self-adaptivity than the Kalman filters. First, neural networks perform well only for the area in which they have been trained. If the tracked object passes beyond the boundaries of the area where the neural network has been trained, the neural network will not be able to localize. Second, when the beacons from which the distance measurements have been used to train the network are moved, the MLP neural network needs to be re-trained and there is a possibility that the architecture of the neural network may need to change. On the other hand, the Kalman filters do not suffer from this problem and they can be used freely over any area once the appropriate noise parameters have been measured.

Compared with Kalman filter methods, the MLP neural network does have some advantages. The Kalman filters iteratively localize, correcting their estimates over time based on the noise parameters. The MLP neural network on the other hand localizes in a single pass of the network. The Kalman filter uses the laws of kinematics to predict the location of the tracked object. If the tracked object's motion is random and spontaneous the neural network's ability to localize in a single pass results in more accurate estimates every time. The Kalman filter, however, requires several iterations before it begins to reach the accuracy of the MLP.

In conclusion, where noise parameters are not expected to change, the localization method using MLP neural networks may be the best option. The high accuracy, minimal computational and memory requirements are highly desirable in embedded systems. If a flexible and easily modifiable method is required, then the Kalman filters