University of California
Los Angeles

Adaptation Using System Identification to Improve Electrolocation

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical Engineering

by

Newton Truong

2014

Abstract of the Thesis

Adaptation Using System Identification to Improve Electrolocation

by

Newton Truong

Master of Science in Electrical Engineering

University of California, Los Angeles, 2014

Professor Mani B. Srivastava, Chair

Electrolocation is a method of sensing found in weakly electric fish that utilizes electrical discharges to sense objects and navigate through their environment. Inspired by these biological findings, this thesis will describe a modeling and processing method to emulate an application using this sensing modality. We approached the problem with System Identification to estimate a non-parametric model instead of relying on complex physical equations. We will describe how a Kalman filter and an estimation function uses the model to process incoming sensory information. From our analysis, we show improvements using online adaptive model over offline training. Finally we will compare experimental results estimating object distance using linear regression and SVM.

The thesis of Newton Truong is approved.

William J. Kaiser

Gregory J. Pottie

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2014

To My Parents

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to personally thank Yasser Shoukry and Henry Herman for the tremendous help and guidance they have given me in the completion of this Masters thesis. I would also like to thank all of my mentors that had a role in shaping my education and the Networked and Embedded Systems Lab for giving me years of invaluable experience.

CHAPTER 1

Introduction

Certain species found in the African rivers, like the African Knifefish, have evolved special organs to discharge and sense weak electrical signals [Lis51]. There are multiple biological studies showing a tremendous range of capabilities and sophistication using this sensing modality. Gerhard von der Emde tested multiple species and demonstrated that they can distinguish objects with capacitive properties [Emd98]. In fact motor control is integrated with electric field sensing in that weakly electric fish exhibit a series of probing behaviors when approaching an object [TB79] [TM84]. They can eventually be trained to either approach or avoid objects [LM58]. Behavior of adaptation and learning have been demonstrated using this sensing modality and it's not simply limited to detecting objects. Electric fishes can use information from their electric organ discharges (EOD) to orient themselves in complete darkness, navigate through turbid waters of unstructured environments, and communicate with other weakly electric fishes [Emd99] [LCG13] [Kra96]. Much of the biological aspects have been studied in the last century between 1950s to early 2000s. It was not until recent decades that engineers are beginning to develop bio-inspired systems around underwater electric field sensing.

Almost every application using this sensing method by weakly electric fish have been replicated with artificial systems to some degree. We are able to mimic charge sensing capabilities with modern circuit technology, and similarly we can mimic detection and control behavior using mathematical models and signal processing

techniques. A common approach from a survey of three different research groups showed the use of Maxwell's equations as the basis for a model describing the the underwater electrical discharges [LCG13] [LJR13] [JKM07]. The resulting mathematical model is parametric and requires many physical parameters like electrical conductivity of the environment and geometric dimensions of the sensor and target object. Learning and/or identifying these physical parameters is a major task and requires complex experiments. Moreover, these parameters change with time, adding another dimension of complexity to these parametric models. These two main disadvantages of parametric models, namely 1) identifying parameters and 2) lack of adaptability, limits the usage of such models.

In this thesis, we are going to address some of these limitations by using non-parametric models, which paves the way to using numerical System Identification techniques. We are going to demonstrate our processing architecture using the test scenario of trying to estimate the distance of an object to our sensor.

CHAPTER 2

System Overview

The objective of this thesis is to explore the use of non-parametric models to design electrolocation systems. These models are used to estimate the location of an object based on the output data measured from the underwater sensor. In our analysis, we show that the flow of charges in the underwater environment can be modeled as a linear dynamic system of low-dimension. We then show that this low-dimensional model can adapt to changes in the environment. We will also show that offline training of the state space model does not provide a good estimate of the channel. The information garnered from adaptation is not only essential for model accuracy but also for the rest of the system in order to make good estimations. This chapter will introduce the three main processing blocks:

1. System Identification Block
2. Kalman Filter Block
3. Estimation Function Block

If the estimated output (calculated using the adaptive model) differs form the sensor measurements, we blame this difference to the existence of an object in the surroundings. We then utilize this difference in order to calculate the location of this object. This intuition lead to the system structure shown in Figure 2.1.

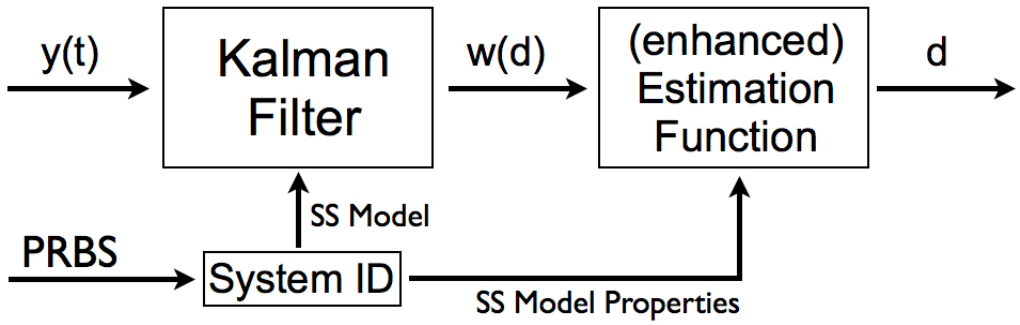

Figure 2.1: The top and bottom blocks can be separated into two processes that are not performed concurrently. The Kalman filter and estimation function blocks are used to detect objects nearby and estimate distance. Variable $y(t)$ is the observed signal out of the underwater channel. During the modeling phase, the channel is probed with a pseudo-random binary sequences (PRBS) that is used to perform System Identification. The state space model matrices are passed to the Kalman filter and information about system dynamics are passed as features to the estimation function.

2.1 System Identification

The first objective of our system is to build and adapt a non-parametric model. We do this through a process called System Identification. We first excite the channel by transmitting a frequency rich pseudo-random binary signal and simultaneously record the input and output data. We utilize this time series of input and outputs to identify a state space model of the form

$$x(t+1) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t)$$

where A, B, and C are matrices describing the dynamics of the system. Then with a convex solver, we numerically obtain the state space matrices for a series of different order complexities and estimation errors for further analysis. Typically one System ID step is not enough because model accuracy degrades over time. This is why the system has to adapt to the environment from time to time. New information about the system dynamics is passed to the remaining blocks.

2.2 Kalman Filter

The Kalman filter block is responsible for utilizing the information about the dynamics of the charges in the underwater environment in order to detect the existence of objects. Basically, it measures the deviation between the expected output (calculated using the model) and the actual sensor measurements. Hence, it is able to separate the interference of the object from the dynamics of the transmitted charges.

2.3 Estimation Function

As discussed before, the existence of an object within the proximity of the sensor results into some mismatch (disturbance) between expectations from the model and the measurements. This mismatch depends on how far the object is to the sensor and the governing relation between the mismatch over distance is non-linear. Therefore, the final step is to make an estimation of the distance based on the information and the statistics extracted from the Kalman filter and the features from the model. This thesis will test the estimation accuracy through the use of linear regression and support vector machines. The machine learning algorithms will be used to train a linear function and classifier. We then compare the accuracy of our estimations against the collected ground truth.

CHAPTER 3

System Identification

In this chapter, we will give details of how to use System Identification algorithms and techniques to build a linear state space model describing the dynamics of the electric field in the underwater environment. The last section shows how one can modify the state space equation with additional variables for the purpose of tracking disturbances that deviate from the model because of the existence of an object.

3.1 Describing System Identification

By definition *Identification* means using only the input and output measurements to model a dynamic system [LZ06]. A physical parametric model may present a more complete view of how the system interacts but may be difficult to implement in an environment where control or detection is needed. This especially true when the hardware resources are severely limited like autonomous vehicles. What Identification techniques provide is a way to estimate a non-parametric model that gives us a relationship between the input and output variations so that we can use the information in other processing blocks. All the physical interactions are numerically encapsulated in a few coefficients that can easily be updated if necessary.

The System Identification is done in these four generic steps [LZ06]:

1. Input/Output data acquisition
2. Selection of model complexity
3. Estimation of model parameters
4. Validation of model

There are many types of methods and techniques for accomplishing each step. Different ways have been developed suited to what needs to be identified and how complex to make the model. To acquire good input/output data, we will follow the channel excitation method described in Landau's book *Digital Control Systems* [LZ06]. The model equation we will use to approximate the channel is one that is commonly known in modeling linear systems. The subspace algorithm for the identification process will be the N4SID algorithm available in MATLAB as part of the System Identification toolbox. The complexity of the model was determined through analysis with the other filter blocks to keep a low order system but still achieve adequate accuracy. The details of this implementation are given in the subsequent sections.

3.2 Linear State Space Model

In order to follow the system processes and make determinations about normality, we first have to approximate it as a linear equation. Most linear systems can be adequately described by the two linear state space equations 3.1.

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.1}$$

The parameter x is a vector that describes the state of the system at time t. Vector x is not explicitly observed and must be calculated over time. The matrices A, B, and C are determined using the System ID solver. The size of the matrices are proportional to the order of the system and large matrices are a disadvantage because matrix operations do not scale linearly. Therefore it is important to regulate the model complexity. The vector u is the known input into the channel and y is the model output.

3.3 Solving State Space Model

We need to probe the channel such that we converge on a numerical solution that has the highest degree of accuracy. Landau suggests that one standard way to go about this is to inject a frequency rich input to excite the channel. One such input is the pseudo-random binary sequences (PRBS), which approximates discrete-time white noise [LZ06]. Landau also notes that one specific requirement of the PRBS must be that the maximum duration of one of the pulses must be longer than the rise time of the signal through the channel. The PRBS used in our experiment is shown in Figure 3.1

The discharge of the PRBS from the transmitter and the sampling from the receivers takes place simultaneously. It is controlled through MATLAB's hardware-in-the-loop xPC Target system with National Instrument hardware (ADC/DAC), refer to Figure 6.4. The data is saved and analyzed in MATLAB. The inputs and outputs are fed into a convex solver, the N4SID algorithm, to find A, B, and C of the state space model.

The N4SID algorithm is always convergent and is numerically stable. The algorithm tries to find the projection of input and output data through QR factorization and singular value decomposition [OM94]. It is also an advantage in terms of processing because the algorithm is non-iterative with all optimization

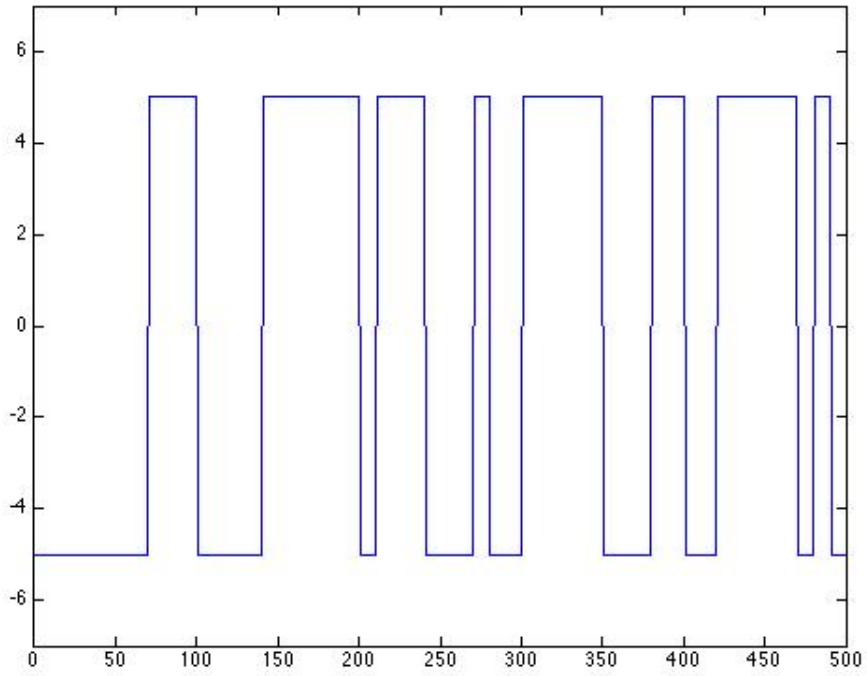

Figure 3.1: Pseudo-Random Binary Sequence used to excite the channel for System Identification.

parts being linear computations [OM94]. The only parameter that is required besides the data is the order of the system, obtained by examining the singular values to determine the set with the highest contributions. Fortunately MATLAB provides a feature that determines the best overall order, which turns out for our case to be a third order system. This can be confirmed if you examine the singular values of the A matrix with the top two values significantly higher than the third. This means that the third singular value has a lesser contribution than the first two and all other subsequent singular values even less.

To summarize, the input into the model will be the pseudo-random binary signal and the output will be the estimation of the signal through the channel. Figure 3.2 has two plots showing the measured ground truth and the estimation from the model. This figure only shows the estimation for one of the electrodes

Electrode	Mean Error (V)	Std (V)
Electrode 1	0.0756	0.0507
Electrode 2	0.0771	0.0522
Electrode 3	0.0760	0.0482

Table 3.1: The table shows the mean error and standard deviation between the sampled output and model estimate. The calculations are shown for three different electrodes to show that the distribution of errors are similar across all of them.

but Table 3.1 shows the mean error and standard deviation between the measured and estimate signals for three electrodes. The errors are much lower than the magnitude of the output signal, which is about 4 V_{pp}. These are three electrodes that are adjacent to each other on the sensor array. The section on Sensor Fusion explains why our estimation is done three electrodes at a time.

3.4 Sensor Fusion

We can choose to have one model for each individual electrode or we can have a single model that tries to estimate the output from a group of electrodes. In this thesis, we did the latter using three adjacent electrodes to detect targets. The intuition is that there are correlations between the disturbances in the adjacent electrodes, although smaller in magnitude, with the strongest disturbance evoked on the center electrode; the center electrode is the one closest to the object. Topologically one electrode is in the center and two are on the side at 45 degree angles relative to the center, refer to Figure 6.3 in the Biomimetic Hardware chapter. Using a group of three electrodes, we hope to maximize the information we can sense to improve the model estimation accuracy.

3.5 Modifying the Model to Track Disturbances

The typical use of the Kalman filter is for correction of an otherwise corrupted measurements due to noise or system dynamics. The model currently described

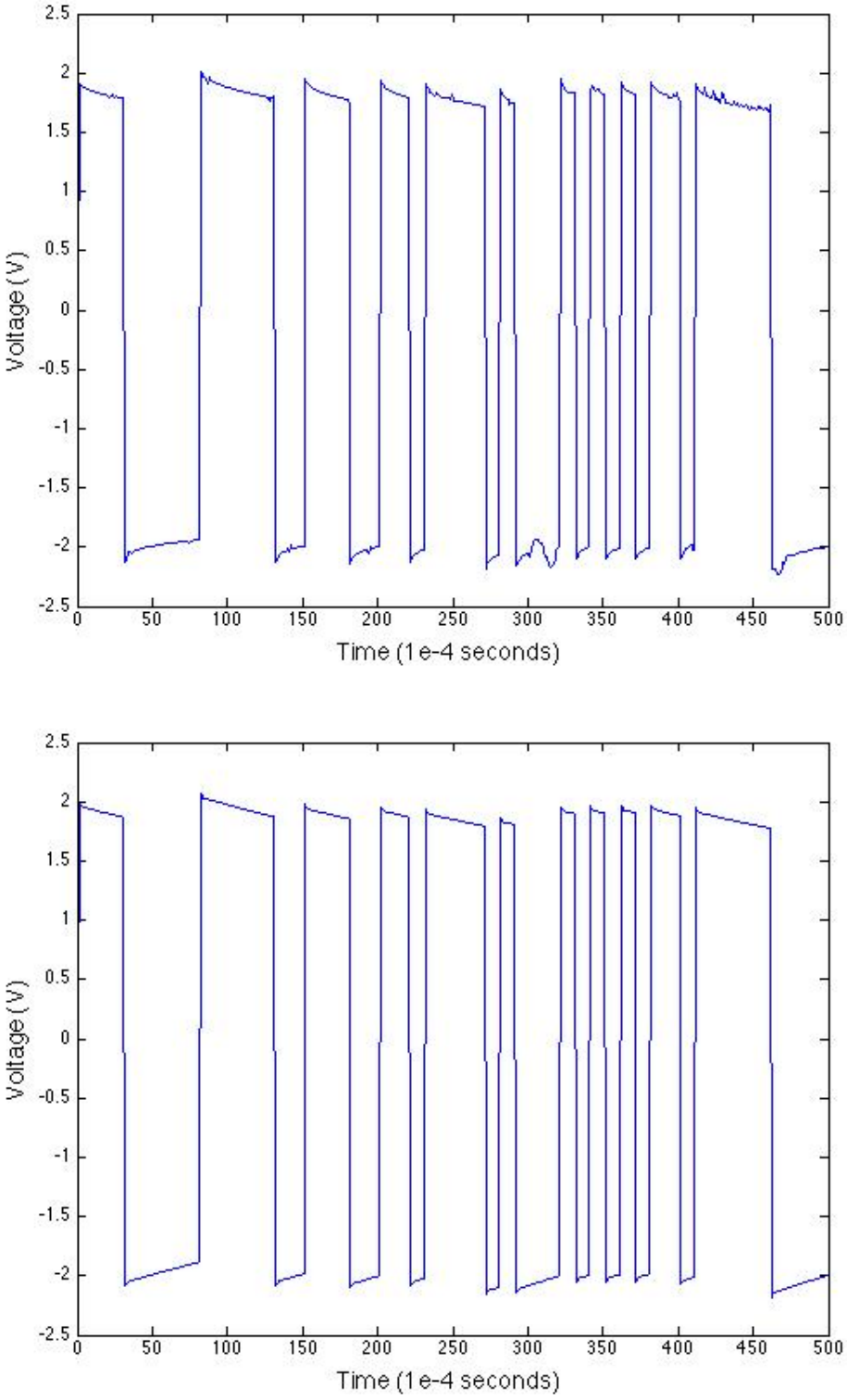

Figure 3.2: The top figure is observed output from the channel and the bottom figure is the estimate from the state space model using the PRBS input.

was estimated based on an object-free environment. Hence, the output estimated by the Kalman filter will try to match the output measured in cases with no object present in the environment. What we have to do is modify the original model with a variable that can account for any type of changes in the channel evoked by objects. When the Kalman filter does a measurement update, it will use the additional variable along with the state vector x to reduce the error variance between the estimated signal and sampled output. The modified equation is shown in 3.2 and the additional variable is vector $w(d)$, which has same dimension as the number of states in model. Vector $w(d)$ captures the effect on the charges to the presence of an object in the environment.

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + w(d) \\ y(t) &= Cx(t) \end{aligned} \tag{3.2}$$

Equation 3.2 must be rewritten so that the structure of the state space equation structure matches the original form. This is so that we can integrate the model into our Kalman filter. The derivation is shown below in equations 3.3.

$$\begin{aligned} \begin{bmatrix} x(t+1) \\ w(d) \end{bmatrix} &= \begin{bmatrix} A & I \\ 0 & I \end{bmatrix} \begin{bmatrix} x(t) \\ w(d) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) \\ \begin{bmatrix} y(t) \\ w(d) \end{bmatrix} &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ w(d) \end{bmatrix} \end{aligned} \tag{3.3}$$

In the state space equation 3.3, the new state vector x now depends on the current state and the additional vector $w(d)$. Matrix I corresponds to identity matrices. To reduce the clutter of the equations, we will from now on refer to equations 3.4 for the rest of the thesis.

$$\begin{aligned} \begin{bmatrix} x(t+1) \\ w(d) \end{bmatrix} &= \bar{A} \begin{bmatrix} x(t) \\ w(d) \end{bmatrix} + \bar{B}u(t) \\ \begin{bmatrix} y(t) \\ w(d) \end{bmatrix} &= \bar{C} \begin{bmatrix} x(t) \\ w(d) \end{bmatrix} \end{aligned} \tag{3.4}$$

The vector $w(d)$ will act as an additional parameter for the Kalman filter to update, should the estimate from the original model differ from the taken measurements. If we have the original channel with no objects in it, then $w(d)$ will simply be an unused zero vector. Going forward, we want to invert the information contained in the subspace vector $w(d)$ into a meaningful value such as distance, but in order to do that we need another function to do the conversion. We don't know the exact parametric equations to do this so we are going to numerically estimate a function using machine learning.

CHAPTER 4

Kalman Filter

In this chapter we will talk about about the Kalman filter and how we can use it to estimate object disturbances in the environment. We will talk specifically about the time varying Kalman filter and another section will outline how we normalize and linearize the output values from the filter.

4.1 Using the Kalman filter

Originally developed for navigating through outer space, the Kalman filter is mainly used to estimate system states that are otherwise difficult to observe [Sim01]. Typically the Kalman filter will correct whatever undesirable noise or fluctuations occur in the measurement output. However for our purpose, we want the part of the measurement output that is corrupted or otherwise deviated from the expected model. This is why we modified the original state space equation in the previous chapter. Another attractive feature of the Kalman filter is that it is implementable on a microcontroller and even possible to run in real time [Sim06].

4.2 Time Varying Kalman Filter

The entire time varying Kalman filter was implemented in MATLAB. The algorithm is fairly short and the pseudo-code is shown below. First we begin with certain intializations:

$Q = 1;$

$R = I;$

$P = \overline{B}Q\overline{B}^T$

$P(4,4) = 1;\ P(5,5) = 1;\ P(6,6) = 1;$

$x_0 = 0;$

$for\ each\ sample:$

$M_k = P_k\overline{C}^T(\overline{C}P_k\overline{C}^T + R)^{-1}$

$x_k = x_k + M_k(y_k - \overline{C}x_k)$

$P_k = (I - M\overline{C})P_k$

$x_{k+1} = \overline{A}x_k + \overline{B}u_k$

$P_{k+1} = \overline{A}P_k\overline{A}^T + \overline{B}\overline{B}^T$

end loop

The matrices $\overline{A}$, $\overline{B}$, and $\overline{C}$ are from the estimated state space model that have been modified, see chapter on System Identification. Matrix M_k is the gain factor that is calculated before the measurement update equation $x_k = x_k + M_k(y_k - \overline{C}x_k)$. The measurement update equation for vector x takes into account the new measured output y at sample k. The time update equation $x_{k+1} = \overline{A}x_k + \overline{B}u_k$ makes an estimation x_{k+1} based on the current state x_k. The measurement and time update equation for vector x are calculated with each cycle of the loop and modify

elements of $w(d)$ in vector x. Enough samples are taken at each probed distance so that the Kalman filter output stabilizes, primarily the elements of $w(d)$. The stabilized outputs are then normalized and linearized for the estimation function.

4.3 Implementing Model Adaptation to Kalman Filter

The Kalman filter is not involved in the System Identification step. However the criteria to initiate a model update is very much dependent on the results of the Kalman filter. Obviously we want to update the model when there is just the channel and no other objects near the sensor. As the model becomes less accurate over time, our ability to make estimates of distance becomes coarser. Luckily we only need to know in a binary sense whether or not there are objects within a detectable distance in order to initiate a new round of System Identification. These decisions can be made based on large changes in the signal, which we assume has nothing to do with natural environmental dynamics. At a certain point when we determine it is safe (i.e. no objects present), we can perform our model update steps. The new information that must be passed to the Kalman filter are the new system, input, and output matrices $\overline{A}$, $\overline{B}$, and $\overline{C}$.

4.4 Setting up Kalman Filter for Real Time Filtering

Referring back to the Kalman filter code, there are some complicated computations such as the matrix inversion, which require computations on the order n^3, where n is the order of the matrix [Sim01]. However there is a clever way to implement the filter so that not all the computations are done all at once. Certain parts of the Kalman filter can be calculated ahead of time, primarily the equations that do not require the new output measurement y_k [Sim06]. This means the P covariance matrix can be pre-computed and stored after every model update.

Ideally it would be right after System Identification has taken place, which is not a critical moment. This shifts the burden away from the microcontroller and onto the memory unit. How much memory is required depends on the sampling rate and the time it takes for the Kalman filter to stabilize. This trade off allows us to use the time varying Kalman filter effectively in a real time fashion.

4.5 Normalize Kalman Filter Outputs

The Kalman filter outputs $w(d)$ are still a time series and not yet suitable as features for a machine learning algorithm. In this section we are going to go through the steps of how we normalize the data. To illustrate these steps clearly, we are going to use one of the experimental data sets collected from the system using the algorithms described up to this point. The particular data set was chosen due to the cleanliness of the sampled outputs; in reality, the experimental data taken are not normally as clean.

At each distance from the target, we transmit a square wave signal for a time duration much longer than the period of each wave. The reason is to allow the Kalman filter output $w(d)$ to produce stabilized values. Figure 4.1 shows the time series data from Kalman filter after it has stabilized. The figure is showing output data when the sensor is 0.5 inches from an aluminum target. The purpose of normalization is to have a quantitative measure for each cycle of the square wave. The reason is that each normalized values is an indicator of the amount of changes to the original signal evoked by the object nearby. The larger the normalized value the closer the sensor is to the object. For each set of samples that represent a single square wave cycle, we calculate the root sum square or the Euclidean distance. There are multiple stable square wave cycles in a single distance measurement, so we do an additional averaging of all the stable normalized values.

In Figure 4.2, we show the results from normalizing the data at different dis-

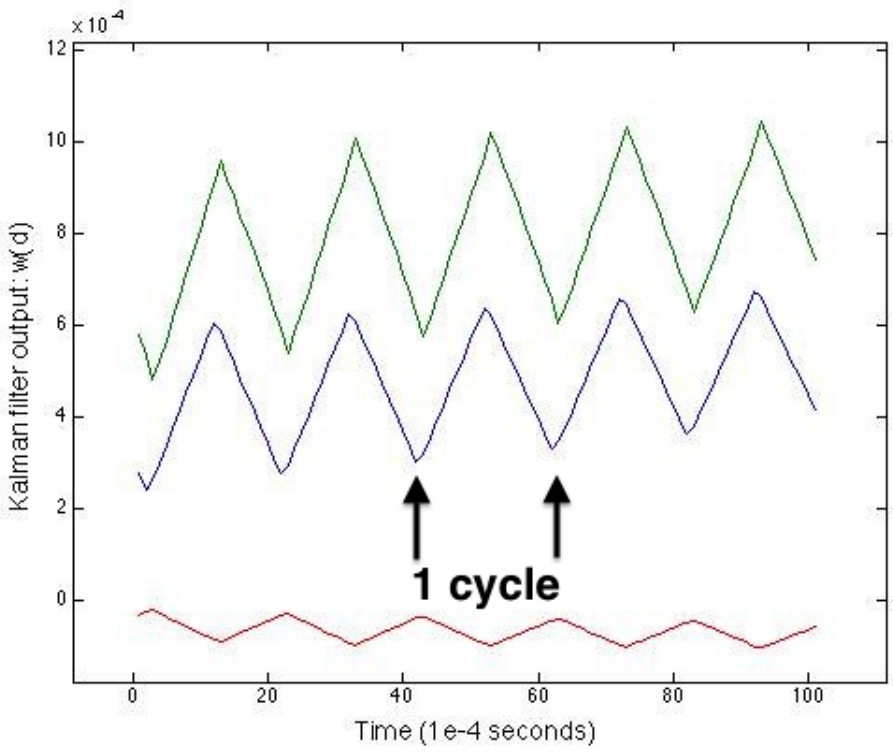

Figure 4.1: Each color represent an element in vector $w(d)$: $w_1(d)$ Blue, $w_2(d)$ Green, $w_3(d)$ Red. The figure shows the Kalman filter output of vector $w(d)$ over time. The example data is an aluminum target being probed at 0.5 inches away from the sensor. The figure is only showing the stabilized output from the filter.

tances for a single experiment. As you can see when it comes to disturbances evoked by an external objects within the channel, Figure 4.2 shows the non-linear effects quite clearly. In order to proceed further in our analysis, we need to linearize the data so that it maximizes the effectiveness of our estimations. Not to mention that trying to deal with non-linear data directly is computationally complex and will be a burden on the processor.

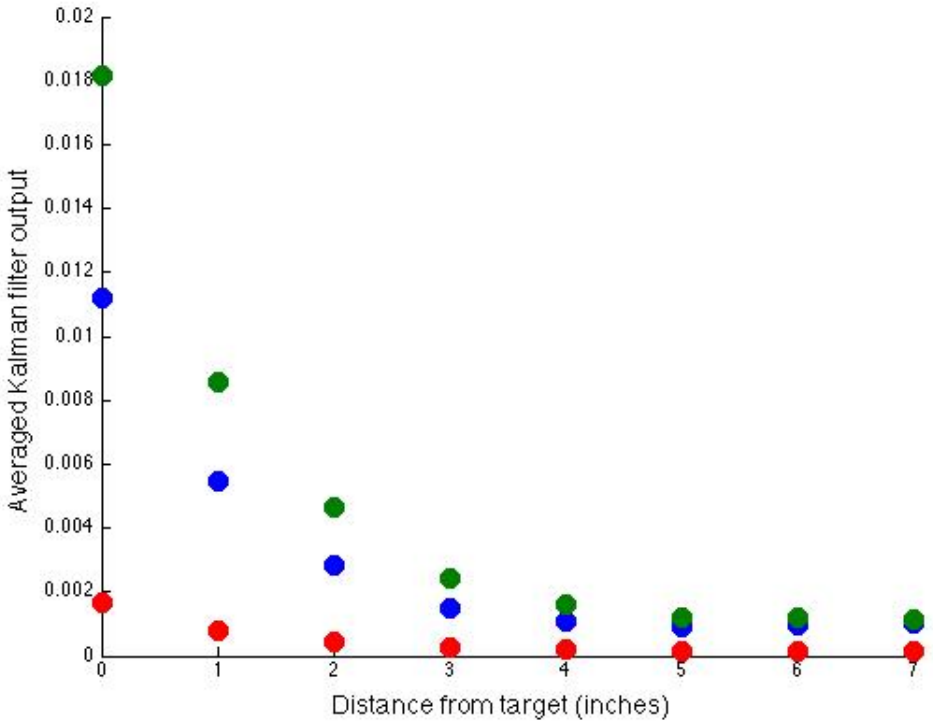

Figure 4.2: Each color represent an element in vector $w(d)$: $w_1(d)$ Blue, $w_2(d)$ Green, $w_3(d)$ Red. The figure shows the processed Kalman filter output normalized into a single value at each probed distance. The data shown in this figure comprises of a single experimental run where the probe discretely approaches an aluminum target.

4.6 Linearized Kalman Filter Outputs

There are many ways you can try to linearize the data from using high order polynomials to exponential functions. Various equations were tested through trial and error for effectiveness but the best one is derived from fundamental principles of Maxwell's equation. We are going to simplify the equation by just examining the order of magnitude drop off in field strength over distance, which is the primary relationship we want linearize. In Henry Herman's Masters thesis, he derives the field strength relationship over distance as equation 4.1 [Her13]. What we simply do is take that relationship and invert the equation to get equation 4.2.

$$E \propto \frac{1}{r^3} \tag{4.1}$$

$$\overline{d} = \sqrt[3]{\frac{1}{w(d)}} \tag{4.2}$$

As you can see in Figure 4.3, the transform using equation 4.2 gives fairly good linearized data. The leveling of sensor data at farther distances from the object shows that we are reaching our detection limitations.

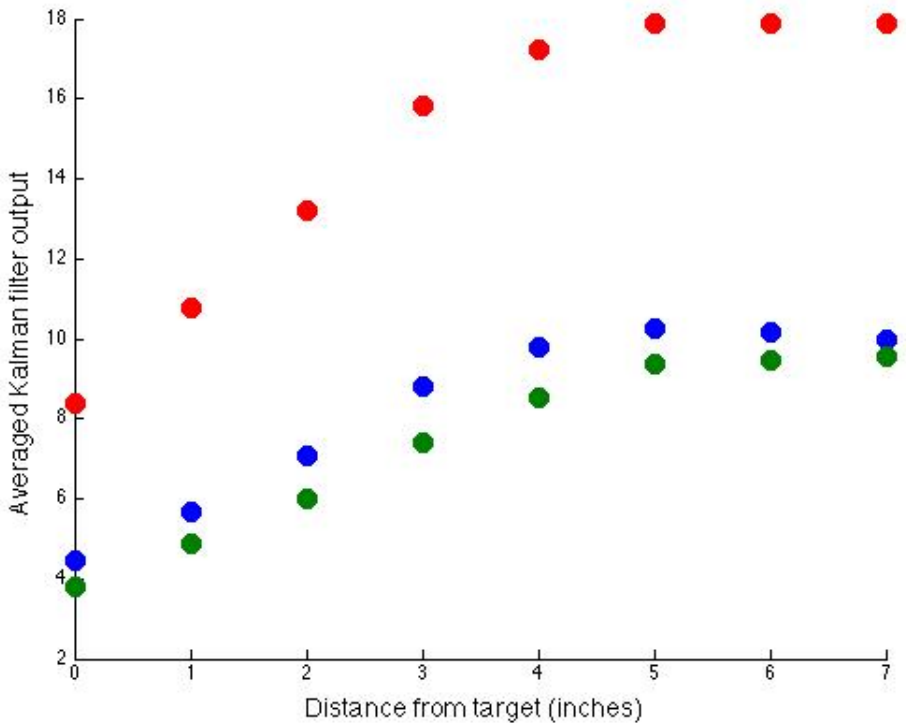

Figure 4.3: Each color represent an element in vector $w(d)$: $w_1(d)$ Blue, $w_2(d)$ Green, $w_3(d)$ Red. This is the same data shown in Figure 4.2 but after it has been linearized by equation 4.2.

CHAPTER 5

Estimation Function

In this chapter we will go over two machine learning algorithms implemented to estimate the location of an object given the sensor measurements. We are also going to go over the set of features selected to help make the distance estimation. Lastly we will briefly cover some book keeping such as cleaning up data sets and a randomization of training data.

5.1 Linear Regression

A linear equation was chosen as the estimation function because the transformed data showed good monotone linear characteristics. The output data is inherently continuous and having a linear function will allow the system to make interpolations from the measured output during the estimation process. The linear regression algorithm in MATLAB first uses rank revealing QR factorization to find the independent columns and then proceeds to compute the Least Square coefficients. The training of the estimation function is done offline. Once we have the model coefficients b and c, the estimation equation is simple to compute as shown in 5.1.

$$\hat{d} = Xb + c; \tag{5.1}$$

$$X_i = [feature_1,\ feature_2,\ ...]$$

where matrix X is a column vector comprising of the features needed to compute

a distance estimate $\hat{d}$. In order to solve for b and c, first we collect a large set of training data. We label the data and record the ground truth distance. The two primary steps in MATLAB's implementation of the regression algorithm are shown in equation 5.2.

$$\begin{aligned} \begin{bmatrix} 1 & X \end{bmatrix} &= QR \\ \begin{bmatrix} c & b \end{bmatrix} &= R(Q^T y)^{-1} \end{aligned} \tag{5.2}$$

Matrix X in equation 5.2 is the training data and vector y is the corresponding labeled distances. We are solving for coefficients b and c simultaneously by embedding a column of ones in X. What is not shown is if X has dependent vectors, then vector b is limited to the rank of the matrix.

5.2 Support Vector Machine

Support Vector Machines was chosen to test the effectiveness of classifying the data through a different set of criteria. Instead of trying to fit a linear function to the data, we are going to find the margins that best separate the different classes of data. In this case, the classes are the discrete distance measurement from 0 to 7 inches at 1 inch increments. We have multiple class labels, however SVM is built for binary classification. In order to work around this issue, we have to employ a trick where we perform multi-class classification through a one versus all grouping. Each discrete distance stated above will have its own set of support vectors. Test data, separate from training data, will be attempted on each set of classifiers and a positive hit will indicate a correlation between the test data and training data at that specific distance. The optimization problem for a single classifier is shown in equation 5.3 [HCL10].

$$\begin{aligned} \min_{w}: \quad & \frac{1}{2} w^T w \\ \text{subject to:} \quad & y_j[w^T \phi(x_j) + b] \geq 1 \end{aligned} \tag{5.3}$$

We are trying to solve for a minimum set of weights w. The vector y_j are labels for classification, either 1 or 0, corresponding to each vector x_j feature data set. The function ϕ is a transformation function to convert data that are not linearly separable. In our case we are going to use a Gaussian radial basis function, which is best suited for our data set. The data clusters are based on the distance to the object, so the best criteria to classify them will be the distribution from the central mass of those clusters. It would be difficult to use a linear or even a polynomial function to perform the marginal separation due to the multiple scattered clusters in the data set. Typically it is better to solve the dual form of the optimization problem 5.3, which means setting up a Lagrange equation.

$$L = \frac{1}{2} w^T w - \sum_j \alpha_j (y_j [w^T \phi(x_j) - b] - 1) \tag{5.4}$$

$$\text{by setting } \frac{dL_p}{dw} = 0 \text{ then } w = \sum_i \alpha_i y_i \phi(x_i) \tag{5.5}$$

substitute (5.5) into (5.4) for

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) + \sum_i \alpha_i y_i b \tag{5.6}$$

We can now use the Gaussian radial basis kernel to substitute $\phi(x_i) \cdot \phi(x_j)$ to give us equation 5.7.

$$\begin{aligned} L &= \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ K(x_i, x_j) &= \exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2}) \end{aligned} \tag{5.7}$$

Then we optimize equation 5.7 subject to maximizing $a_i \geq 0$ and minimizing b

with the constraint $\sum_i \alpha_i y_i = 0$ using a quadratic programming algorithm. To obtain the vector weights w which define the hyperplane of each set of support vector, we use this previously derived equation: $w = \sum_j \alpha_j y_j \phi(x_j)$ and b is found by inverting the equation with the data points that satisfy $y_i(w \cdot x_i - b) = 1$. Finally the classification computation is defined in 5.8.

$$classify(x) = \text{sign } (\sum_j \alpha_j y_j K(x_j, x) - b) \tag{5.8}$$

Before the SVM algorithm is processed and as part of good practice, the data will be zero mean shifted and scaled to be between [-1,1]. This is to prevent large numeric values from dominating the classifier because the SVM kernels must perform an inner product calculation with the feature vectors [HCL10].

5.3 Feature Selection

The parameters that we can choose as features are limited to what we know about the latest updated model and the output from the Kalman filter. We have to be careful not to include information that the system cannot practically extract from the environment through sampling.

Through experimentation and multiple iterations, the following set of parameters were found to train the most accurate model. The features from the state space model include the norm, which is the the root mean square of the impulse response of the dynamic system, and the dampening frequency of largest real eigenvalue of the system A matrix. Note that these feature calculations are done on the original state space models without the additional vector $w(d)$. The features from Kalman filter output include vector $w(d)$ and the variance of its elements. The features from $w(d)$ and subsequent statistics are extracted after normalization and linearization.

There are a total of six feature parameters used for each estimate, as listed below. However not all of the parameters are calculated in real time. The parameters from the model are only computed when a new model is estimated through online System ID.

- w(d) = $[w_1(d), w_2(d), w_3(d)]$ output from Kalman filter
- var(w(d)) variance of kalman filter output
- norm of state space model
- damping frequency of real eigenvalue of state space model

5.4 Cleaning and Folding Data Sets

The data collection process is not by any means clean. There were experiments where the data set did not follow the expected output. The physical interaction between the channel and the charges that are causing this behavior is beyond the scope of this thesis. These data sets that did not show good linearity were removed to improve the accuracy of the trained estimation functions. Figure 5.1 shows one of these poor data sets after normalization and linearization. Compared to Figure 4.3, Figure 5.1 is not monotonically increasing and has a clear reduction in sensor sensitivity. The signal strength at distances near the target object are similar to ones farther away. The data sets were manually examined and excluded from the training process.

Even within the curated data, there were some subsets that proved to be better training data than others. This why we use a technique where multiple subsets of training data were picked at random to train a model and tested with the data that was withheld. This helps avoid unknown correlations that might otherwise overfit the model.

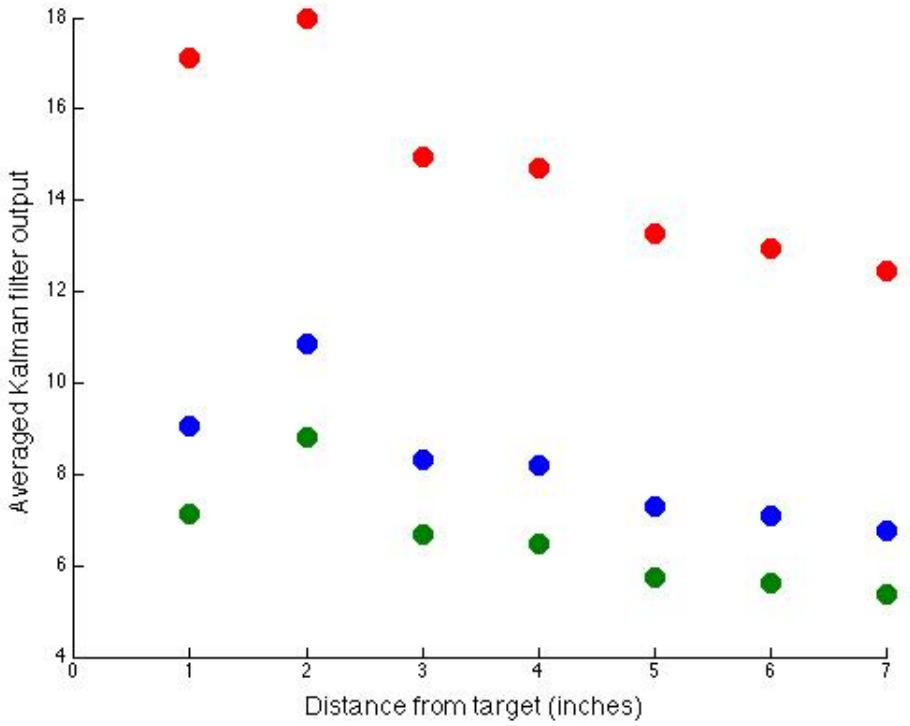

Figure 5.1: Each color represent an element in vector $w(d)$: $w_1(d)$ Blue, $w_2(d)$ Green, $w_3(d)$ Red. The figure shows an example of an experimental run where poor data sets were collected. The expected trend after normalization and linearization should have values increasing with distance.

CHAPTER 6

Biomimetic Hardware

This chapter will outline the biomimetic hardware implementation starting from the basic unit of the sensing organ. We will compare the functionality between the biological and the engineered counterparts. The last section will specify the types of signals we use for modeling and testing. For greater details, refer to Henry Herman's Masters thesis, whom built the original hardware for this research project. [Her13]

6.1 Analog Front End and Sensor Array

The biological system in weakly electric fish that we are trying to emulate has an organelle that can sink charges. The analog circuitry that best models this behavior is a current sensing op-amp, in Figure 6.1.

These electroreceptor organs in weakly electric fishes are point contacts on the skin that are connected to the brain through the peripheral nervous system. The main cell body is not directly in contact with the outside environment but instead have short canals that guide the charges to the sensory cell membrane [Kra96]. The biological process of sensing is a chemical one, with each incoming charge causing a secretion of neural transmitters. The circuitry used to mimic this process will be a sense resistor, acting as our "membrane." The op-amp will be the main cell body that amplifies and relays a stronger signal to be processed. Multiple circuit units were assembled to form a sensing array, in Figure 6.2.

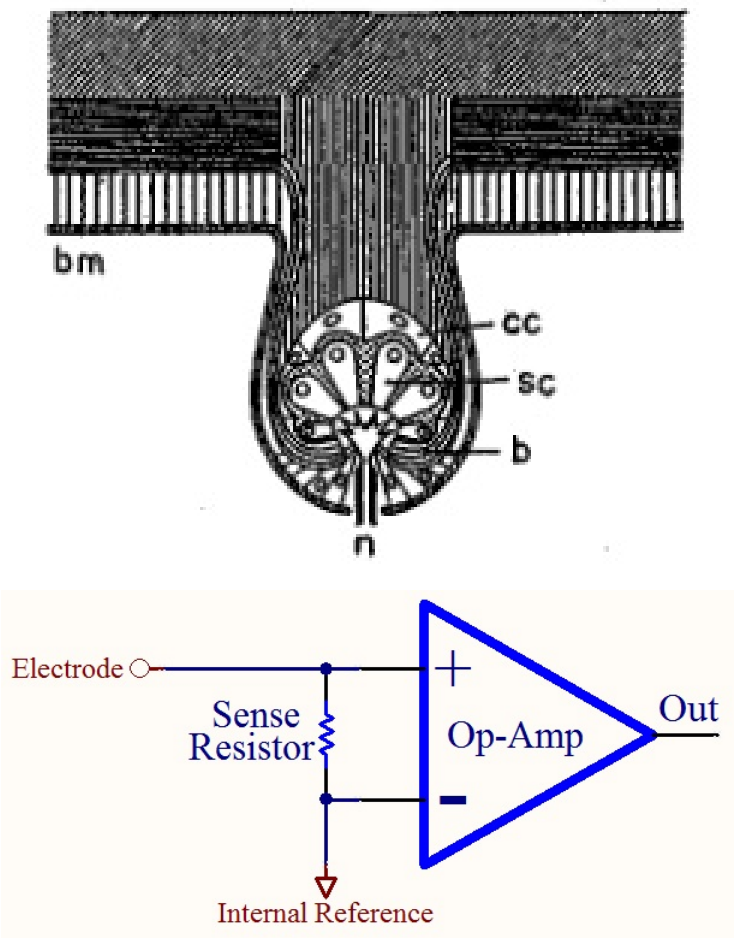

Figure 6.1: The biological electrorecetor organ (top) and functionally similar circuit configuration (bottom) [Kra96].

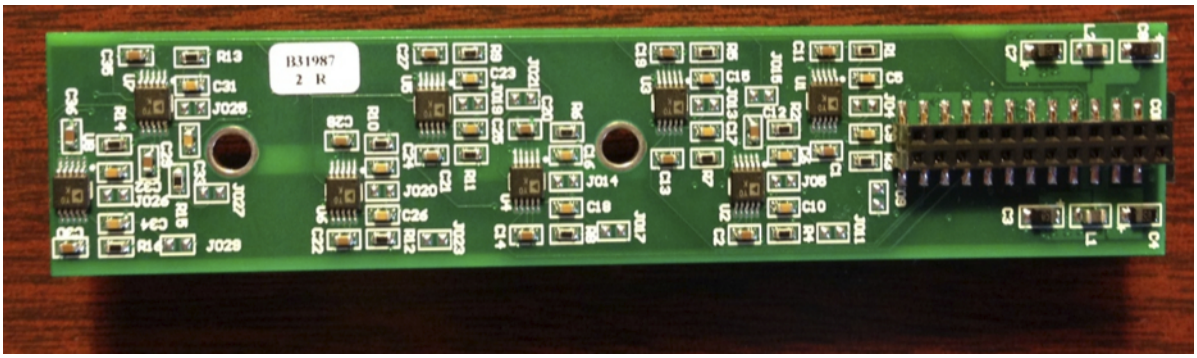

Figure 6.2: PCB with multiple analog front end units [Her13].

6.2 Sensor Body

The sensor body itself is modeled after the shape of a fish. Geometrically it is long and tubular with an insulating skin made of neoprene. Weakly electric fishes have their electroreceptors spread out throughout their bodies but we differ in that all our receptors are placed together in a circular array at the opposite end from the transmitter.

Figure 6.3 shows how the fish and our engineered system establishes the potential field around the body. When we instantiate a discharge, the field is established from the bottom to the top. The charges will go through the underwater channel and into the low impedance electrodes. Each electrode is connected to its own op-amp unit on the PCB in Figure 6.2. Lastly to perform our analysis in MATLAB, we will have to digitize the signal.

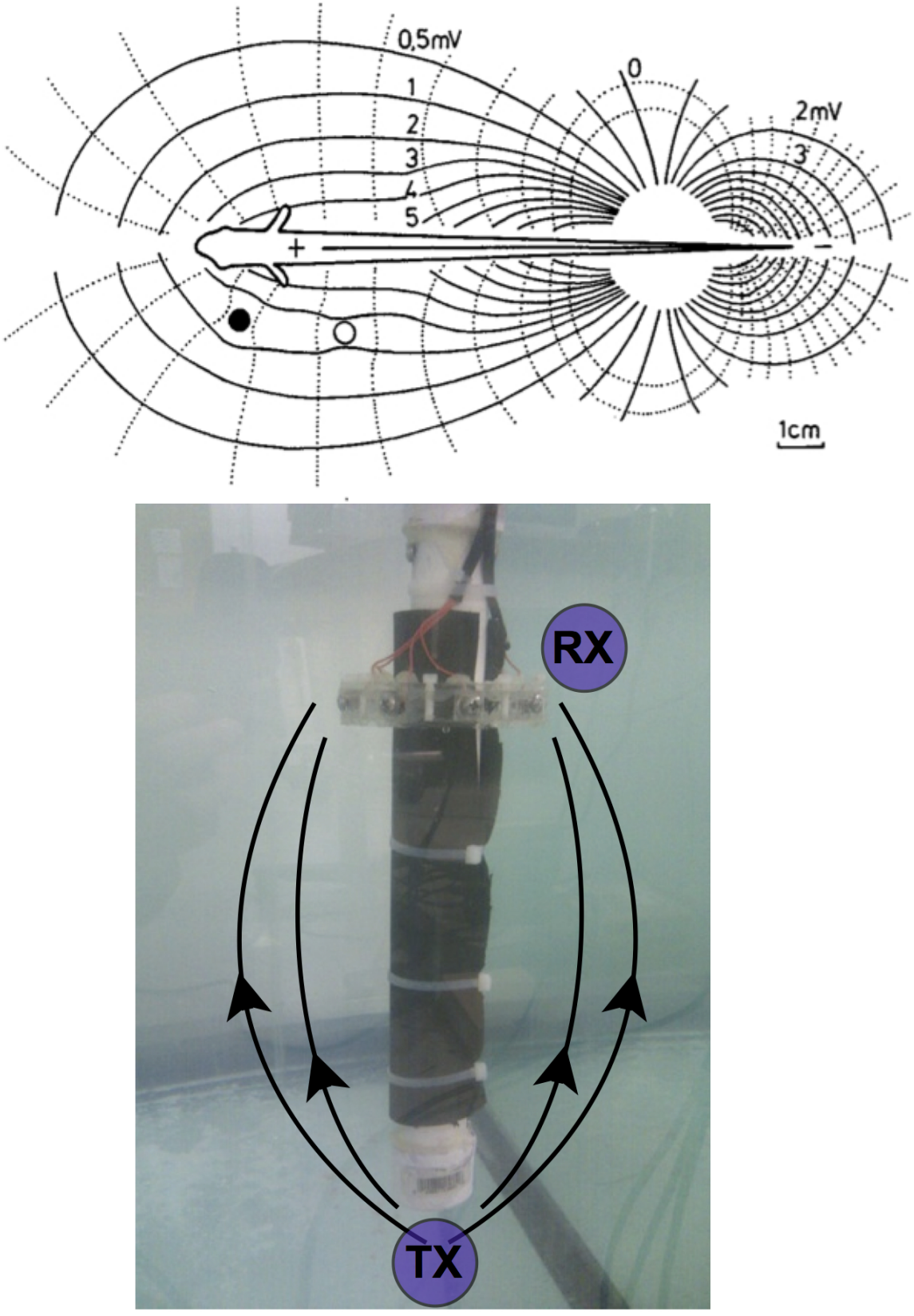

Figure 6.3: The electric field established by weakly electric fish and sensor body [Kra96] [Her13].

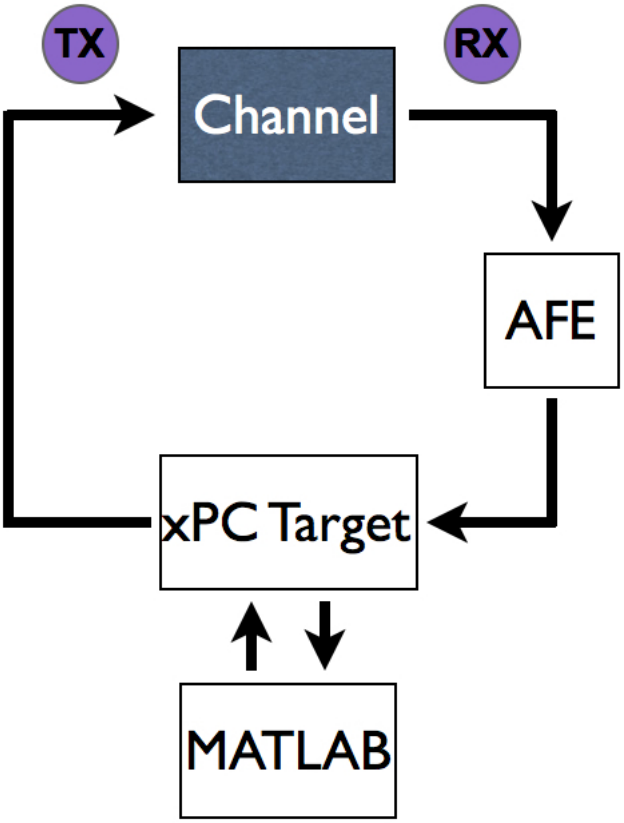

Figure 6.4: The xPC target function as both the signal generator and analog signal digitizer. The signal travels through the underwater channel to the analog front end (AFE) where it gets amplified before being sampled by the xPC target. The data is then analyzed through different filtering blocks in MATLAB.

6.3 Digitizer

The simultaneous electrical discharge and sample is done by a National Instrument and controlled using MATLAB's xPC Target system. This method of synchronous discharge and sampling is not unlike the biological functions of the weakly electric fishes. The tuberous electroreceptors have the ability to both mark units of time with high sensitivity and detect small changes in amplitude from the fish's own electrical organ discharges [Kra96] Figure 6.4 shows the high level overview of the major blocks talked about in this chapter.

CHAPTER 7

Experimental Setup

I will briefly go over the mechanisms needed to carry out the experiment of our test scenario, how they were designed, and the technical specifications. It will fill in any details not explicitly stated in the previous chapters and provide necessary context for the results.

7.1 Test Application

There are many applications that this sensing modality can be tested for. We know that nature already uses it for tracking prey, localizing prey, classifying objects, and communicating with other fishes [Kra96]. However this thesis will only cover one application which is estimating the distance of the sensor to the target. The majority of the system discussed so far from the hardware to the identification process and Kalman filtering can remain largely unchanged for the other applications as well. What does differ is the estimation stage where choosing the right machine learning algorithm and tuning the parameters to provide good results.

One complete experimental run involves two parts: (1) the System Identification and (2) collecting target distance data. The experiments are executed in order, with the second experiment performed minutes after the identification process. Multiple experimental tests were performed over the course about two months with semi-regularity.

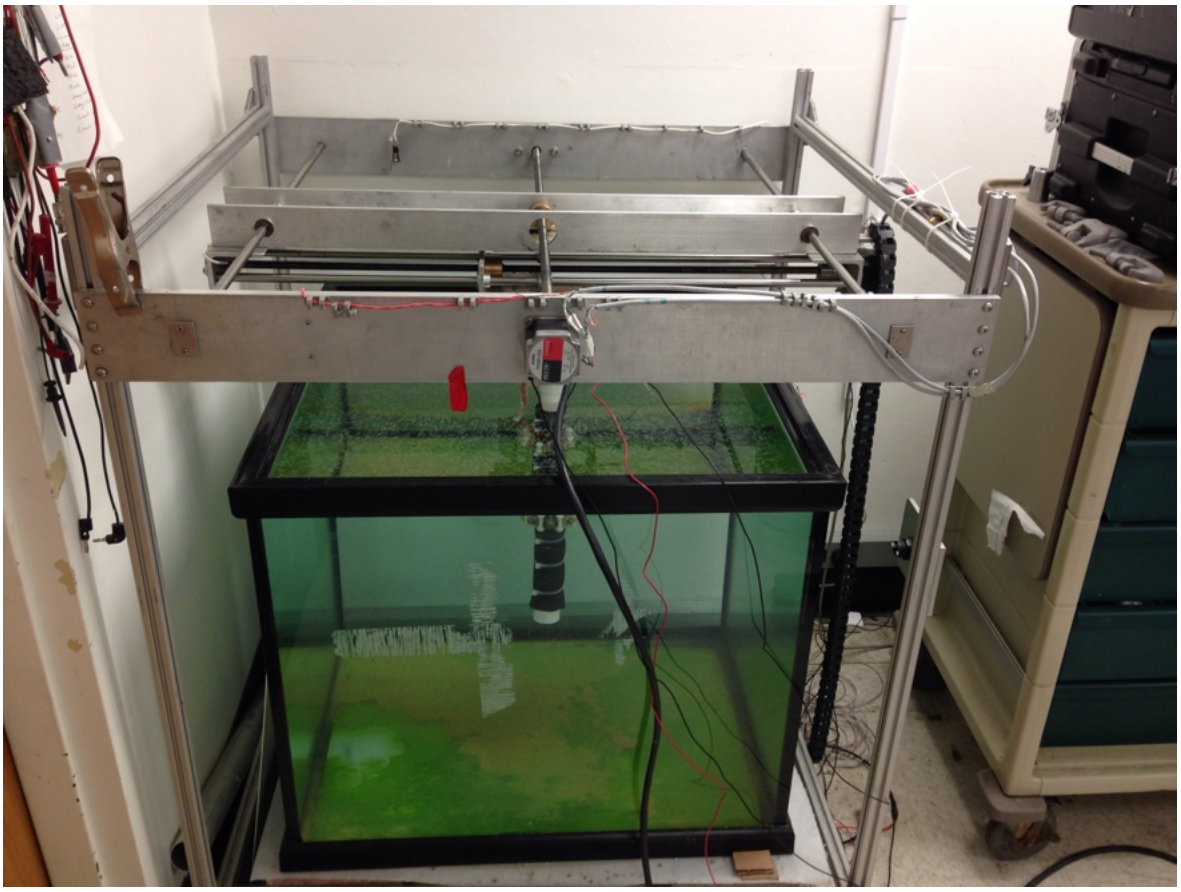

Figure 7.1: The gantry and tank environment with the sensor in the center.

7.2 Test Environment

There are two other major pieces of equipment needed to carry out repeated experiments. One is a sizable tank, which is about 30 by 30 inches, to simulate the underwater environment and the other is a precise gantry system. The gantry serves to move the sensor in a two dimensional grid and also provide ground truth distance measurements. Both of these are shown in the Figure 7.1

7.3 Performing System Identification

When we want to perform System Identification, we want to remove as much of the effect of the tank walls on the sensor values as possible. Logically the sensor will be

placed at the center of the tank. The signal used to perform System Identification was state in the previous chapter to be a pseudo-random binary sequence. The highest frequency in the transmitted PRBS is 500 Hz and the length of the entire PRBS signal will last for 0.5 seconds. The amplitude is between 5 and -5 volts. The output measurements are sampled at 10 kHz, which produces 5000 samples for each electrode.

7.4 Collecting Target Distance Data

For our specific application Figure 7.2 shows how the experiment was initiated and carried out. The aluminum tube will be placed at one end of the tank while the sensor will start the other end. The sensor will step towards the aluminum target and at each stop, a simultaneous electrical discharge and sample will be performed. There are eight electrodes equally spaced on the sensor array. The three electrodes that will be actively sampling are the ones facing the target. The angle of the diagonal electrodes are 45 degrees from the center facing electrode.

The signal used to probe the underwater environment is a 500 Hz square wave signal with amplitudes between 10 and -10 V_{pp}, transmitted for 0.1 seconds, and the output measurements are sampled at 10 kHz, which produces 1000 samples for each electrode.

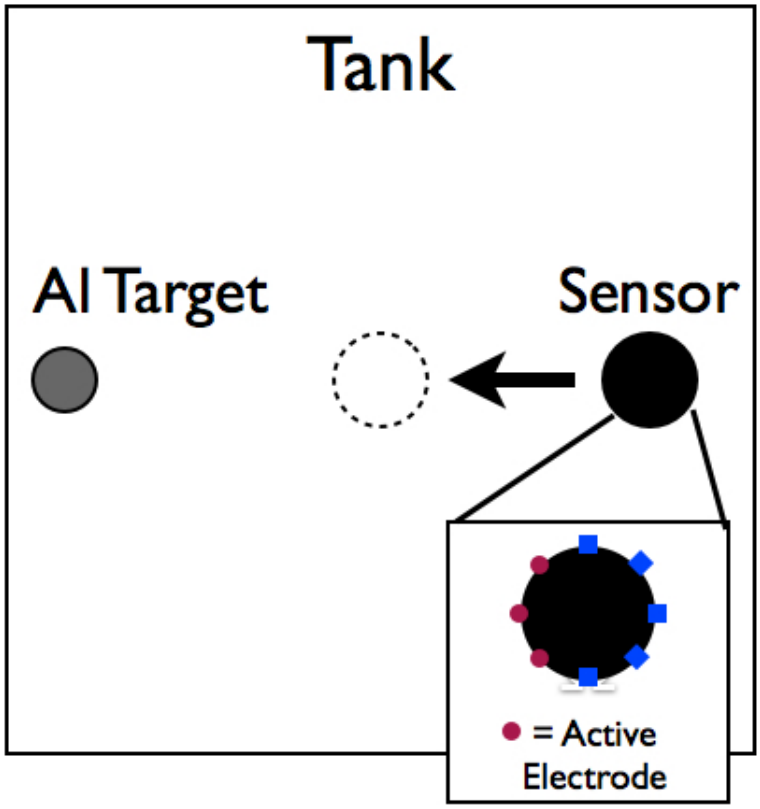

Figure 7.2: This figure shows location of aluminum target and the electrode sweep direction during an experimental run.

CHAPTER 8

Experimental Results

This chapter will start with examining two different ways of training the non-parametric model and showing why we need to adapt the model from time to time to the channel dynamics. We will also show estimation results made from the function trained through linear regression and the discrete classifier trained through SVM. Then we are going to compare the pros and cons of the two. The entire data set consists of 30 experiments which have been collected over two months. Five of the data sets are withheld for testing, while the rest were used for training. Each data set contains an experimental run which consists of data collected at 8 different distances from an aluminum target.

8.1 Training Offline Model

The simplest processing chain we can have is the one shown in Figure 8.1. Here the linear state space model model will be trained offline using training data. In a real operating scenario, the training data will have to be collected through a survey of the environment beforehand. Once the system is trained, the model will be fixed at runtime. The Kalman filter will update vector $w(d)$ based on a static model with each new incoming data. One of the experimental data sets that was chosen as an example is shown in Figures 8.2. These are results from the Kalman filter that had used the static model and have been been normalized and linearized.

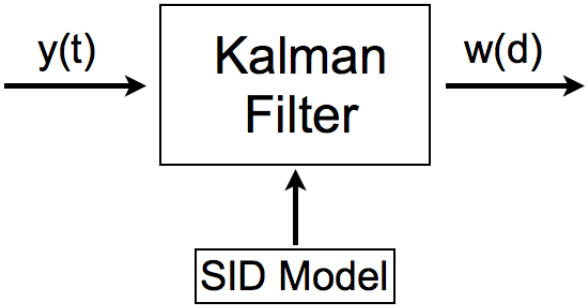

Figure 8.1: Static model

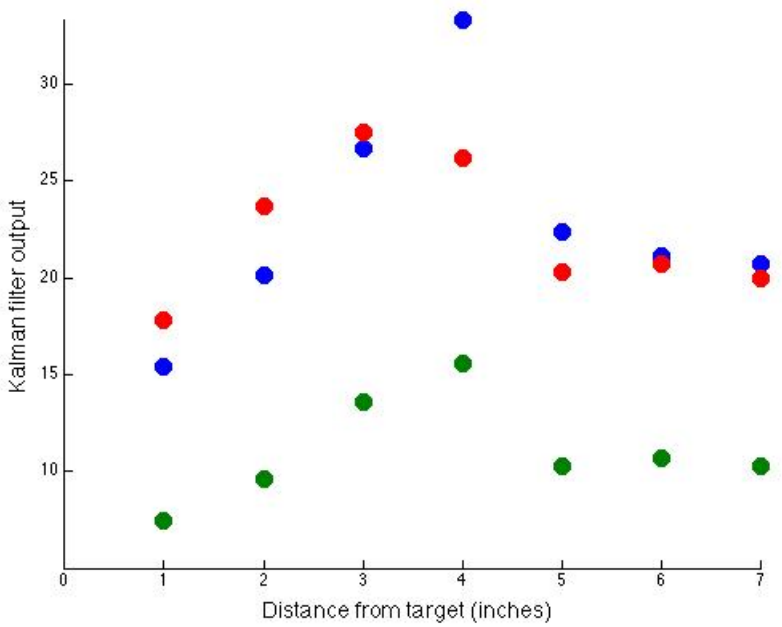

Figure 8.2: Each color represent an element in vector $w(d)$: $w_1(d)$ Blue, $w_2(d)$ Green, $w_3(d)$ Red. Results of the normalized and linearized Kalman filter output using the static model.

Already the results of 8.2 do not look intuitively correct. After using the linearization equation 4.2, the data magnitude should increase with increasing distance. It seems that a third order state space equation is inadequate to account for all the subtle changes in the channel dynamics. However moving towards a higher order may risk running out of computational resources on an embedded system [Sim01].

8.2 Training Online Model

Training a model offline with aggregate data is clearly not working. What has to be done is the model must be updated periodically as shown in Figure 8.3. At some point, the System Identification takes place to remodel the channel before collecting the distance data.

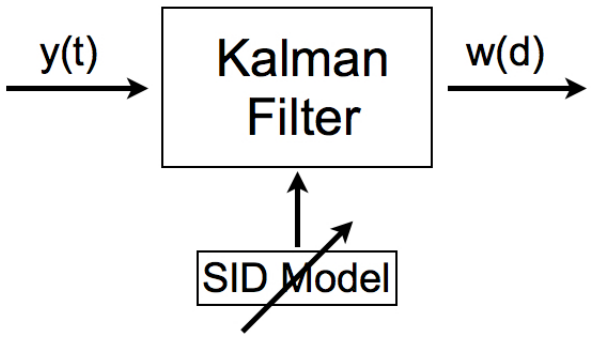

Figure 8.3: Adaptive model

Figures 8.4 shows the same data set as in Figure 8.2, but this time, the model was updated before the distance data was collected in the experimental run.

You can see the improvements garnered from having a fresh model; we now have the expected relationship with distance. It is because we are using numerical methods to discover the model that we can perform this update in the environment without having to change the linear state space equation. The model created in this instance uses only the latest pseudo-random binary sequence. It also seems that we don't need to keep a collection of past data to create a good model, which saves memory. The trade off is that we have spend computational resources to monitor channel dynamics and perform updates to the model, which is the adaptive aspect of this system.

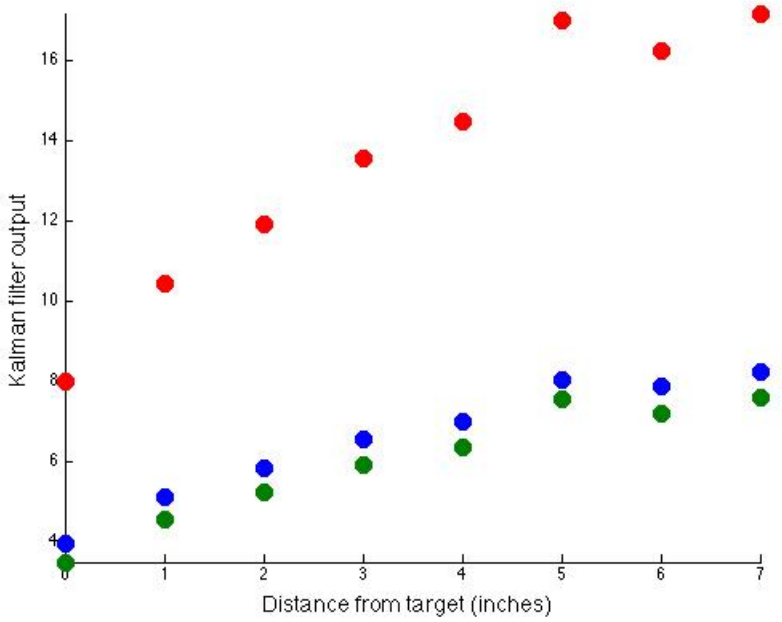

Figure 8.4: Each color represent an element in vector $w(d)$: $w_1(d)$ Blue, $w_2(d)$ Green, $w_3(d)$ Red. Results of the normalized and linearized Kalman filter output using adaptive model.

8.3 Linear Estimation

The results of the estimates using the testing set from the linear function are shown in Figure 8.5 and Table 8.1. The training and testing sets were randomly selected multiple times and the best linear model, with an R-squared statistic of 0.7572, was chosen out of the iterations. The estimations look fairly linear with a slight curve. There is a bias towards an overestimation of the distance except at 7 inches, which is close to our detection limit. The total root mean square error is about 0.8373 inches.

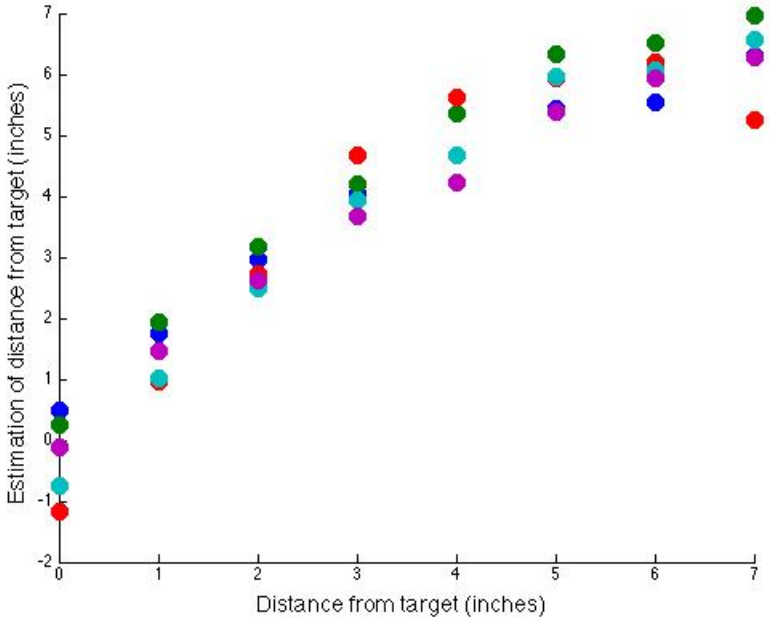

Figure 8.5: Each color represents an experimental run where data at different distances were collected together. Different colors represent experimental runs on different days that were selected at random out of the total set of experimental runs. Results are estimations from the linear estimation function.

Distance (inches)	0	1	2	3	4	5	6	7
Test 1	0.487	1.763	2.949	4.043	4.675	5.442	5.545	6.296
Test 2	0.241	1.923	3.160	4.187	5.361	6.318	6.504	6.957
Test 3	-1.169	0.949	2.723	4.681	5.619	5.933	6.204	5.239
Test 4	-0.750	1.022	2.499	3.925	4.667	5.957	6.062	6.554
Test 5	-0.109	1.453	2.614	3.681	4.218	5.376	5.944	6.266

Table 8.1: Table of values from the linear estimation function.

8.4 SVM Classification

To review, we perform a binary classification using a one vs all grouping for each discrete distance. Sometimes we get multiple positive hits from different classifiers. To reduce the number of positive hits we tighten the bounds of the Gaussian radial basis function to a sigma of 0.5. The test results that still had multiple positive classifications were usually clustered together by distance. For example, a data point with the sensor at 6 inches from the target may have a positive classification

of 6 and 7 inches using SVM. If this is case, we take a simple average among all the distance values that had a positive hit.

The same test data used in the linear function estimate was also tested on the SVM classifier. The results of the classification are shown in Figure 8.6 and Table 8.2 with a total root mean square error of 0.3953 inches. The SVM classifier applies a hard binning to the values rather than an interpolated result from the previous linear estimator. The advantage is lower variances in our estimations but the limitation is we are restricted to pre-selected classification labels.

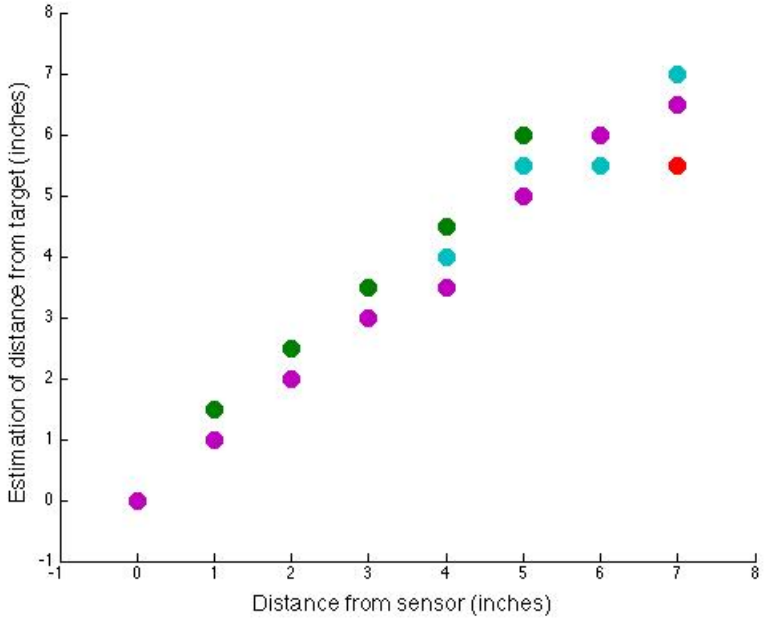

Figure 8.6: Results of classification from SVM. The same test set was used from testing the linear estimation in Figure 8.5

.

Distance (inches)	0	1	2	3	4	5	6	7
Test 1	0	1	2	3.5	4.5	5.5	5	7
Test 2	0	1.5	2.5	3.5	4.5	6	6	6.5
Test 3	0	1	2	3	4	5	6	5.5
Test 4	0	1	2	3	4	5.5	5.5	7
Test 5	0	1	2	3	3.5	5	6	6.5

Table 8.2: Table of results from the SVM classifier.

8.5 Comparing Linear Estimation and SVM Classification

From the analysis in the previous two sections, we see that SVM produces better results but are the improvements justified in using its algorithm. Table 8.3 shows the root mean square error at each distance estimation between the linear estimator and the SVM classifier. The root mean square error from the linear estimator on average is about 0.5 inches higher.

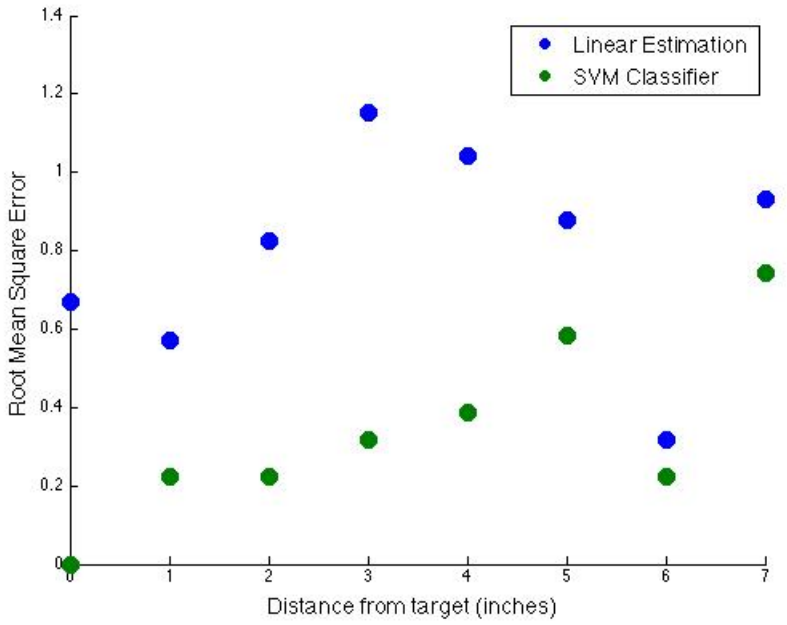

Figure 8.7: The root mean square error of the distance estimation from the SVM classifier is lower than the linear estimation.

.

Distance (inches)	0	1	2	3	4	5	6	7
RMSE (LE)	0.668	0.573	0.824	1.152	1.041	0.879	0.319	0.931
RMSE (SVM)	0	0.223	0.224	0.316	0.387	0.548	0.224	0.741
Δ	0.668	0.350	0.600	0.836	0.654	0.331	0.095	0.190

Table 8.3: Root mean square error from linear estimation (LE) and SVM classifier at each discrete distance. The delta of the two RMSEs are shown in the third column.

Using linear regression has an overall lower computational cost than support vector

machines. To solve for the coefficients of a linear estimation function, we showed that two steps are required: QR factorization and the Least Square equation. Both algorithms are complexity $O(n^3)$ where n is the size of the matrix [Sim01] [Par00]. On the other hand SVM poses a convex optimization problem and the complexity is dependent on the convex solver. There is a higher possibility of being able to implement linear regression on a limited resource system if necessary. This would provide even more adaptability in the environment if the system can update the coefficients of the linear estimator. As for support vector machines, typically computations for the training stage are offline and not performed in real time.

If we consider the estimation stage with the linear function or classifier already trained and deployed for usage, then approaching the problem using linear estimator has only a slight advantage over the SVM classifier. The linear estimator is a simple inner product between two vectors while the SVM classifier complexity depends on the kernel that is used. Since we use a Gaussian radial basis function, there is an additional exponential computation. However this is not constraint that is difficult to overcome.

CHAPTER 9

Conclusion

I showed in this thesis a numerical approach using System Identification to model the charge interaction through an underwater channel, an alternative to using the loaded physical models. Using the model information in a time varying Kalman filter, we can make estimations about target distance based on the differences between the model estimation and the observable data. My initial analysis of using an offline model showed poor Kalman filter results. In order to remedy this issue, we adapt our system to the channel by performing a model update before we take distance measurements. This adaptation method improves the linearity of the normalized Kalman filter output.

The scenario we used to test our processing chain was estimating the distance of an aluminum object. The two approaches used to build the estimation function was the linear regression algorithm, for building a continuous estimation function, and the support vector machines algorithm, for building a discrete classifier. We showed that although linear regression worked fairly well, SVM trained a better estimator with lower root mean square error. Both have their advantages and disadvantages in that a trained linear estimation function can make continuous value estimations through interpolation while SVM can only give discrete estimates.

The advantage of this system approach in this thesis is that we make no assumptions about the physical properties of the channel nor do we incorporate simplifications in the geometry of the electrode placement. The proposed state space model and processing blocks are also implementable in a micro-controller

of an autonomous vehicle. The work and analysis done in this thesis are in fairly ideal settings but it is possible, should someone choose, to build an advance robotic prototype.

9.1 Future Work

This work was targeted at working in an underwater environment, which is in many ways the most hazardous working environment for autonomous vehicles. There are still questions that need to be answered such as how well this method works in a real ocean environment and how often does the model need to adapt in time and space. Since collecting a good distribution of data and having a good model is key, what is the best way to have autonomous vehicles collaborate with each other to lessen the burden of learning and adapting in a big ocean environment. The methods proposed in this thesis shifts the engineering focus more towards the application and modularity of each processing blocks makes it easier to add or remove steps that suits the needs of the scenario.

References

[Emd98] G. von der Emde. "Capacitance detection in the wave-type electric fish *Eigenmannia* during active electrolocation." *J. Comp. Physiol*, **A 182**:217–224, 1998.

[Emd99] G. von der Emde. "Active electrolocation of objects in weakly electric fish." *J. Exp. Biol*, **202**:1205–1215, 1999.

[HCL10] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. "A Practical Guide to Support Vector Classification." Technical report, National Taiwan Univeristy, Taipei 106, Taiwan, 2010.

[Her13] Henry E Herman. *"Electrostatic Sensing for Underwater Object Detection and Localization."*. Master's thesis, Universtiy of California, Los Angeles, 2013.

[JKM07] Solberg J, Lynch K, and MacIver M. "Robotic electrolocation: active underwater target localization." *Proceedings of the International Conference on Robotics and Automation*, 2007.

[Kra96] Bernd Kramer. *Electroreception and Communication in Fishes*. Gustav Fischer, Germany, 1996.

[LCG13] Vincent Lebastard, Christine Chevallereau, Alexis Girin, Noël Servagent, Pol-Bernard Gossiaux, and Frédéric Boyer. "Environment reconstruction and navigation with electric sense based on a Kalman filter." *The International Journal of Robotics Research*, **32(2)**:172–188, 2013.

[Lis51] H. W. Lissmann. "Continuous electric signals from the tail of a fish, *Gymnarchus niloticus* Cuv." *Nature*, **167**:201–202, 1951.

[LJR13] Chen L, House J, Krahe R, and Nelson M. "Modeling signal and background components and electrocsensory scenes." *Journal of Comparative Physiology*, **A 191**:331–345, 2013.

[LM58] H. W. Lissmann and K. E. Machin. "The mechanism of object location in *Gymnarchus niloticus* and similar fish." *J. Exp. Biol*, **35**:451–486, 1958.

[LZ06] Ioan D. Landau and Gianluca Zito. *Digital Control Systems: Design, Identification and Implementation*. Springer-Verlag, London, 2006.

[OM94] Peter van Overschee and Bart De Moor. "N4SID: Subspace Algorithms for the Identification of Combined Deterministic Stochastic Systems." *Automatica*, **Volume 30, issue 1**:75–93, 1994.

[Par00] B. N. Parlett. "The QR Algorithm." *Computing in Science and Engineering*, **2**:38–42, 2000.

[Sim01] Dan Simon. "Kalman Filtering." *Embedded Systems Programming*, June 2001.

[Sim06] Dan Simon. *Optimal State Estimation.* Wiley and Sons, Inc., Hoboken, New Jersey, 2006.

[TB79] M. J. Toerring and P. Belbenoit. "Motor programmes and electroreception in mormyrid fish." *Behav. Ecol. Sociobiol*, **4**:369–379, 1979.

[TM84] M. J. Toerring and P. Moller. "Locomotor and electric displays associated with electrolocation during exploratory behavior in mormyrid fish." *Behav. Brain Res*, **12**:291–306, 1984.