A Software and Hardware System for the Autonomous Control and Navigation of a Trained Canine

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Winard Britt

Certificate of Approval:

David M. Bevly, Co-Chair
Associate Professor
Mechanical Engineering
Auburn University

John A. Hamilton, Jr., Co-Chair
Professor
Computer Science and Software Engineering
Auburn University

Saad Biaz
Associate Professor
Computer Science and Software Engineering

George T. Flowers
Dean
Graduate School

A Software and Hardware System for the Autonomous Control and Navigation of a Trained Canine

Winard Britt

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 10, 2009

A Software and Hardware System for the Autonomous Control and Navigation of a Trained Canine

Winard Britt

Signature of Author

Date of Graduation

Vita

Winard Ronald Britt, son of Ronald H. Britt and Catherine S. Britt, was born September 15, 1982 in Montgomery, Alabama. In 2000, he graduated with Highest Honors from Booker T. Washington Magnet High School in Montgomery. He attended Auburn University, graduating Summa Cum Laude and University Honors Scholar with a Bachelors of Software Engineering in 2004. In 2007, he graduated with a Master of Science in Computer Science.

Dissertation Abstract

A Software and Hardware System for the Autonomous Control and Navigation of a Trained Canine

Winard Britt

Doctor of Philosophy, August 10, 2009
(Master of Science., Auburn University, 2007)
(Bachelor of Software Engineering, Auburn University, 2004)

157 Typed Pages

Directed by David M. Bevly and John A. Hamilton, Jr.

This dissertation demonstrates the autonomous command and navigation of a trained canine to multiple waypoints. A system is described consisting of a canine that can be guided autonomously to a number of waypoints by an automatic software control algorithm. A hardware system has been developed in order to interface with GPS, accelerometers, gyroscopes, magnetometers, and tone and vibration generators for the purpose of accurately commanding and dictating the motion, path, and commands given to a canine. A canine has been trained to effectively follow audio and vibration commands for guidance with a high degree of accuracy (71% mission success for simple paths and 63% mission success for complex paths). Both a Neural Networks approach and a State Machine Based approach to canine anomaly detection are presented, as well as strategies for anomaly correction. An operational control algorithm for autonomous guidance of the canine is described in detail. Finally, empirical results of an autonomously commanded canine are demonstrated with an 73% mission success rate for simple paths and a 62% mission success rate for complex paths.

Acknowledgments

- The K-9 project was financially supported by an ONR YIP award N00014-06-1-0518.
- The Information Assurance Scholarship Program materially supported my doctoral work.
- Thanks to my graduate advisers and committee members for their support, advice, and encouragement.
- Thanks to the Auburn University Veterinary School and the Canine Detection Research Institute, particularly Dr. Paul Waggoner, Daniel Johnson, and Tim Baird for their time and efforts.
- Thanks to the engineers in the GAVLAB for helping a Computer Scientist learn to do Electrical and Mechanical Engineering work (a little of it, anyway). Particular thanks go to Ben Clark, David Hodo, Matt Lashley, and Will Travis. Further thanks go to GAVLAB alum Hank Henderson and Rob Daily.
- Thanks to William Lyles for his tireless dedication to the K-9 project in a wide variety of technical tasks.
- Thanks to Stephan Henning for his work on the circuitry for the sensor board.
- Thanks to Major and Aries, the two friendly canines who participated in these studies.
- Thanks to Brennan Sweeney for his enhancements to the Google Earth plotter scripts used in some of the Google Earth images presented in this work.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliography follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Trained canines have historically proven very effective in a wide range of potentially dangerous security applications such as the tracking and detection of people, drugs [Curtis and Cupp 1989], and explosives [Bureau of Diplomatic Security 2004]. However, canine units generally require the guidance of humans with some animal handling expertise to perform their tasks. Specifically, most trained canines act as an augmentation to existing human teams, rather than autonomous units in and of themselves. Relatively autonomous units, such as robots, suffer from a number of deficiencies (a lack of smell and intelligence, for example) that canines do not. Therefore, there exists an opportunity to take greater advantage of the sensors and mobility of canines by incorporating autonomous control to canine teams.

A number of researchers have utilized electrodes and strong stimuli for the purpose of direct guidance of creatures including pigeons, rats, sharks, and even cockroaches [United Nations University 2001, Talwar et al. 2002, Song et al. 2006, Brown 2006, Gomes et al. 2006, Shandong University of Science and Technology 2007]. This can raise practical and ethical concerns when dealing with creatures like canines. Other research efforts have examined tracking and virtual fencing of cows to both contain herds and to prevent overgrazing [Schwager et al. 2008, Correll et al. 2008].

The primary goal of this research is to develop algorithms which utilize information available from sensors on-board the canine to provide audio and vibration command and control signals for the purpose of autonomously directing the canine to waypoints. Once the

canine is deployed for a given path, all command information should be produced using data collected on the system, without the need for human guidance. The specific contribution of this work is to model the guidance commands produced by the human operator using the canine behaviors as input, as described by GPS and sensor information. Once behaviors have been identified, appropriate commands should be given (for example, a correcting command when the canine begins to deviate from his path). Hardware and software have been developed to support this application. An eventual goal of the work would be to not only guide the canine to a predetermined location, but also to be able to recognize when a canine detects something of interest. For example, the canine could demonstrate some known response (walking in a circle or sitting down) upon detecting narcotics.

In essence, determining whether or not a canine is behaving correctly at any given time can be considered a modeling problem, where the model is that of a human operator. A human operator can quickly identify the heading, location, distance to destination, and other parameters to make a decision whether or not the canine needs to adjust course. In order to automatically determine whether or not the canine is behaving correctly, a control algorithm needs to be able to process information regarding location, heading, and movement behaviors much like a human observing the canine would. In this dissertation, Machine Learning (ML) algorithms were utilized on processed sensor data gathered from an embedded system developed for the purpose of canine navigation. The ML algorithms were trained on this data and then used to classify future, unseen behaviors by modeling the commands given by a human operator. Evolutionary Algorithms (EAs) have been used to optimize the system parameters for the ML algorithms.

Machine Learning algorithms have been used for a wide variety of control [Chowdhury et al. 2001] and modeling problems [Britt et al. 2007]. In the task of vehicle control, ML has been used to model the control of the vehicle from available sensors such as in [Glasius et al. 1994, Davis 1995, Baluja 1996, Kalkkhul et al. 1999, Xu and He 2002, van de Ven et al. 2005]. In these cases, typically sensor inputs (heading, velocity, and location) are collected and represented in some fairly direct fashion and then commands like steer angle can be predicted. ML has further been applied to both guidance and navigation applications in robotics [Ma and Yuan 1995, Tani and Fukumura 1996, Floreano and Mondada 1998, Fierro and Lewis 1998, Araujo and de Almeida 1999, Yang and Meng 2003, Islam and Murase 2005, Ye 2008], limb and locomotion control [Cymbalyuk et al. 1998], and concept learning [Mahadevan and Theocharous 1998, Sporns and Alexander 2002]. In biological modeling applications, machine learning has been successfully applied to a wide array of diverse tasks including the modeling of echolocation in bats [Simmons et al. 1995], echolocation in dolphins [Helweg et al. 1993], horse locomotion [Calvert et al. 2003], human motor control [Wolpert and Kawato 1998, Dean and Porrill 1998, van Heijst et al. 1998], and animal navigation [Mudra and Douglas 2003].

However, system parameters for MLs generally must be optimized to achieve good performance in terms of success and efficiency. Evolutionary Algorithms (EAs) have a rich history of being used to successfully optimize machine learning algorithms in an array of applications, such as process parameter optimization [Cook et al. 2000, Ferentinos 2005] and mobile robot control [Shinchi et al. 2001]. For a more in-depth literature review on optimization of machine learners, see Section 2.3.2.

Machine Learning algorithms have proven successful at accurate classification, even in circumstances with relatively few training instances and with incomplete data. In this problem, since the demonstrable behaviors of the canine are practically infinite, the data will always be an abstraction of the idea of a behavior. The initial investigation into the canine anomaly detection problem utilized a machine learning approach with results that demonstrated that such a technique is possible. However, as the hardware system evolved, an alternate state machine approach to canine guidance proved effective using rules rather than a machine learner. The state machine algorithm is the algorithm utilized in the autonomous canine system described in Chapter 6.

1.1 Goals

The work in this dissertation fulfills a number of primary goals. The following list itemizes the goals followed by a brief synopsis of how those goals were fulfilled and where to find that content in the manuscript.

- A contribution to canine guidance in the form of a "remote control" canine unit with an advanced sensor pack, using a canine that responds to stimuli such as tones and vibrations. This is validated through trials on a live canine. Extensive field trials with a live canine coupled with analysis of the command and navigation of the canine is discussed in Chapter 5.

- A contribution to canine guidance in the form of being able to classify canine behavior as "on course" or "off course." This is validated using sensor results produced from a previous model of sensor pack in conjunction with a live canine. A discussion of and results for modeling canine behavior in this fashion is provided in Chapter 4.

- General state machine control algorithms for the guidance of autonomous canine units. These control algorithms should be able to be reparameterized for different canines or mission objectives. The control algorithm and the parameterization thereof are discussed in Chapter 6.

- To produce the ability to determine if a canine model is failing to predict commands due to the canines exhaustion or reaction to unobservable events. Mechanisms for failure detection are discussed in Chapter 6.

1.2 Challenges

This research faced both common obstacles and novel issues compared to traditional control and modeling problems. The key project challenges are as follows:

- Modeling the guidance of a canine traditionally provided by a human operator is difficult, since the human possesses a considerably wider range of senses (vision and awareness of the total environment) with which to make judgments about issuing the tones to the canine. The information provided by sensors is a small subset of the information available to the human being.

- While vehicles and robots exhibit fairly predictable, deterministic behavior, canines exhibit considerably more variability in their movements and behaviors. This required a modeling algorithm that was able to tolerate both considerable noise and non-deterministic responses from the canine.

- The canine used in this study is trained for multiple purposes. Training a canine for a wide variety of missions makes them less precise in their response to guidance commands. This proves to be a significant source of canine error.

- Numerically describing the behavior of a canine is a research task. It is not necessarily straightforward to represent a behavior as a vector of numerical values which are needed for the anomaly detection algorithms. In other words, capturing the concept of a canine behavior is difficult.

- New hardware had to be integrated and interfaced with sensors for each new phase of the project. Further, custom software had to be written for the embedded microprocessor to collect the data at each phase.

- The canine's rapid movements are brutal to hardware leading to the nearly constant need for repairs, maintenance, and redundancy.

- The ability to perform fast, accurate anomaly detection in canines does not automatically translate into improved mission success. Perhaps counter-intuitively, it is sometimes preferable to allow the canine to exhibit anomalous behaviors for a longer period of time rather than to attempt corrections more often.

- Evaluating the effectiveness of a control algorithm through field trials is extraordinarily time-consuming, providing a limit on the amount of different control algorithms that could be empirically verified on a live canine.

1.3 Outline

The remainder of this dissertation is organized as follows:

1. Chapter 2 gives a high-level discussion of the proposed ML algorithms, evolutionary algorithms, and past efforts optimizing MLs with EAs. Although ultimately a neural network strategy was not the chosen algorithm, this literature review still provided design strategies for modeling anomaly detection.

2. Chapter 3 provides an architectural view of each phase of the research, including the initial prototype, the training architecture, and the final autonomous canine system.

3. Chapter 4 discusses the initial modeling effort focused on classifying canine movements to determine if the canine is on or off course.

4. Chapter 5 describes in detail the hardware and software used to command the canine and to perform navigation during. Additionally, experimental results are described providing data on the performance of the canine in response to human given commands.

5. Chapter 6 describes the hardware, software, and algorithms needed to autonomously command the canine. Further, experimental results and analysis of autonomous control trials are provided.

6. Chapter 7 enumerates the key contributions of the dissertation, describes areas of future research, and provides some final concluding remarks.

Chapter 2

Theoretical Background

2.1 Chapter Introduction

This chapter introduces concepts relating to classification and optimization that are broadly applicable to many different types of problems. In this case, the ability to classify canine behaviors is of interest and the ability to optimize a classifier for this purpose is also of interest. Several different modern strategies for machine learning classification will be discussed. In additional, strategies for algorithm optimization will be discussed.

2.2 Machine Learning

2.2.1 Introduction

Machine Learning (ML) broadly defines algorithms created with the intent of allowing computers to learn patterns or rules from data observations. A large family of algorithms constitute ML, including decision trees (for example, [Quinlan 1993]), inductive logic programming [Muggleton 1991], K-Nearest Neighbor [Fix and Hodges 1951], perceptrons [Rosenblatt 1957], neural networks [Haykin 1999], radial basis function networks [Moody and Darken 1989], and support vector machines [Cortes and Vapnik 1995]. For classification problems (such as determining whether or not a canine's behavior is on-course or off-course) the algorithms typically learn by analyzing patterns in datasets where the classification is known (training data) to build a model which can then be applied to new, unseen data (test data).

Terminology

Throughout this work, an effort has been made to keep terminology as close to generally accepted terms as possible. For the convenience of the reader (or for readers unfamiliar with Machine Learning), this section will describe terms relating to ML.

instance A single vector of numerical values relating to a single entity to be classified.

desired output The correct classification value associated with a training example.

training instance An instance used for the purposes of developing a model in the Machine Learning algorithm.

validation instance An instance used to test a trained model for the purpose of determining system parameters and other settings that are not directly determined by the training process.

test instance An instance not used for training or for the determination of system parameters that can be used to estimate the ability of a model to generalize. These instances are used to evaluate the success of a model.

unseen instance A new instance that was not used to train or validate the model. Test instances should always be unseen instances.

generalization A desirable property of machine learning models is that they perform well on data that was not used to train. A machine learning model that performs well on validation data, but poorly on test data is said to generalize poorly.

overfitting When a ML algorithm "memorizes" training data during the training process, but then performs very poorly on unseen data [Engelbrecht 2002]. Typically, overfitting is prevented by periodically testing the current model on unseen data to verify that the model is not being overfit.

offline training time The time required to actually develop a model for a machine learner. In dynamic applications where a model must be learned frequently, this becomes a concern.

classification time The time it takes to apply a learned model to a new unseen instance. Typically, this is the most important time consideration since many applications can tolerate a large amount of offline training time as long as the classification time remains low.

For this work, two different Machine Learning algorithms are utilized: General Regression Neural Networks (GRNNs) and a Parameterized State Machine. In addition, in this literature review two popular algorithms were chosen to be discussed: Support Vector Machines (SVMs) and Radial Basis Function Networks (RBFNs). Each of these algorithms was included for a different reason. The GRNN was included due to its ability to classify with only a very small amount of training data [Specht 1991] as well as its similarity to very mature statistics-based K-Nearest Neighbor algorithms [Fix and Hodges 1951]. However, GRNNs require that all the training data is stored in the model which can be non-trivial. The State Machine approach was chosen since it made incorporating the human knowledge of the trainer and the natural state-based structure of the control problem more accessible in the face of limited training data. RBFNs have the ability to have smaller models than GRNNs since they do not have to store all the training data in the model. Since model size

is a concern for implementation on an embedded device, this has a strong appeal. However, deciding topology is not always straightforward. SVMs were chosen due to the availability of mature, freely available implementations in SVM_{light} [Joachims 1999] and SVM_{perf} [Joachims 2005; 2006] as well as being a state-of-the-art classification methodology [Russell and Norvig 2003]. In fact, it has been demonstrated that SVMs perform very well on a wide variety of problems when compared to other algorithms [Meyer et al. 2003].

To solve a classification problem with ML, a general strategy was defined from past efforts:

1. Determine an objective means to represent the entity to be classified. This is not always a straightforward procedure, given that numerically representing any entity inherently abstracts away some of its properties. For example, while the inputs to a mathematical function may naturally translate to a representation, deciding how to represent a text document for author attribute classification is an open research question. Similarly, determining how to represent concepts such as "behaviors" for a canine is a non-trivial task.

2. Collect numerous representative instances to train a model.

3. Choose some ML algorithm (for example, a Neural Network) based on the needs of the application. ML algorithms vary in terms of generalization quality, offline training time, online application time, and model size. This requires an understanding of resource constraints and needs.

4. Split the instances into three sets: training, validation, and test. Typical splits might be 70/15/15 or 60/20/20 among training, validation, and test.

5. Train a model on the training set. This is done to improve the weights in neural networks and to determine support vector locations in SVMs.

6. Apply the model to the validation set to get an estimate of its quality in terms of an appropriate metric. Popular success metrics include accuracy, F1 Score [van Rijsbergen 1979], precision/recall, and (for regression problems) mean-squared-error.

7. Iteratively adjust any system parameters that are not learnable through the basic training algorithm. Examples of this include topology of the learner, learning rates, or even the features selected to represent an instance.

8. Finally, choose the best model and apply it to the test set. These results provide an indication of the model's ability to generalize.

In general, the effectiveness of a ML depends on [Haykin 1999]:

1. The presence of learnable characteristics in the training instances. In other words, are the features chosen to comprise the instances actually correlated to the desired classification?

2. The number of training set instances that are available. Too few training instances can lead to the algorithm lacking the ability to generalize. Too many training instances can cause the model to become large, slow to train, and slower to test.

3. The representativeness of the training set. Specifically, does the distribution of training instances resemble the real application? Large, diverse training sets typically improve the performance of the resulting model.

4. System parameters of the Neural Network, which must be determined in the validation phase.

2.2.2 General Regression Neural Network

A General Regression Neural Network (GRNN) is one of a sub-type of MLs referred to as Neural Networks. Neural networks attempt to emulate the biological neural networks in the brain, which depend on massive parallelism and interconnectivity to process information quickly.

A GRNN is a one-pass learning algorithm which attempts to approximate a continuous variable that is dependent on many representative vector training instances. This is advantageous over many other neural network strategies which require iterative learning, since they often require many training iterations to produce workable solutions [Specht 1991]. In practice, any analysis system to be deployed on an embedded system with relatively low computational resources should be made to be as simple and efficient as possible.

As shown in Figure 2.1 (taken from [Specht 1991]), elements of training instances $x_1...x_n$ are passed into the network (Input Units) and collected into pattern units (where each unit represents a single training instance). The output from the pattern units (shown in the diagram as $A^1..A^n$ which comprises the numerator in Equation (2.1) and $B^1...B^n$ which comprises the denominator in Equation (2.1) is collected in the summation units (as described below) and then the output of the summation units is ultimately collected to provide an estimate for $\hat{Y}$ corresponding to an unseen input vector X.

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} Y_i * exp(-\frac{C_i}{\sigma})}{\sum_{i=1}^{n} exp(-\frac{C_i}{\sigma})} \tag{2.1}$$

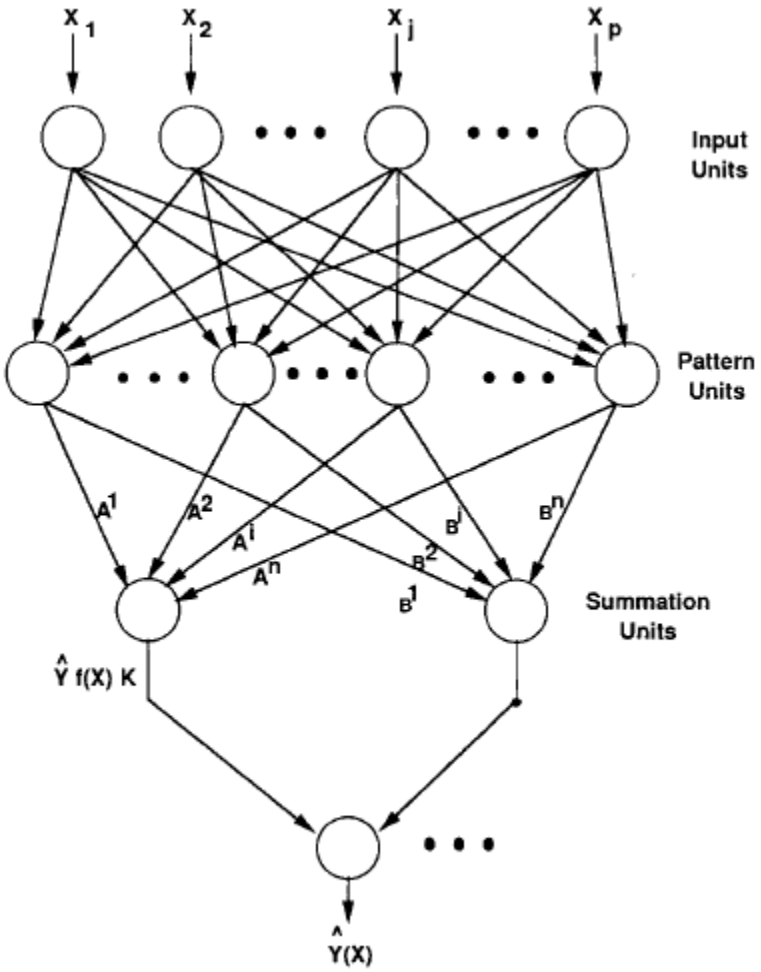


Figure 2.1: A GRNN Block Diagram from [Specht 1991]

Each training instance in the GRNN consists of a vector T and a desired output Y. Providing an estimate of the value of $\hat{Y}$ for an unseen instance X is shown in (2.1):
where n is the number of training instances, Y_i is the desired output for a given training instance vector, σ is a constant system parameter of the GRNN, and the value of C_i is the distance squared between X and a given training instance vector T, as shown in (2.2):

$$C_i = \sum_{i=1}^{p} (X_i - T_i)^2 \tag{2.2}$$

where T_i represents a single element of the training instance vector, X_i represents the corresponding element in the instance to be classified, and p is the number of elements in the vector.

While the GRNN provides the useful ability to begin classifying with very few training instances (since it simply applies a sophisticated K-Nearest Neighbor scheme), this comes with the cost of a need to store all of the training data in the model. For large training sets, this memory cost is prohibitive for embedded systems.

2.2.3 Support Vector Machines

Support Vector Machines (SVMs), developed by Vapnik in [Vapnik 1995; 1998] with a tutorial explanation given in [Burges 1998], are statistical learners which consist of support vectors which (as best as possible) attempt to divide data into one of two different classifications. For example, given data describing a canine action, an SVM could be designed to divide behavior instances into on-path or off-path classifications. This is accomplished by learning from available training data and finding the locations for the support vectors which maximize the distance between the training elements of each class. In other words,

the support vectors should divide the data in such a way as to best divide up the vectors into separate classes. In practice, the training of the SVM is treated as a quadratic programming problem solved through numerical means.

Motivation

SVMs offer some advantages and disadvantages when compared to the other machine learners presented in this work. One of the primary advantages is that once the support vector model is learned from the training data, the model can be represented fairly compactly (compared to GRNNs, which must retain all of the training data as part of the model itself). However, this potentially presents a disadvantage in that the process of learning the model can lead to a dramatically longer offline training phase. In applications where offline training time is an issue, this could pose a significant disadvantage. In many applications, only the online runtime (the time it takes to classify a new unseen instance) is of concern, and SVMs can be implemented to perform well in this situation [Joachims 1999].

2.2.4 Radial Basis Function Network

Radial Basis Function Networks (RBFNs) are a form of ML known as neural networks. In [Moody and Darken 1989], one of the earliest approaches to RBFNs, the authors define an architecture with a single internal layer of processing units. The single layer strategy relies on identifying a reasonable number of "centers" which should ideally be near the middle of clusters (or classifications) of instances. One cluster should contain only instances from one class, but it is possible that multiple clusters belong to the same class. Given a good distribution of cluster centers, it is easy to identify where a new instance is based on its distance from said cluster centers. Although the strategy in general requires fewer weights

to learn than multi-layer feed forward neural networks (typically optimized by the very slow backpropagation algorithm), RBFNs still need one layer of weights and a good way to determine the number and location of clusters. As described in [Tao 1993], the general procedure for creating an RBFN is as follows:

1. Use some subset of the training instances as clusters centers. If all centers are used, this approach becomes more similar to a GRNN.

2. Determine appropriate system parameters for the kernel functions (for example, the value of the standard deviation for a Gaussian Kernel).

3. Solve for the values of the weights (see further discussion in Section 2.3).

A number of methods have been proposed for determining good locations for the centers in RBFNs, some based heavily in mathematics such as Orthogonal Least Squares Learning [Chen et al. 1991] and Learning Vector Quantizer [Kohonen 1989; 1997]. Other methods that have been suggested choose some fixed number of centers randomly from the available training data or simply use all the available training data as centers (this strategy is typically not-scalable for large amounts of training data). In general, more effort to determine good (representative) centers can improve performance of the RBFN in terms of classification accuracy and computational time. However, this gain comes at the cost of offline training time.

2.3 Optimization of Machine Learners

Traditionally, the system parameters and weights for a ML must be discovered through some optimization procedure. Many techniques have been proposed to assign appropriate

values to weights. Typically, analytically solving for the weights is infeasible [Barnard 1992], so iterative techniques such as backpropagation [Rumelhart et al. 1986] or stochastic techniques like evolutionary algorithms (see Section 2.3.2) are typically utilized to discover the weights. For SVMs, the required optimization procedure reduces to a quadratic programming problem, for which practical implementations of the optimization procedure exist (for example, [Joachims 1999]). This section briefly discusses a few of the more common traditional optimization approaches for ML, as well as providing a survey of some related evolutionary approaches to ML optimization.

2.3.1 Optimization using Traditional Approaches

A traditional iterative, deterministic approach to the weight optimization problem for Neural Networks (such as the RBFN or Feed-Forward Neural Networks) is to establish a criterion function relating training sample inputs and weights to training sample outputs (the desired outputs). The "training" of the neural network is a procedure that assigns values to the system weights such that when a new, unseen sample passes through the network, the inputs modified by the weights will produce an output that approximates the desired output as closely as possible. In other words, the criterion function relating the inputs and weights to the desired output should be small if and only if the combination of the inputs and the weights is approximately equal to the desired output. Put this way, the training process involves optimization (minimization) of the criterion function for a large number of training samples. Several deterministic optimization approaches such as backpropagation, conjugate gradient, and steepest-descent techniques seek to minimize

the criterion function (for a comparison of several deterministic and stochastic traditional methods, see [Barnard 1992]).

One of the general problems with deterministic methods of weight optimization has to do with local minima, or a sub-optimal solution discovered by the algorithm. Since the optimization of the weights in a neural network results in the equivalent of a search problem, an algorithm can find a local good solution which does not result in the best set of weights. A deterministic search algorithm runs the risk of discovering a good solution, but failing to effectively search elsewhere since all possible localized steps lead to lesser solutions. Additionally, deterministic algorithms like gradient descent typically have difficulty optimizing system topology parameters such as the number of nodes and connections in a Feed Forward Neural Network [Yao 1992]. Specifically, the topology optimization problem is complex, noisy, non-differentiable, multimodal, and deceptive making it very difficult for traditional optimization procedures [Miller et al. 1991]. This presents a non-trivial problem, since the network topology significantly impacts the performance of a neural network. This suggests the application of a stochastic approach.

2.3.2 Evolutionary Algorithms

This section provides a brief description of the stochastic search methodologies collectively referred to as Evolutionary Algorithms (EAs) or Evolutionary Computation (EC) [Holland 1975, Koza 1992, Back et al. 1997]. For this work, the term "Evolutionary Algorithm" will be applied consistently.

EAs have been shown to be effective in a wide variety of difficult search problems including function optimization [Golberg 1989], distributed constraint satisfaction [Britt et al.

2006, Dozier et al. 2007], space vehicle optimization [Dozier et al. 2006], game-playing [Fogel 1995], and many others [Spears et al. 1993]. EA encompasses several types of algorithms, which work off of the principle of survival of the fittest. In the context of neural network optimization, evolutionary algorithms offers a stochastic approach to searching for effective weights, network topologies, network parameters, and input features.

Key terminology on EAs appears as follows:

candidate solution A single vector which represents a possible solution to a problem. In the case of optimizing MLs, this could take the form of a specific set of system parameters.

fitness The objective, numeric quality of a candidate solution. This is produced by a numeric fitness function. The fitness of a candidate solution to a classification problem could be the success rate. Multi-objective fitness functions are possible.

population In an EA, there may be one to many candidate solutions at any given time which collectively form the population.

mutation An operator used to produce a new candidate solution (a "child") which is similar to, but not the same as, another candidate solution (a "parent").

crossover An operator used to produce a child from two parents using some attributes from both.

A more detailed description of the specific algorithms used in this work appears in the chapters in which they appear. In general, EAs create a population of candidate solutions (CSs) to a search problem, evaluate their quality, and then perform mutations and combinations to create new CSs. Those new CSs are also evaluated in terms of quality. Next, a

replacement of weak CSs occurs. This is typically repeated iteratively until some quality level is achieved or some amount of time passes.

EAs vary in terms of the way in which population replacement is performed, when child CSs are created, and in the way that mutation and perturbations occur to create child CSs. This section discusses two common EAs as well as a survey of EAs applications to ML optimization.

Evolutionary Hill-Climber

In an Evolutionary Hill-Climber (EHC), there is only one CS in the population at any given time. Children are created via mutations and only replace the parent if they have higher fitness. This strategy has the advantage of being very simple to implement and tends to be strong at function optimization. However, depending on the disruptiveness of the mutation operator, there can still be issues with converging prematurely to a local minima (as seen in the deterministic optimization approaches).

The Steady-State Genetic Algorithm

The Steady-State Genetic Algorithm (SSGA) differs from the EHC in that it has multiple candidate solutions in the population at any given time and includes the ability for crossover operations (combinations of multiple CSs to produce the child CS).

First, the SSGA [Davis 1991] generates a fixed size population of distinct, random CSs. Initial values are determined randomly within the allowable range dictated by the problem type. These CSs are evaluated using the fitness function. Then, two binary tournament selections are used to select a first parent followed by a second parent. These two parents are then used to create a child via a crossover mechanism. Next, a Gaussian Mutation operator

is applied to the created CS with some probability μ. The mutation magnitude is modified by a coefficient δ. The child will always replace the worst CS in the population. This process is repeated until the maximum number of allowed function evaluations has expired. Since this algorithm has a larger population, it is less susceptible to early convergence and local minima (and larger populations provide more insurance against this problem).

Optimizing Machine Learners using Evolutionary Algorithms

As suggested in Section 2.3.1, traditional optimization techniques often face difficulties when dealing with complex search spaces. In [Maniezzo 1994], the author explores using Genetic Algorithms (GAs), a sub-type of EA, for the purpose of evolving the weights in a feed forward neural network, as well as topological features like the number of nodes and connections. In terms of classification, smaller networks with fewer connections prove to be more efficient. However, too few nodes and connections will produce classification error. Additionally, the system evolves some of the meta-information relating to the weights as well. Specifically, the granularity of the weights themselves is an evolved attribute. This represents a significant concern since more precision in weights is desirable from a classification perspective, but is undesirable from an optimization perspective since it increases the size of the search space. The author evolved neural networks for a wide variety of applications, including a test suite of theoretical problems (e.g. XOR) and for the control of a small robot. The EA demonstrated steady improvement in the neural networks for both sets of problems over time, as well as a reduction in the size of the network architecture.

Another approach to using EAs to optimize Neural Networks appears in [Han et al. 1996], where the authors take advantage of the strong search capabilities of genetic algorithms to search for good system parameters (the values of coefficients for the sigmoidal activation function used in the work) for Feed Forward Neural Networks. That work presents a hybrid approach, in that after a candidate solution is generated by the genetic algorithm, a more traditional backpropagation cycle begins to tune the network weights. The fitness in the genetic algorithm is defined by the number of training cycles necessary to reach a desired error tolerance. In other words, all accuracy features being equal, the algorithm that trains faster survives. In that work, the authors successfully test the methodology on the spiral problem, which typically proves very difficult for traditional optimization techniques.

EAs can also be applied to evolve the topology of RBFNs, as in [Billings and Zheng 1995]. Specifically, the authors sought to determine the number of hidden layer nodes and center locations (see Section 2.2.4) using a GA. The number of hidden layer nodes is encoded into the candidate solution, and the centers are chosen among the training data points. This approach is attractive given that solving for the number of hidden layer nodes leads to a discontinuous and non-differentiable objective function. Additionally, the means for evaluation can be formulated easily as a fitness function for a GA. In other words, to test a candidate solution, build the RBFN specified by the CS, and test it on a set of validation instances to classify. To solve for the weights, rather than evolving them (which can dramatically increase the search space for the GA), any deterministic method can be used (backpropagation, singular value decomposition, etc.). The authors test the methodology on predicting water level in a water system using roughly 500 data samples for

training and testing respectively. The best evolved RBFN shows a quite low generalization error of 0.002.

A somewhat different evolutionary approach to the RBFN optimization problem presented in [Billings and Zheng 1995] is the work presented in [Whitehead and Choate 1996]. Both strategies attempt to evolve the most accurate RBFN possible, the prior evolves a population of RBFNs with varying numbers of nodes and center locations and then employs deterministic means to solve for the weights. In [Whitehead and Choate 1996], the EA evolves a population of radial basis functions which collectively form a single RBFN. In other words, the candidate solutions cooperate with each other to form a single network to solve a problem, while at the same time competing with each other to eliminate redundant or ineffective nodes. Candidate solutions consist of encoded centers and widths. To determine survival, the RBFN is tested on a potentially random set of interesting test instances and the test error is apportioned to the candidate solutions. Once the topology is determined, the weights themselves are solved for using the Least Mean Squares algorithm [Widrow and Hoff 1960]. This cooperative-competitive approach outperforms traditional clustering approaches for topology when applied to time series prediction.

The authors in [Leung et al. 2003] present another cooperative approach in which the population of candidate solutions comprise a neural network. In this case, the candidate solutions represent nodes, switches, and weights in a fully connected three-layer feed-forward neural network architecture. The switches allow any given link to be turned off, which effectively allows the topology to be dynamically changed during evolution. If all of the switches corresponding to a single node are turned off, the node is effectively removed. Both the incoming and outgoing weights for all the nodes are also evolved in this strategy,

eliminating the need for an additional optimization algorithm for weight optimization. By manipulating the switches, the architecture can be reduced or expanded (up to a maximum of the initial size). However, discovering a good initial size is achieved through incremental expansion over successive runs. The authors likely avoided evolving this parameter in an effort to reduce the complexity of the search problem. The authors applied this approach for the purpose of forecasting sunspots, for which they produced impressively accurate predictions over the recorded history for which sunspot data was available.

EAs have also been applied to reduce the number of features used as inputs into MLs. One straightforward approach is to treat feature selection like a search problem, encoding all potential attributes in a binary string (or "wrapper") [Kohavi and John 1997]. In this case, zero bits indicate the exclusion of a given feature and one bits indicate the inclusion of the given feature. The candidate solutions can be evolved using an EA, deterministic hill-climbing, random walks, etc. As in the previous examples, fitness can be assigned by testing the neural network using the given features and calculating the error.

2.4 Chapter Conclusion

In this chapter, a general introduction into Machine Learning and Optimization was given, with a focus on algorithms that are the current state of the art. Algorithm descriptions were provided for GRNNs, RBFNs, and SVMs. Additionally, descriptions were provided for the Evolutionary Hill-Climber and Steady-State Genetic Algorithm EAs. A survey was provided of past efforts evolving Machine Learners using Evolutionary Algorithms. This chapter provides the theoretical basis for the initial modeling of the canine operator presented in Chapter 4.

Chapter 3

System Architectures

3.1 Chapter Introduction

This chapter provides a high-level description of the interacting components which contribute toward the Autonomous Canine. Three architectures are presented, which describe existing systems designed for the purpose of testing the various sensors, gathering data for the development of filtering models, and supporting the autonomous guidance of the canine.

3.2 Hardware Elements Used

Some of the hardware elements used in the various phases of the project are similar. Thus, they are discussed here and referred to throughout this manuscript. The primary purpose of this chapter is to provide a collected, high-level view of the various systems for the purpose of comparison. More detailed descriptions of the hardware and software used in each phase are provided in the individual chapters.

3.2.1 Rabbit Microprocessor Core Modules

These small embedded computer system are compact, use relatively low power, and have several serial connections in order to interface with a variety of sensors and communications devices. The focus on small, low-power systems results from the need to have a system that can ride on the dog's harness comfortably.

3.2.2 GPS Receivers and Antenna

A discussion on the workings of the Global Position Satellite (GPS) system is outside the scope and purpose of this document. In this work, GPS provides latitude, longitude, velocity, course, and elevation information. However, GPS sensors generally produce this information with a relatively low frequency compared to other sensors that do not depend on receiving communications from satellites. This motivates the use of additional sensors for higher-frequency updates of information and for times when GPS cannot be accessed due to physical conditions (for example, moving indoors or into a forest). These measurements are provided via serial connection to the Rabbit core. All three phases of the project discussed below make use of GPS.

3.2.3 Inertial Measurement Unit

Inertial Measurement Units (IMUs) contains accelerometers and gyroscopes, which provide high frequency measurements of acceleration and yaw rate, respectively. However, they do not provide information on position. Aggregating the data from GPS and the IMU using an Extended Kalman Filter can improve position estimates [Miller and Bevly 2007].

IMU measurements are provided via serial connection to the Rabbit core. All three phases of the project discussed are capable of using the IMUs, although the autonomous control algorithm does not actually use this data at present.

3.2.4 Magnetometers

Magnetometers measure magnetic fields surrounding them. In the absence of a nearby magnetic presence, they can provide an indication of heading by measuring the magnetic

field of the earth. This is an additional source of data for potential aggregation. Magnetometers are available in Training Phase II.

3.2.5 Xbee Radio Modem

This small, power-efficient serial radio modem communicates at a 38400 baud rate. It is used for communication with an external data sink (a laptop) since the Rabbit has limited data storage capacity to be used for experimental data. The Xbee is used by all three project phases.

3.2.6 Command Module

The command module is simply a piece of hardware capable of issuing the commands that the canine recognizes. In training phase I, the command module simply issued a radio tone. In Training Phase II and the Autonomous Canine Phase, the command module provided a variety of tones and left and right vibrations.

3.2.7 Data Sink

The Data Sink is a Windows-based laptop system which collects sensor data from the Xbee radio modem through a second Xbee connected to the laptop. Particularly during Training Phase I and II, this provides an outlet for large amounts of data storage to use for training data and to evaluate filtering algorithms. In the autonomous canine phase, the laptop executes the control algorithm and communicates those commands back to the embedded device.

3.3 Training Phase I

The training phase I system was developed by Ben Clark and the GPS and Vehicle Dynamics Laboratory (GAVLAB) at Auburn University prior to the efforts of this dissertation. It is discussed here briefly because this system setup provided the data used in Chapter 4. A graphical representation of the system appears in Figure 3.1, and a discussion follows.

In the initial training phase, the human trainer observes the dog to determine if he is "on course" or "off course" and issues a tone (which has been previously established as a conditioned response for the canine) if the dog is traveling in the correct direction. He uses an analog handset which directly transmits the sound to the dog; it operates independently of the sensor pack and the Rabbit core. Simultaneously, the sensor pack collects GPS and IMU data and transmits it continuously through the Xbee modem to the Data Sink. In order to coordinate the commands being issued by the trainer with the data being produced by the sensors, the computer operator had to hold down pre-assigned command keys to log that information in data files. This caused a lag between the human issuing commands and the computer operator actually logging those commands.

3.4 Training Phase II

From the perspective of attempting to understand the canine's position, Training Phase I was promising. From a modeling perspective, the initial data was useful and allowed the training of some models. However, to develop a more robust and general model, additional

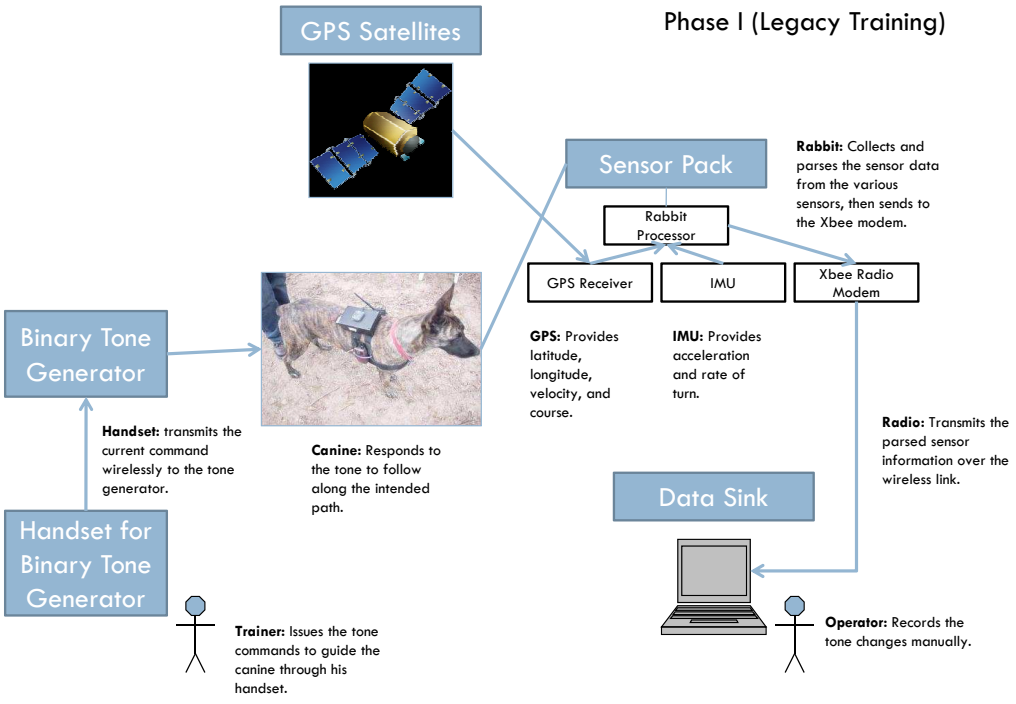


Figure 3.1: High Level Operational View of Phase I Training

data was needed, as well as a more advanced command set including both tones and vibration. The Training Phase II system sought to address the following needs not met by the Training Phase I system:

1. Support for a wide variety of tones and vibration commands to be sent to the canine, rather than a single binary tone. Specifically, this added command concepts like move left, move right, and stop.

2. Faster underlying hardware taking into account the advancements in embedded systems in the years since Training Phase I.

3. A wider array of sensors, specifically the inclusion of an Xsens sensor including accelerometers, gyroscopes, and magnetometers.

4. Improvements in battery life and packaging.

5. Improvements in the GPS antenna used.

6. Automated recording of the commands issued by the trainer, as opposed to manual recording by a human computer operator.

To these ends, the Training Phase II system was developed (for a more detailed discussion of the Phase II system, refer to Section 5.3. As shown by the system architecture in Figure 3.2, the human trainer issues commands through a programmable handset which are transmitted to the command module which is located on the dog. Different tones represent different specific commands, such as "move left" or "stop." This differs from Training Phase I in which tone commands only indicated "on course" or "off course." As the command module receives new commands, it sends a message to the Rabbit, which stores the

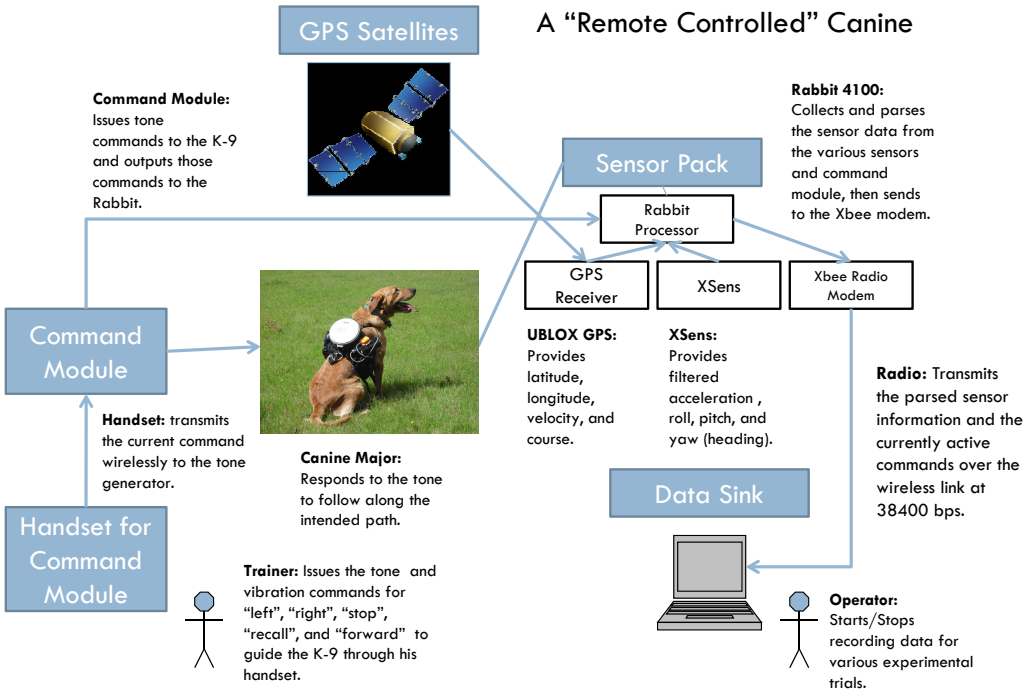


Figure 3.2: High Level Operational View of Phase II Training

currently active command. Concurrently, the sensors (GPS and XSens) are producing messages which are collected by the Rabbit. The XSens operates at a higher frequency than GPS, so whenever new IMU data is available, a message is sent via the Xbee modem to the Data Sink. In practical terms, this means that at any given time, the XSens data will always be new, whereas the GPS data will be the last GPS data received (which often will not have changed since GPS produces data more slowly in our system). Each message sent by the Xbee modem includes the current command.

3.5 Autonomous Canine Phase

In the Autonomous Canine phase, a working prototype for autonomously commanding the canine has been built and demonstrated. This required a few modifications to the system (illustrated in Figure 3.3 and enumerated below):

1. A control algorithm operating on the laptop replaces the need for human guidance of the canine.

2. The XSens was removed from the sensor pack because real-time filtering algorithms were not available to make use of it. The hardware still has the capability to support the XSens, so that if such algorithms become available, they can be used.

3. Two-way communication between the laptop and the embedded system became necessary in order to perform the control algorithm computation on the laptop and report those commands back to the embedded system.

3.6 Chapter Conclusion

This chapter provided a high-level description of the various sensors and hardware used for the canine system. These descriptions are provided to give a basic understanding of the utility of this hardware, rather than to attempt to fully describe the technical specifications of each individual component. Additionally, this chapter describes the motivation and evolution of the various design phases of the project at an architectural level. Future chapters will reference which architecture were utilized and provide a greater level of technical detail on specific sensors.

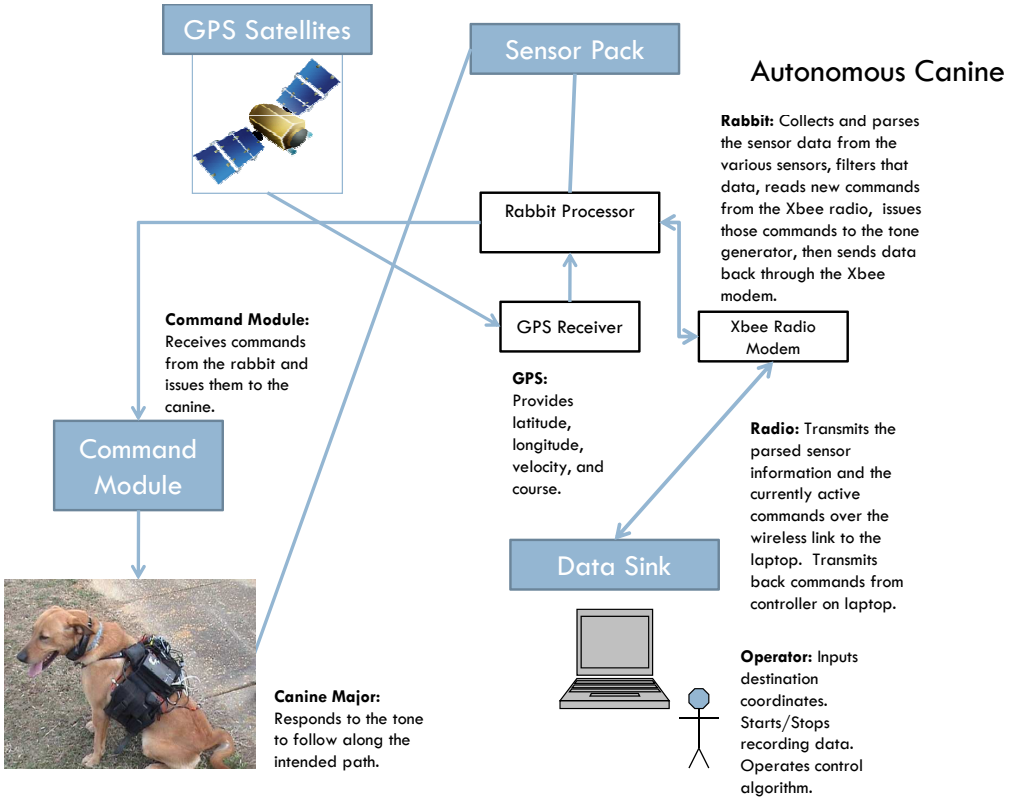


Figure 3.3: High Level Operational View of Autonomous Canine Phase

Chapter 4

Automated Modeling of the Guidance of a K-9 with a Binary Tone

4.1 Chapter Introduction

This chapter attempts to automate and replace human guidance in the control of a canine unit by modeling that guidance from observations. The ultimate research goal seeks to contribute towards the autonomous command of a trained canine unit by analyzing the movement and the behavior of the dog as it responds to command tones. Specifically, GPS and a binary command signal (from a human trainer) is recorded as a canine follows instructions to moves toward a destination. The data is then processed into training instances and used as training data for a General Regression Neural Network (GRNN). Then, the network is used to classify previously unseen test instances to determine if the behavior at that moment is normal (on-course) or anomalous (off-course and in need of correcting tones). Both representation of training instances and the system parameters of the GRNN are optimized using a simple Evolutionary Hill-Climber (EHC). Given even fairly limited initial data for training, the system performs well, producing relatively few false positives and false negatives in classification.

The purpose of these particular experiments is to provide some empirical results to demonstrate that the human guidance of the canine can be directly modeled using ML techniques. In addition, this modeling was performed with a relatively low cost range of sensors. A condensed version of this chapter appears in [Britt et al. 2008].

4.2 Data Collection

Experimental studies through the Auburn University Canine and Detection Research Institute have already been performed on the ability of a human to command the canine to specific locations. In these experiments, a single tone was issued over a radio to inform the dog that the direction being pursued was correct (normal behavior heading toward a pre-determined goal). Once the dog made a wrong turn, the tone would be deactivated (hereafter referred to as "anomalous behavior"), and the dog would begin to pursue other directions until the tone was reactivated. When asked to make one direction change during the course (either to turn left or to turn right), the canine made the correct decision 76 out of 79 (96.2%) times . When commanded to make multiple direction changes (in one path) while following the static tone to a designated location and to then return to home base also following the static tone, the canine correctly completed 189 out of 206 (91.7%) total trials (one trial consisted of multiple direction options and direction changes). The results from this current work have revealed that dogs can be trained to obey audible direction-oriented commands relayed wirelessly over long distances consistently and reliably for relatively lengthy periods of time with no human contact or intervention.

In later experiments performed by the Auburn University GPS and Vehicle Lab (GAVLAB) using the Training Phase I architecture (see Section 3.3), a GPS/INS radio on the canine unit (shown in Figure 4.1) wirelessly communicated to a laptop acceleration, velocity, latitude, longitude, and course data. This information was recorded as the dog was guided by a human trainer from a start position to a known goal. A corresponding tone ("on" for normal or "off" for anomaly) that was being produced by the trainer was recorded alongside the GPS/INS data. An example trial is shown in Figure 4.2, where the x-marks represent

Figure 4.1: Remotely Commanded Canine with Radio and GPS/INS Receiver

the dog going off path and the tone being turned off. This was repeated for a total of seven trials (each trial represents a single path from start to destination). The data collected from these trials was collected into text files which were processed for the purposes of training data for machine learners. The ultimate goal of the algorithm is modeling the human trainer in order to determine whether or not a canine's behavior is normal (in need of no change in tone) or anomalous (needs a change in tone).

The GAVLAB has performed additional work with canine units, GPS, Inertial Measurement Units (IMU), and Extended Kalman Filtering [Stengel 1994] in order to make the GPS measurements reported more precise [Miller and Bevly 2007]. The authors demonstrate that combining the sensor measurements from GPS and IMUs provides the capacity to not only have more precise measurement, but also to be able to produce data when GPS becomes temporarily unavailable (for example, due to blockage).

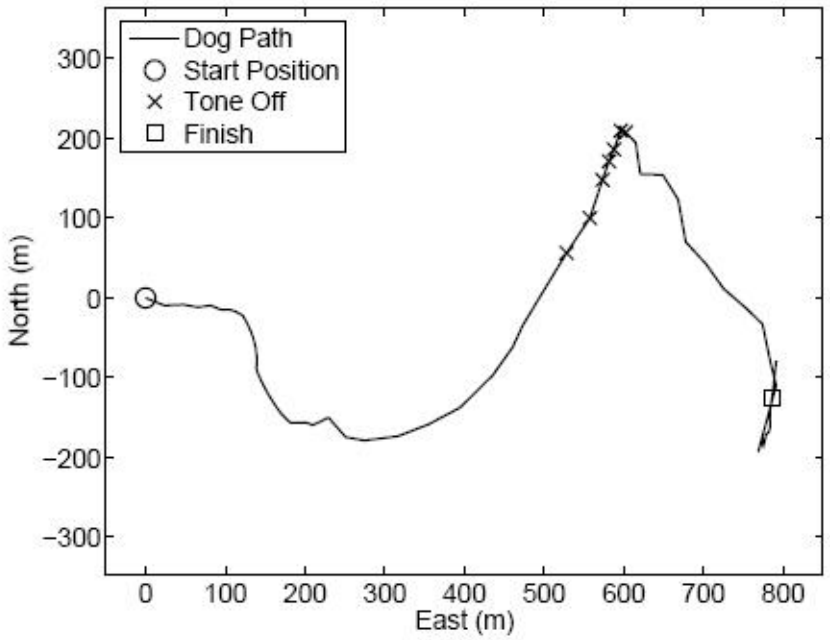


Figure 4.2: Example of a canine path and command tones

4.3 Classifying Using a General Regression Neural Network

For this initial work, the GRNN (as discussed in Section 2.2.2) was utilized to perform classification. This work attempts to carefully address each of the key factors discussed with respect to the performance of ML algorithms. Learnability is believed to be present given that our devices attempt to capture the same information that a human uses to guide the canine visually. While limited, the current training data consists of a number of instances of both normal and anomalous behavior. Further, since the data comes from actual canines attempting to follow paths, the data seems likely to be representative of the problem. The system parameters (including which attributes to use and the system parameter for the GRNN) are optimized using a simple evolutionary method (described in Section 4.4) in a fashion that would be done off-line in a real system. In other words, the system parameters

would be optimized prior to deployment to avoid the computational cost of running an optimization routine on an embedded system. However, once the algorithm is optimized, the model can be applied more quickly.

4.4 The Evolutionary Hill-Climber Algorithm

In order to discover an appropriate value of the GRNN system parameter standard deviation σ and to determine which attributes from the training instances should be included (in other words, what elements comprise the vector X in Section 2.2.2), a simple Evolutionary Hill-Climber was utilized. This section describe the specifics of the algorithm used, as opposed to the general description of EAs provided previously.

4.4.1 Algorithm Description

Traditional deterministic Hill-Climbing methodologies tend to inadequately deal with complex search spaces and get stuck at local minima [Golberg 1989]. Instead, stochastic methods of search often prove preferable in instances where there exists no guarantee of smoothness or of singular maxima in the function to be optimized [Russell and Norvig 2003]. In optimizing σ and the attributes for the GRNN, there are no such guarantees. The Evolutionary Hill-Climber is outlined in Algorithm 1, based loosely on the Random-Mutation Hill-Climber described in [Forrest and Mitchell 1993]. The EHC first randomly creates a single candidate solution consisting of a chromosome with a number of values corresponding to parameters being optimized (in this case, the σ value and whether or not to include each of the thirteen possible attributes described later). Initial values are chosen from within the allowable range dictated by the problem type (in this case, the range of

values for the σ was chosen from the range (0,1.0] and each of the attributes could either be considered "included" or "not included"). This candidate solution is then evaluated by a fitness function and the fitness is assigned to that individual.

Algorithm 1 Evolutionary Hill Climber

```
1: t ⇐ 0
2: Initialize candidate solution i with random initial values i_0..i_N = rand(lb, ub)
3: i.fitness ⇐ i.evaluate
4: best.fitness ⇐ i.fitness
5: while i.fitness < desired && t < MAXITERATIONS do
6:    Create a new candidate solution c ⇐ i
7:    Randomly choose one value in the candidate solution r = rand(1, N)
8:    Mutate c_r
9:    c.fitness ⇐ c.evaluate
10:   if c.fitness > best.fitness then
11:      best.fitness ⇐ c.fitness
12:      i ⇐ c
13:   end if
14: end while
```

On each iteration, a single Uniform Mutation (taken from the range [-1,1]) is added to the σ value, multiplied by the mutation amount δ (chosen to be 0.25 for these experiments) and the starting value of the gene itself. There is a small chance (0.15, for each attribute) that an attribute bit will be reversed (effectively deleting or adding an attribute from consideration). This new candidate solution is then evaluated. If the new CS has a better fitness than the previous, then the previous CS is replaced. However, if the new candidate solution has a worse fitness than the previous, then it is rejected.

The overhead associated with the EHC is quite low on each iteration, requiring only a handful of elementary operations. The primary motivation for adding or deleting attributes is to generate a minimal set of attributes which will provide the most accurate results.

Having fewer attributes has the added benefit of reducing the complexity of the GRNN each time it is used for classification.

4.4.2 Fitness Function

To evaluate the fitness of a candidate solution, the GRNN is run with the indicated attributes and value of σ on test instances that were *not* used as training data. The fitness formula is given by:

$$fitness = C - (3 * f_n + f_p) \tag{4.1}$$

In (4.1), C is the number of instances correctly classified by the GRNN, f_n is the number of false negatives (anomalies incorrectly identified as normal behavior), and f_p is the number of false positives (normal behaviors incorrectly identified as anomalies). The fitness function was biased against false negatives to promote networks that more effectively identified anomalous behavior in the canine. In order to break ties in cases where the success rates for two candidate solutions were equal, the average distance from the desired output to the resultant output over all the test cases was used.

4.5 Representation of Instances

What information and how that information is represented affects the ability of the system to correctly classify behaviors. This section discusses the raw data from the device on the canine and the processing done to that data to make it most useful to the GRNN.

4.5.1 Raw Data

Many times a second (60 Hz for IMU data and 4 Hz for GPS data), the mobile system reports the measurements shown in Table 4.1:

Name	Units
ax, ay, az	(g)
gx, gy, gz	(deg/s)
velocity	(m/s)
course	(deg)
latitude, longitude	(deg)
signal	unitless

Table 4.1: Raw Measurements

In all, there were seven measured trials which produced between 200 and 350 training instances (differing trial paths varied in length). Of those seven, two contained no anomalous behavior while each of the other five contained some anomaly which caused a tone to be changed while the dog found the new path.

4.5.2 Data Processing

To produce training instances, the raw data was processed into a series of derived metrics. The transformations were made using only data available at the point of the measurement (no future knowledge) and were generally differences between two raw data entries to illustrate potential anomalies through unusually large or small changes. For example, very large changes in velocity between readings might be significant. Additionally, three derived metrics were utilized. The first is the Distance (D) in degrees between the current latitude and longitude and the coordinates of the destination (the value could be converted to meters, but since the data will ultimately be normalized this conversion would make little difference and require more computation). The second is the Readings Since

Improvement (RSI), which is the number of readings that have passed since the Distance has decreased, which provides some indication of whether or not the dog is making progress toward the known destination. Finally, the Deviation from Desired Course (DEV) in degrees shows the difference between the current course and the ideal course which would lead the dog to the goal (calculated using the current coordinates and the destination coordinates). A summary of all 13 metrics is provided in Table 4.2, where the Δ symbol indicates the attribute is the absolute value of the difference between the attribute in the current raw data instance and the attribute in the previous raw data instance. All of the metrics were normalized between [0,1.0] (using the highest known values of a given attribute as the maximum) for input into the GRNN.

Name	Units
Δax, Δay, Δaz	g
Δgx, Δgy, Δgz	(deg/s)
Δvelocity	(m/s)
Δcourse	(deg)
Δlatitude, Δlongitude	(deg)
D	(deg)
RSI	unitless
DEV	(deg)

Table 4.2: Processed Metrics

4.6 Experiments

To evaluate the classifier system, the data from the seven trials was divided into two groups: a training set and a larger testing set. The training set consisted of the results of a single trial (340 processed instances, which included some anomalous behavior) and the testing set consisted of the remaining six trials (roughly 250 processed instances each, some trials did not contain anomalies). Two different experiment types were run: one

using trial-specific parameters and the other using general parameters optimized over all the trials.

4.6.1 Trial-Specific Parameters

In these experiments, the EHC was run to optimize the attributes and parameters for specific trials. In other words, parameters were discovered for each of the six trials in order to minimize classification error (as described in Section 4.4.2). Optimizing over a specific path has the benefit of providing a nice improvement in accuracy, as would be expected from tailoring the parameters. However, it has the disadvantage of losing generality. In other words, using path-specific parameters on a different path will generally perform badly. In many applications, however, path-specific parameters would be the ideal choice. For example, if a canine were to be used to routinely check a path around an airport for drugs, then path-specific parameters would be preferable since generally the goal would be to minimize error even at the cost of generality.

In order to optimize the GRNN parameters, the EHC was given 500 cycles. In Table 4.3, the results of the optimization, alongside the results of running the optimized GRNNs are shown. The "Trial" column indicates which path was used as the test set (there is no other significance to the trial number). An asterisk (*) indicates that the given trial did not contain anomalies (hence, none should be detected). The σ column indicates what value of σ (calculated by the EHC) passed to the GRNN yielded the results shown. The "attributes" column shows which attributes were included in the GRNN calculations. Columns "total" and "correct" indicate the total number of instances and the raw number that were classified correctly. The value of "Success Rate," "FN," and "FP" indicate the percentage of correct

classifications, the number of false negatives, and the number of false positives, respectively. The value of "fit" is given by (4.1).

Trial	σ	attributes	total	correct	SR	FN	FP	fit
2	0.1488	Δax,Δgy,Δgz,Δlat.,Δlong.,ED,RSI	250	228	0.912	1	21	204
3	0.0379	Δvelocity, Δlat.,Δlong.,D	215	183	0.851	13	19	125
4*	0.0262	Δcourse, Δlong.,D,RSI,DEV	204	204	1.0	0	0	204
5	0.1091	Δlong.,D,RSI,DEV	284	246	0.866	17	21	174
6*	1.0	Δlong.,D,RSI,DEV	322	322	1.0	0	0	322
7	0.0970	Δax, Δaz, Δgy,Δgz,Δvelocity,Δlong.,RSI,DEV	230	210	0.913	9	11	172

Table 4.3: Path-Specific Parameter Results

In general, the method gives fairly good results given the limited amount of training data available. Even in the worst cases (Trials 3 and 5), the accuracy was still over 0.85 with relatively few false positives and false negatives - in both those trials, enough anomalous instances were identified such that a control system could recognize it as something other than an outlier. In Trials 2 and 7 (both with anomalies), the accuracy improved even more, exceeding 0.90, and there were very few false negatives. An example (Trial 7) is shown in Figure 4.3, where the thick line indicates the path that the dog actually traveled along, the small crosses indicate the points where the human trainer ended the tone (indicating the dog needed to correct itself), and the small x's indicate where the GRNN indicated that the tone should be dropped. The GRNN makes only a few relatively isolated errors near the end of the dog's path, which could likely be resolved with increased training data or by only changing the tone given a certain number of anomalies detected. In the remaining trials (which contained only normal behavior), no false positives were detected at all.

4.6.2 General Parameters

In these experiments, the EHC was run to optimize one set of attributes and parameters for all of the (non-training) trials together. Parameters were discovered only once to give

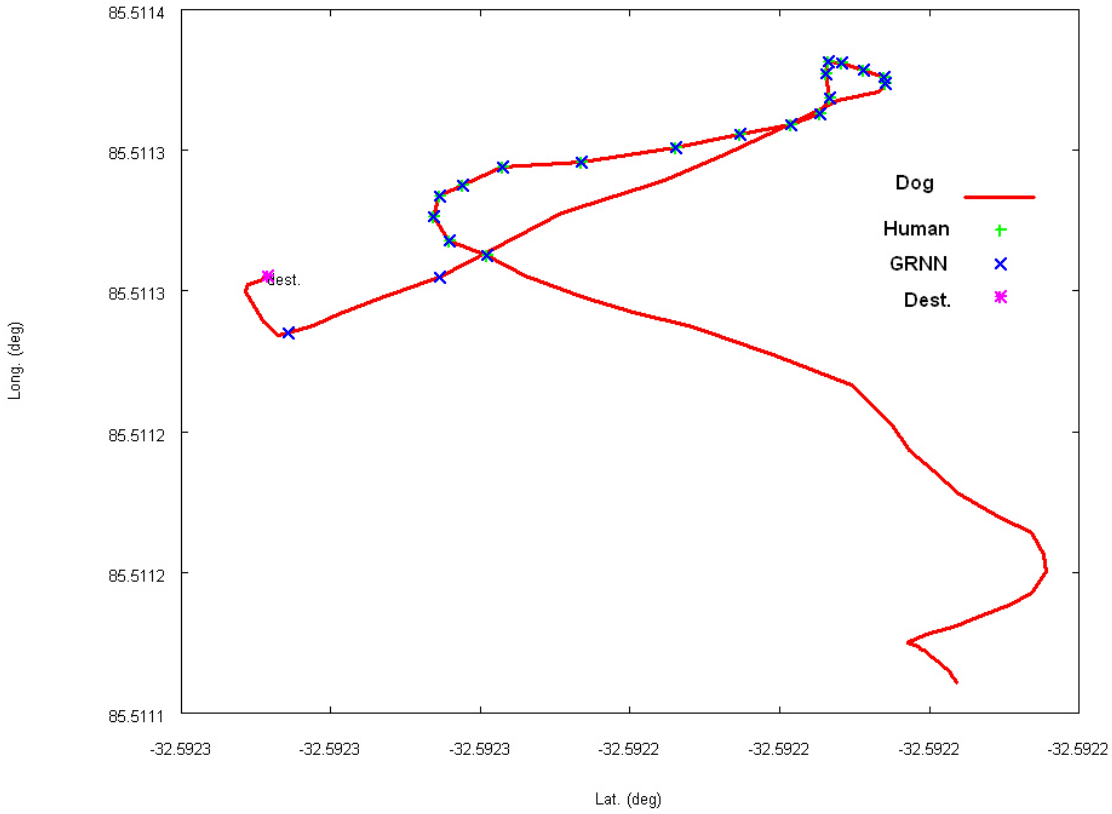


Figure 4.3: Sample Trial with GRNN Output

the best average results over all trials. The benefit of choosing general parameters is that they generate better anomaly detection over a wider range of paths. However, the best general parameters will often perform worse on any individual path. General parameters would be preferable in a situation where either the canine must deal with a wide variety of paths, the path is unknown in some way (such as an area with vehicles), or simply the path itself is unseen, other than the coordinates of the destination.

In Table 4.4, the results are broken down by trial for comparison purposes. The column headings have the same meaning as above, although it should be noted that the σ and attributes are the same for all trials. While performance is still fairly good, there is a roughly 0.05 drop in SR across nearly all of the trials. There is also a moderate increase in both false positives and false negatives. It is worth noting that the best performing settings used less than half of all of the available attributes, indicating that the classification is largely dependent on the information given by a few derived metrics.

Trial	σ	attributes	total	correct	SR	FN	FP	fit
2	0.1234	Δaz,Δlong.,D,RSI,DEV	250	215	0.860	4	31	172
3	0.1234	Δaz,Δlong.,D,RSI,DEV	215	178	0.818	23	16	91
4*	0.1234	Δaz,Δlong.,D,RSI,DEV	204	197	0.966	0	7	190
5	0.1234	Δaz,Δlong.,D,RSI,DEV	284	230	0.810	27	27	122
6*	0.1234	Δaz,Δlong.,D,RSI,DEV	322	322	1.0	0	0	322
7	0.1234	Δaz,Δlong.,D,RSI,DEV	230	205	0.891	16	9	148

Table 4.4: General Path Parameter Results

4.7 Discussion

These results provide indication that the concept of an "anomaly" (as defined by the tones given by a human trainer) can be detected with a relatively high degree of accuracy. Most likely as more trials are gathered, the training data can be refined and expanded, allowing for even greater accuracy for a larger variety of paths. Further, it should be noted

that in practice a single instance being classified as "anomaly" should not necessarily result in an immediate change of stimulus to the canine. Rather, several "anomaly" classifications in a short period of time should be used to indicate a change in control signals. This would ameliorate the impact of scattered false positives and false negatives in practice. Another important observation is that even a human trainer is not completely accurate in identifying that the canine is not behaving correctly, so some error is inherent in the system.

With respect to performance, a single classification operates in less than a second, even when file I/O is required. In an actual implementation, many classifications could be performed in a second. The optimization routine takes considerably longer (on the order of minutes) but should be performed in an off-line fashion to discover parameters once and then reuse them.

4.8 Chapter Conclusion

In this phase, a method for classifying canine behavior as either normal or anomalous was presented. A General Regression Neural Network was used on a set of training data taken using a GPS/INS unit on-board a dog being directed by a human trainer and then used to predict the canine behavior in future trials. An optimization method, the Evolutionary Hill-Climber, was discussed as a means to determine which attributes to include in the training and to determine good system parameters for the GRNN. Both path-specific and general settings were presented, and their results on several test sets were shown. In general, path-specific results were better, but in many applications general results might be necessary. Overall, the results prove to be quite promising, but more training data would be required to know if the approach is viable in practice.

Chapter 5

The Command and Navigation of a Trained Canine

5.1 Chapter Introduction

This chapter demonstrates the capability to use a developed embedded sensor suite housed in a vest to consistently track the position, motion, and orientation of a canine. This ability to quantify and record canine position and motion in real time extends the range at which a canine handler could control a canine; thus, the system could improve canine team capabilities to search areas for narcotics or explosive devices when they are out of immediate line of sight. Within the immediate context of the dissertation, this provides a platform for gathering data to monitor the human trainer's interactions with the canine and ultimately provides a platform upon which to build the autonomous canine system. In the context of canine training, the ability to record trials provides immediate benefits in terms of allowing for sophisticated post-trial analysis and quantification of canine behaviors such as behavior associated with detection. A detailed description of the sensor equipment is provided, including the Global Position Satellite (GPS) receiver and antenna, accelerometers, gyroscopes, and magnetometers. The chapter describes the current accuracy of the sensor suite and proposes that a multi-sensor system is superior to a GPS-only approach for a number of applications. Specifically, the additional sensors provide for higher frequency readings, a tolerance to GPS loss, and the ability to characterize canine orientation. The system further supports the actuation of tone and vibration commands using a tone and vibration generator; the system records and wirelessly reports these commands in real time alongside sensor data to allow for on-line or off-line analysis of canine trials using either raw

sensor data or visualizations. To illustrate the capabilities of the command and tracking system, an extensive case study is performed in the remote command of a trained canine by a professional canine trainer during repeated field trials. The case study demonstrated the ability to consistently track the canines position, track canine orientation, consistently receive data in real time, and ultimately visualize that data to analyze canine behavior in terms of mission success or failure.

The Phase II system as it stands allows for several immediate applications:

- Canine trainers can gain sensor information from a variety of sensors about the dog's location, movements, and orientation in real time.
- Canine trainers can record trials with canines which show the commands of the canine coupled with sensor data, allowing them to effectively observe the dog's behavioral responses to given commands in given circumstances. Further, these trials can be shared and analyzed in a quantifiable way (data files).
- Canine trial data can be easily translated into more readily understandable forms, such as position plots with command overlays or GPS plots using freely available software such as Google Earth ©.
- The system could be used in an operational setting to guide a canine based on the real-time sensor data, even when out of line of sight, as long as radio connectivity is maintained. This would expand the immediate operational range of the canine units.
- As is demonstrated in the case study, it is feasible for a trained canine to be reliably guided in a "remote control" fashion using tones and vibration commands.

- The system has provided data which have enabled the adaptation of sensor aggregation techniques to improve sensor data and to compensate for potential GPS losses.

In the context of the dissertation, this research contributes a platform to be used in conjunction with the development of control algorithms which utilize information available from sensors on-board the canine to provide audio and vibration command and control signals for the purpose of autonomously (no human in the loop) directing the dog to waypoints (i.e. specific locations).

The possible uses for this type of platform and capability in both homeland security and canine training are numerous. For example, routine searches of ports by canines could be improved by extending the range and relative autonomy of canines. Dog trainers would have increased ability to analyze trial results and more readily share information about training techniques because they could show the exact time that commands were given and show the canine response. In explosive-detection scenarios, canines could be more accurately guided to locations keeping both the canine and people safer. It has been shown in [Ferworn et al. 2007] and [Ribeiro et al. 2008] that canine augmentation systems using accelerometers and wireless radios could be beneficial for search and rescue teams by giving indication of canine pose in a variety of environments. Our sensor suite could support these kinds of activities in addition to providing tracking and command information.

In the veterinary community, this platform could provide additional utility (in terms of more robust and descriptive sensor measurements) for research efforts which attempt to characterize animal behavior by analyzing their movements. Researchers have successfully used accelerometers to characterize the motion behavior of cats during their daily routine [Watanabe et al. 2005], the activity types of sows [Cornou and Lundbye-Christensen 2008],

and goats during grazing [Moreau et al. 2009]. In veterinary sports medicine, having a high quality means to measure and analyze canine gait analysis provides direct benefits to clinicians [Gillette and Angle 2008]. Additional research efforts have examined tracking and virtual fencing of cows using GPS to both contain herds and to prevent overgrazing [Schwager et al. 2008, Correll et al. 2008].

The remainder of this chapter is organized as follows. In Section 5.2, the canine, his training, and the relevant ethical approvals are described in detail. Section 5.3 presents the hardware and software systems used to track and command. Section 5.4 describes the experimental setup used to validate the sensor pack on a real canine and Section 5.5 describes the results of field trials on the canine. Some concluding remarks are provided in Section 5.6.

5.2 The Trained Canine

5.2.1 The Trained Canine

The dog (shown in Figure 5.1) that participated in these experiments was a male Labrador Retriever named "Major" that was approximately 4-years old and weighed 32 kg. Prior to his acquisition, Major had undergone the initial stages of traditional field/hunt trial training such that he was capable of reliably executing basic "blind retrieves" (i.e., without first seeing the throwing of an object and without obvious visual cues, being directed by handler in the direction of that object through a series of voice, visual, and whistle commands to the object, which the dog then retrieves) of approximately 100 meters.

Figure 5.1: The trained canine, Major, with the sensor pack.

5.2.2 Ethical Approval

The use of the canine in this experiment and other ongoing canine detection technology development efforts was approved and monitored by the Auburn University Institutional Animal Care and Use Committee (IACUC), which ensures compliance with the Animal Welfare Act (7 USC, 2131-2156). Auburn's IACUC is approved by the Office of Laboratory Animal Welfare of the U. S. Public Health Service and Auburn's animal housing and care is inspected annually and has been approved by the Animal Welfare Division of the Animal and Plant Health Inspection Service of the U. S. Department of Agriculture. The canine was housed in IACUC inspected kennels at Auburn University controlled property at Fort McClellan, AL and his care and use were conducted by Auburn University personnel who had successfully completed training in the care and use of dogs used in research or training activities. In addition to the internal IACUC surveillance, the activities in which Major has participated, his housing and care, and the qualifications of those performing research and training activities with Major were reviewed and approved by the Animal Care and Use Review Office (ACURO) of the U. S. Army Medical and Materiel Command (USAMRMC).

5.2.3 Training Procedures

Section 5.2.3 on training procedures is quoted verbatim with expressed permission from the author [Waggoner 2009]. It is provided to give the reader understanding of Major's background and training prior to and during this project.

Equipment

A custom made harness (Blackthorn K9 Equipment, Inc.) upon which was mounted a prototype K9 Remote Sensor System was employed in training canine Major. The Remote Sensor System was designed to provide first responders with a remotely controlled explosive detection capability that could also serve as a platform for a variety of other sensors (e.g., video, audio, radiological, air quality, and physiological monitoring of the dog), the output of which can be transmitted to responders in real-time. For the purpose of the present experiments, the important function of the Remote Sensor System was in delivering radio controlled auditory and vibration signals to canine Major. The canine command functions were controlled by a tone generator worn by the dog and actuated by a pair of Motorola EX600 radios, one worn by the dog and the other controlled by the handler of the dog. The key pad of the EX600 radios served as the handler-dog interface and pressing different keys of the radio resulted in the presentation of different tones from a speaker on the harness driven by the tone generator or actuation of one of two vibrators mounted on the underside of the harness. The commands used included: "Forward" (leave the handler and go in straight line from initial physical orientation), actuated by a distinct tone; "Stop" (dog sits), actuated by a distinct tone; "Over Left" (dog goes left in relation to its present orientation), actuated by vibrator on left rear of the harness; "Over Right" (dog goes right in relation to its present orientation), actuated by a vibrator on right front shoulder of the harness and; "Recall" (dog returns to starting point or current position of handler), actuated by a

distinct tone. An electronic collar (Dogtra, Inc.) was used in initial training to establish and apply negative reinforcement in the form of brief (less than 2s) mild (30-80 mA) electrical stimulation and vibration (such that the vibration was established as a conditioned negative reinforcer that could intermittently function effectively without presentation of the electrical stimulation) for the emission of correct responses signaled by voice, whistle, visual, and, later, tone and vibration commands.

Odor Detection Training

Canine Major was initially trained to "sit" in the presence of odor from C-4 explosive material by prompting him to place his nose near a box containing C-4 explosive (target) and issuing the verbal command "sit", which he already reliably emitted. Sitting at the target box resulted in delivery and opportunity to chase and play with a tennis ball or other toy. This activity was repeated until Major placed his nose near the opening of the box and sat in the absence of the verbal command. Next, a second identical box was added to the training scenario that did not contain C-4 (i.e., blank) and Major was prompted to sample from both boxes, the order of which was randomly alternated. With two boxes, one blank and one with the target, sitting at the target containing box was reinforced and Major was encouraged to NOT sit at the blank box by the verbal command "No", which was previously established as a conditioned punisher by association with mild leash corrections and interruption of opportunities for reinforcement in obedience type training, and mild leash pressure to not allow

him to complete the sit response. Once Major reliably discriminated between the target and blank box (i.e., approximately 20 contiguous errorless trials), a 3rd and subsequently 4th blank box was added to the scenario. Once Major was reliably sitting at the target box and NOT sitting at any blank boxes in this, so called, "4-hole variable" in typical detector dog training, training to search for and alert to the target odor in varied environments was initiated.

Initially, the target material was placed in relatively easy to detect locations taking relatively a short time to encounter from the point at which Major was prompted to begin searching for the target odor. Again, detection of the target, was reinforced with delivery and opportunity to play with a ball or other toy and sitting at other locations was either interrupted by the verbal command "No" or resulted in the command "No" and discontinuation of the particular trials and thus, opportunity to receive the ball for responding correctly. The locations of the target "hides" were gradually made deeper, higher, or otherwise more difficult for detection of the target and the distance, and thus amount of time searching required, was extended.

Finally, target odor detection training was incorporated in the context of Major's pre-existing field trial training. Initially, trials would be alternated between blind retrieves in which Major would be directed to an area where there was a bumper to retrieve and trials in which he was directed to an area where the target material was hidden. When Major encountered the odor from the target material, he would, as is typical, show a noticeable interest and begin to interrogate the area with his nose and when he was near the target material hide

would sit, or if he did not sit, the handler would verbally command Major to sit, then issue intermittent whistle commands for Major to return and along the way be verbally praised and then thrown a ball when in range of the handler. This training progressed by hiding the target material in more challenging and unexpected places including intermittent locations along the usual routes of a field trial run and in and around objects, buildings, and vehicles to which major would be directed or come near during a trial. This phase of training was considered complete when Major's following of directional commands was reliably interrupted by encountering a target odor and he searched for and sat (i.e, alerted) as close to the source of the target odor as he was capable.

Remote Tone and Vibration Control

Control of canine Major's movements was transferred from handler voice, whistle, and visual commands to remotely delivered tone and vibration signals by presenting both types of commands simultaneously and then gradually fading the original handler voice, whistle, and visual commands. For example, the "Back" tone would be issued immediately before and continue concurrently with the verbal "Back" command and across many repetitions of this pairing, the handler would reduce the intensity of saying "Back" to the point where only the initial syllable "B" was muttered softly.

Transferring the more complex right and left "over" commands, which include both the verbal command "Over" and visual cue of the handler making arm and/or actually moving in the desired direction, required both fading of the

original command and some additional prompting to obtain the desired control by the respective left and right vibrations. First, for the operations intended for the remotely controlled dog, it was desired that it remain in its previous orientation or heading rather than orienting toward the handler to receive a subsequent command, which was a necessity of the original manual handler commands, upon being commanded to "Stop". Therefore, during fading of the continuous whistle that signaled the dog to stop and concomitant issuing of the tone, another handler would stand on the opposite end of the track the dog had been commanded to move and prompt the dog to remain looking in his or her direction when it stopped. The presence of the additional handler was transferred to the visual cue of a pile of retrieval bumpers, which itself was eventually faded such that there were no visual cues. To aid in obtaining control of the left and right vibrator commands, initially a retriever bumper would be thrown in the desired direction upon issuing the tone to be associated with moving right and left. This thrown cue was replaced by a static pile of bumpers in the direction to be associated with the respective right and left tones, which was then replaced by hidden bumpers and finally the target odor the dog had been trained to detect in the direction associated with the respective right and left tones. Once this tone and vibration repertoire was established, the distance and complexity (e.g., number of changes in directions) of trials were increased.

Major's performance was further refined and made more reliable through daily practice sessions. In response to occasional occurrences of repeated movement errors, the electronic collar was used to deliver mild electrical stimulation and

collar vibration, but more often vibration alone, to reinforce correct movements. Use of electrical stimulation or the collar vibrator, both of which Major was accustomed to from prior field trial training, was used only after the tone controlled movement repertoire was well established and never to punish incorrect movement. Actual use of the electronic collar was rare; however, Major continued to wear the collar for most practice and experimental sessions such that its function was available if needed.

Final Training Status

Prior to the initiation of the experiments reported here, canine Major was fully proficient at being remotely guided by tone and vibration commands and his ongoing following of such guidance being interrupted by the presence of target odor to which he would follow to its source and sit (i.e., alert). Canine Major would exhibit this repertoire in a variety of environments including open fields, hard surfaces, around and about the exterior of buildings, and inside of large structures, such as subterranean mass transit (i.e., subway) venues.

One complicating aspect of Major's final performance for the execution of the current experiments was that he tended to take shortcuts to locations where he had previously encountered target odor and/or to structure and objects similar to those where he had encountered target odor in the past. This propensity to disregard movement commands to go to such productive (i.e., areas associated with presence of target odor in past) areas was considered desirable as compared to the added control afforded by more consistent use of negative reinforcement

that would be necessary for following movement commands. Such added control might interfere with the precedence established for disregarding movement commands to respond to the presence of target odor, which was the immediate operational imperative of the ongoing technology development project in which Major was participating before, during, and after the experiments reported in the present chapter. It is the belief of the authors that the guidance accuracy reported in these experiments could ultimately be improved with consistent training, albeit at the expense of performance on other required tasks. There are no plans to change the canine training routine at this time.

5.3 System Architecture

5.3.1 Overview

This section describes the interacting components which constitute the command and navigation system. The descriptions describe the existing system which has been designed for the purpose of testing the various sensors, gathering data for the testing of filtering techniques, and testing the responsiveness of the dog to complex series of commands. A high level operational view of the phase II system was previously described in Figure 3.2.

5.3.2 The Vest

All of the sensor and actuation equipment was mounted upon a custom made harness (Blackthorn K9 Equipment, Inc.). The vest has adjustable straps to make sure that the vest fits snugly while minimizing shifting over the canine's fur as much as possible and while maintaining the comfort of the canine. The completely packed vest (with all batteries,

sensors, and actuation devices) weighs approximately 3.2 kilograms, or about 10% of the canine's body weight.

5.3.3 The Command Module

A Rabbit microprocessor system was used to interface the sensors, the tone generator, and the radio modem. These small embedded computer systems are compact, use relatively low power, and support several serial connections in order to interface to a variety of sensors and communications devices. The focus on small, low-power systems results from the need to have a system that can ride on the dog's harness comfortably. The Rabbit 4100 Model was used for the results reported in this chapter. Further hardware details of this module are available at [Rabbit Semiconductor 2008].

A custom circuit board was developed to interface the Rabbit, the GPS receiver, the XSens, and the Xbee radio modem. The embedded system was programmed using the Dynamic C environment. Its primary responsibilities are to read data from the sensors and tone generator and then encapsulate that data in a packet transmitted over the Xbee radio modem connection.

5.3.4 Tone and Vibration Generator

The tone generator described in Section 5.2.3 is connected via a serial connection to the command module and communicates a small data packet indicating the start and stop of a command by the human trainer. This allows for the real-time recording of the commands in conjunction with the sensor data.

5.3.5 Sensor Suite

This section discusses the basics of the sensor suite used to track the canine.

GPS Receivers and Antenna

The GPS receiver used for this phase was an off-the-shelf UBLOX Antaris TIM-4H series receiver. In this work, GPS provides latitude, longitude, velocity, course (when moving), and elevation information. However, GPS sensors can only produce this information with a low frequency compared to other sensors that do not depend on receiving communications from satellites. Further, course measurements become noisy at low velocities since the receiver calculates course from the components of velocity. A rigorous discussion of the relationship between velocity and course accuracy appears in [Daily and Bevly 2004]. This necessitates the use of additional sensors for higher-frequency updates of information and for times when GPS cannot be accessed due to physical conditions (e.g., moving close to a large building, indoors or into a forest). All of these measurements are provided via serial connection to the Rabbit core.

Initially, a small active GPS antenna, an Antenna Factor SH model, was used in conjunction with the receiver. However, after extensive testing, it appeared that the Xbee Radio Modems produced interference with the GPS signal on this smaller antenna. This caused loss of satellites which reduced GPS accuracy (4m or worse). Switching to a larger, higher-quality antenna (Nov Atel 702-L) dramatically reduced the impact of interference leading to consistently far greater GPS accuracy (1.5m - 2.5m). A lightweight piece of wood and and an aluminum bolt were used to affix the antenna to the vest so as to be as comfortable for the canine as possible and to minimize roll.

XSens

The XSens is a miniature inertial measurement unit (IMU) containing six low-cost Micro-Electro-Mechanical Systems (MEMS) grade sensors: three accelerometers and three gyroscopes. Further, it contains integrated 3D magnetometers. The XSens unit provides measurements at a much higher frequency than GPS (for the specific details of the measurements, see Section 5.4.3).

In this system, the XSens produces sensor data for acceleration in the x,y, and z axes as well as orientation data (roll, pitch, yaw). In this case, the x-axis runs along the canine's back pointing forward toward the canine's head. Roll will be interpreted as angular movement around the x-axis measured in degrees (-180,180). The y-axis points out of the canine's left side. Pitch is angular movement around the y-axis measured in degrees (-90,90). The z-axis points up from the canine's back and yaw is angular movement around the z-axis measured in degrees (-180,180). In order to compare yaw data to GPS course, yaw is converted to eastward deviation from North (0,360).

These measurements alone are less accurate than GPS for determining position due to inherent sensor drift and bias. The XSens can be configured to output filtered measurements for orientation. The filtering in this case is the result of the application of the XSens Kalman Filter algorithm which uses the accelerometers' measurement of gravitational acceleration to stabilize the attitude (i.e., roll and pitch) estimates, which are derived from the gyroscopes. Further, it uses the magnetometer readings in order to filter out drift for the measurement of yaw. Additional details of these algorithms are described in [XSens 2009]. In order to parameterize the XSens Kalman Filter, the device's "Human: High Acceleration" profile was chosen, since these motion characteristics most closely match the high accelerations

produced by the canine gait. The key result of this filtering is an output of measurements that are far more accurate and less prone to drift than using gyroscopes, accelerometers, or magnetometers alone. This is particularly true at high accelerations, such as those presented by the movement of the canine. Since ferromagnetic materials were present on the vest and these materials are measurable by the magnetometers, the XSens' calibration process was utilized to compensate for the presence of static ferromagnetic materials.

5.3.6 Data Sink

The Data Sink is a Windows-based laptop system which collects sensor data from the Xbee radio modem through a second Xbee connected to the laptop via a Maxstream USB Xbee Development Kit. The primary role of the laptop is to be an outlet for the large amounts of data collected by the sensors. No command information travels from the laptop to the command unit. The receiver software is a stand-alone C++ program compiled using the free Borland 5.5 compiler.

5.3.7 Xbee Radio Modem

This small, power-efficient serial radio modem communicates at 38400 baud rate, which is sufficient to communicate the amount of data being produced by the sensors, GPS, and the tone generator. It is used for communication with an external datasink (a laptop) since the Rabbit has limited data storage capacity. In these trials, the Xbee with only a small external antenna provided virtually error-free dataflow up to approximately 150m. Beyond that distance, packet losses became more frequent.

5.4 Experimental Setup

5.4.1 Experiment Setting

In order to reduce potential bias in the canine, sets of trials were performed using different waypoints and where possible, different fields. This was done to prevent the canine from learning the specific waypoints of interest rather than learning to follow the tone and vibration commands. An additional observation is that the canine is less likely to be willing to go to an area that he has already searched recently (thus potentially skewing the results). The waypoints used for the trials consisted of both existing waypoints (trees, vehicles, abandoned buildings) and artificial waypoints (trashcans, cones, and flags). Waypoints were generally between 10 meters and 120 meters apart. In many trials, small bumpers (plastic objects the dog finds in search training) were hidden at each waypoint to give the dog something to find. In these cases, multiple bumpers were placed at different waypoints to guarantee that the dog was making the correct waypoint decision (not merely finding a bumper). Trials were performed during the daytime, generally in the morning. The trial sets reported in this chapter were performed over nine different days days over a six month period.

Overall, there were a total of nine distinct trial sets with a total of 129 distinct trials (with between six and nineteen trials per day, usually depending on weather). Especially during early trials with the system, there were several cases where equipment failures caused a trial to be excluded from consideration. These failures were typically caused by the dog's movements knocking loose cable connections or the battery being knocked off. If the equipment failure interfered with the dog (e.g. a cable falling near his face), that trial was excluded from consideration. It should be noted that as adjustments were made to the

configuration and placement of the equipment on the canine vest, these issues occurred far less often.

5.4.2 Trial Descriptions

After each trial set consisting of four to five trials, the canine was given breaks without any equipment on him. In order to test the system, it was carefully attached to the canine's vest. The canine was then commanded using the Motorola radios to run to one or more waypoints and then to return to the point of origin.

The simplest trials required the canine to leave the point of origin, go to a waypoint, stop, and then finally return to the point of origin. This scenario might be comparable in an operational setting to inspecting a building with a known potential to harbor detectable materials. In this case, a "success" means that the canine came within at least seven meters of the destination waypoint and stopped on command. The distance of seven meters was chosen because at this distance the canine could either see a physical waypoint (like a building) or would be close enough to smell explosives under most environmental conditions. A "failure" indicates that the canine could not be commanded to arrive at the waypoint and was recalled by the human trainer. It is generally not necessary for the canine to land directly on the target waypoint, because trained search dogs will automatically begin searching a target at that close proximity without any additional directional commands from the canine trainer. There are some trials when the canine was stopped slightly early because it was difficult for the trainer to see precisely how close the canine was to a target; these trials are still considered a success in terms of the dog's behavior. An example of a simple trial is shown in Figure 5.2.

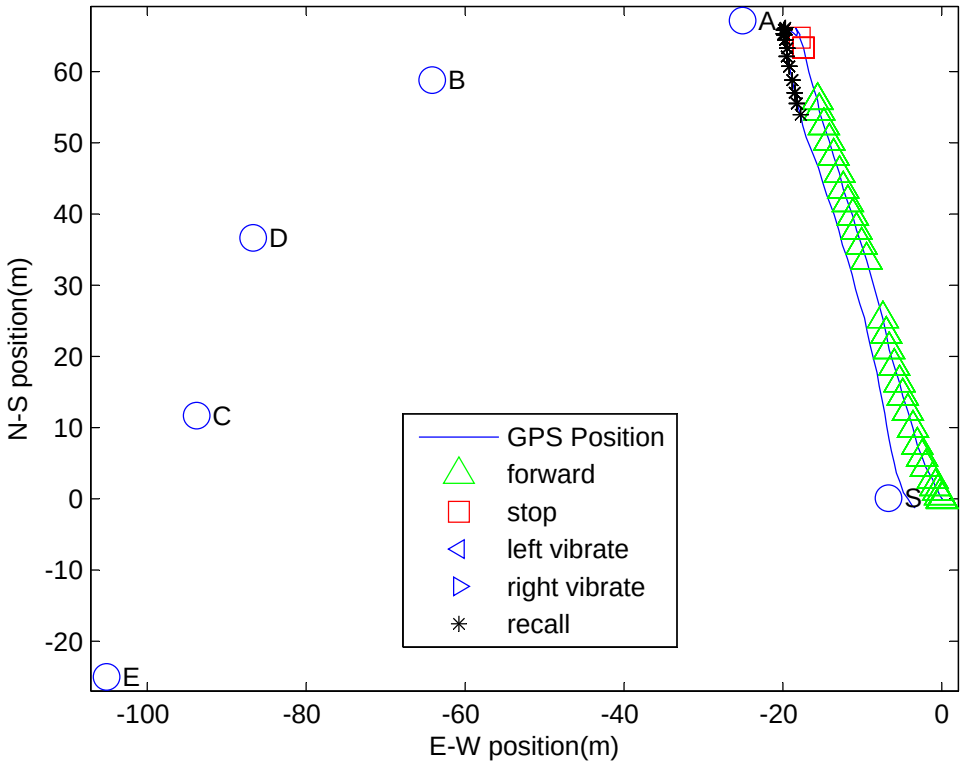


Figure 5.2: GPS navigation of the canine on a simple path from the start position S. A continuous forward command is held until the canine reaches close proximity to the goal waypoint A. The other waypoints B,C,D, and E are foil waypoints. Then, a stop is issued followed by a recall. This would be considered a success.

More complicated trials included intermediary waypoints before the ultimate destination. These trials are more difficult for the canine because they require a greater number of directional commands (like turn left and turn right) which the canine does not respond to in the same manner as human beings or vehicles might. In other words, the canine has a non-deterministic response to a command such as "turn right" or "turn left." Rather than a hard right or left turn, he will generally focus on some potentially searchable object in the right or left direction and move toward it. In an operational setting, multi-point paths could exist due to a need for avoiding obstacles or simply because the ultimate destination is far away. An example of a more complicated successful trial is shown in Figure 5.3.

5.4.3 Sensor Data Produced

As trials are being run, the sensor data is being displayed and saved to file in real-time on the laptop. Currently, this data is being displayed in a labeled text format. However, future efforts could reasonably change the display to fit the needs of trainers. For example, it was relatively easy to create a Google Earth view (see Figure 5.4) during post-processing using the freely available KML API. The possible applications of this data are far-ranging in operational scenarios. Efforts are currently underway to move to a real-time visualization display. The specific data being collected from the GPS receiver, XSens, and tone generator is shown in Table 5.1. The "Measurement" column gives a description of the metric, the "Units" column indicates the units the data is taken in, the "Frequency" column indicates the sensor collection rate, and the "Device" indicates which device produced the measurement.

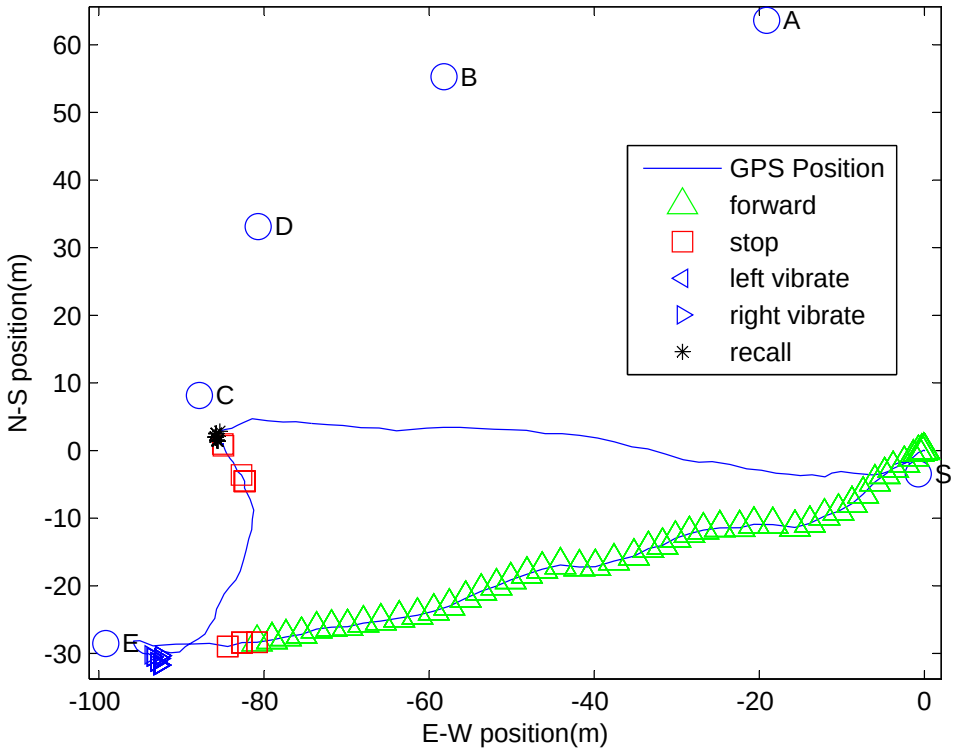


Figure 5.3: GPS navigation of the canine on a multi-point path from the start position S. The canine is first guided to waypoint E, stopped, and then reoriented using the right vibration. He then travels to point C, is stopped, and then is issued the recall. This would be considered a success.

Figure 5.4: A visualization of recorded sensor data from a canine trial. The KML file was created using a script and interpreted/plotted using Google Earth.

Measurement	Units	Frequency	Device
longitude	$degrees$	4Hz	GPS Receiver
latitude	$degrees$	4Hz	GPS Receiver
horizontal accuracy	mm	4Hz	GPS Receiver
height	mm	4Hz	GPS Receiver
North velocity	cm/s	4Hz	GPS Receiver
East velocity	cm/s	4Hz	GPS Receiver
Up velocity	cm/s	4Hz	GPS Receiver
3D ground speed	cm/s	4Hz	GPS Receiver
2D ground speed	cm/s	4Hz	GPS Receiver
course	$degrees$	4Hz	GPS Receiver
course accuracy	$degrees$	4Hz	GPS Receiver
acceleration in X,Y, and Z	m/s^2	25 Hz	XSens
roll	$degrees$	25 Hz	XSens
pitch	$degrees$	25 Hz	XSens
yaw (heading)	$degrees$	25 Hz	XSens
command	N/A	4Hz	Tone Generator

Table 5.1: Sensor Measurements from the GPS Receiver, XSens, and Tone Generator.

It should be noted that the XSens produces data at a higher frequency than the GPS receiver. The XSens is capable of reporting measurements at even higher rates than 25Hz (up to 112Hz), however there was no need for faster readings at this point in time. The command module produces a packet every time the XSens has new data, even if the GPS data is old. It includes the most recent GPS data with every packet that it reports, along with the current command being reported from the tone generator.

All waypoints are independently marked by GPS before the trials so that afterward the recorded canine paths can be compared to the known locations of the waypoints. This provides a way to determine the relative quality of the navigation measurements. A useful means of comparison was to overlay the canine path in Google Earth to determine if the imagery lined up with the GPS measurements.

While the measurement of position, velocity, and course have numerous applications (several of which have been previously discussed in this chapter), the use of canine orientation warrants some discussion. The roll shows the rotation of the pack on the canine (as well as the rotation of the canine himself). This could be used in studies attempting to determine what vest is best suited for a particular canine to determine if the rotation of a given vest is too great. The pitch measurement gives one key indication of the canine's pose. The change in pitch can be used in conjunction with other measurements to determine if the canine is sitting down, for example. Sitting is a common response to the canine's detection of explosives, narcotics, or other objects of interest. Finally, yaw can be used to determine the direction that the canine is facing in situations where GPS course is not accurate or undefined due to low/zero velocity. A simple example would be when the canine stops moving forward, but turns his body around. GPS would no longer be able to accurately track course in this instance, but the XSens's measurement of yaw (heading) would still be able to determine the change in the direction that the canine is facing.

5.5 Results and Discussion

In this section, the results of the trial sets from the case study will be shown and discussed, as well as some data on simulated GPS outages to demonstrate the data aggregation's capability to compensate for limited GPS outages. The p-values reported here are the results of a two-tailed Fisher's test, with $p < 0.05$ being considered statistically significant.

Date	Total	1PT S.	1PT F.	1PT SR	MPT S.	MPT F.	MPT SR
11-07-08	19	8	2	0.80	7	2	0.78
12-05-08	15	4	0	1.00	8	3	0.73
01-23-09	16	9	4	0.69	3	0	1.00
02-11-09	22	2	0	1.00	7	13	0.35
02-13-09	13	2	1	0.67	6	4	0.60
03-11-09	14	3	2	0.60	6	3	0.67
03-18-09	6	1	1	0.50	3	1	0.75
04-10-09	7	4	0	1.00	2	1	0.66
04-17-09	17	2	4	0.33	8	3	0.73
Summary	129	35	14	0.71	50	30	0.63

Table 5.2: A Breakdown of Single Point and Multi-Point Trial results taken over a six-month period with the trained Canine.

5.5.1 Canine Trial Results

The success or failure of each trial was noted by the human trainer and the operator of the laptop. The success or failure of a trial was denoted independently of the computer data. This was done so that the canine's behavior could be correctly categorized even if there was some unknown failure such as an undetected sensor failure, for instance. In Table 5.2, a breakdown of the trial results shows the total number of trials for a given day, the successes and failures for single point trials, and the successes and failures of multi-point trials.

This data provides some interesting results. First, the success rate for simple paths (71%) is higher than the success rate for the more complicated multi-point paths (63%). However, this result is not statistically significant ($p = 0.34$). The lower success rate for complex paths is expected, given that the number of commands that the canine must follow correctly increases considerably on paths which require more turns. However, it is promising since the data and statistics indicate that successfully performing directed missions of this

type are feasible. Second, a large number of errors occurred on a single trial date (02-11-2009). This particular trial date contains many of the multi-point errors observed, which indicates anomalous behavior - the success rate (35%) for that day was lower than the success rates on the rest of the trial days ($p = 0.006$). This can likely be attributed to environmental factors; the trial location included an unusually large number of searchable targets (dense buildings) in the immediate vicinity which posed as a distraction. Recall that the canine's search training takes precedence over his directional training.

With respect to the sensor navigation, the sensor provides the best data when there are few obstructions to GPS (accuracy as described in Section 5.3.5). The most difficult situations are when large nearby buildings introduce error into the GPS by blocking signal. In these cases, the sensors indicate a loose bound on the movements on the canine. GPS measurements were less accurate (3 to 4 meters horizontal accuracy) in the presence of large buildings, but otherwise the GPS measurements proved to be consistently quite accurate. In general, the canine demonstrates reasonable response to tones such that with continued training and refinement, this would be a reasonable platform from an operational standpoint.

5.5.2 Analysis of a Field Trial Based on Sensor Data

The goal of this section is to visualize and analyze a sample canine trial using the data produced by the system. While the data in this section is taken from the log file associated with this trial, all of the data itself is available in real-time. In other words, there is no additional post-processing of the data itself shown here; a discussion of post-processing capabilities is reserved for Section 5.5.3. This section further illustrates the availability data

which could be used for a number of applications discussed elsewhere in the chapter. This description does not intend to infer that these exact motion characteristics (accelerations, pitch while sitting, etc.) would be exactly the same in every canine, but merely that many motions would be observable from the data even if the canine was out of the line of sight of a trainer or handler.

A position plot appears in Figure 5.5, which shows the canine being commanded from starting position S to waypoints A, B, C, and D in that order. A set of four subplots detailing 3-axis acceleration, roll, pitch, yaw, and command data for this same trial is shown in Figure 5.6. All four of these subplots are shown synchronized in time with respect to their x-axes.

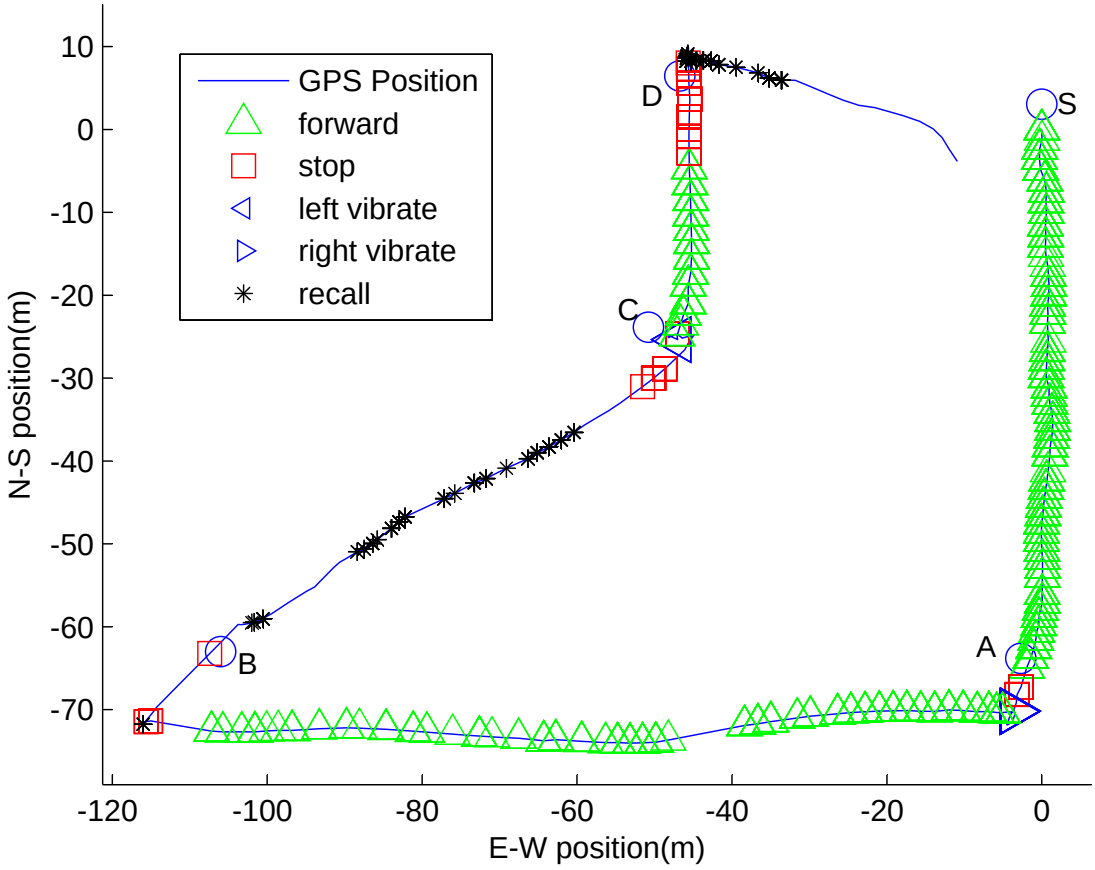


Figure 5.5: Sample human-guided canine trial.

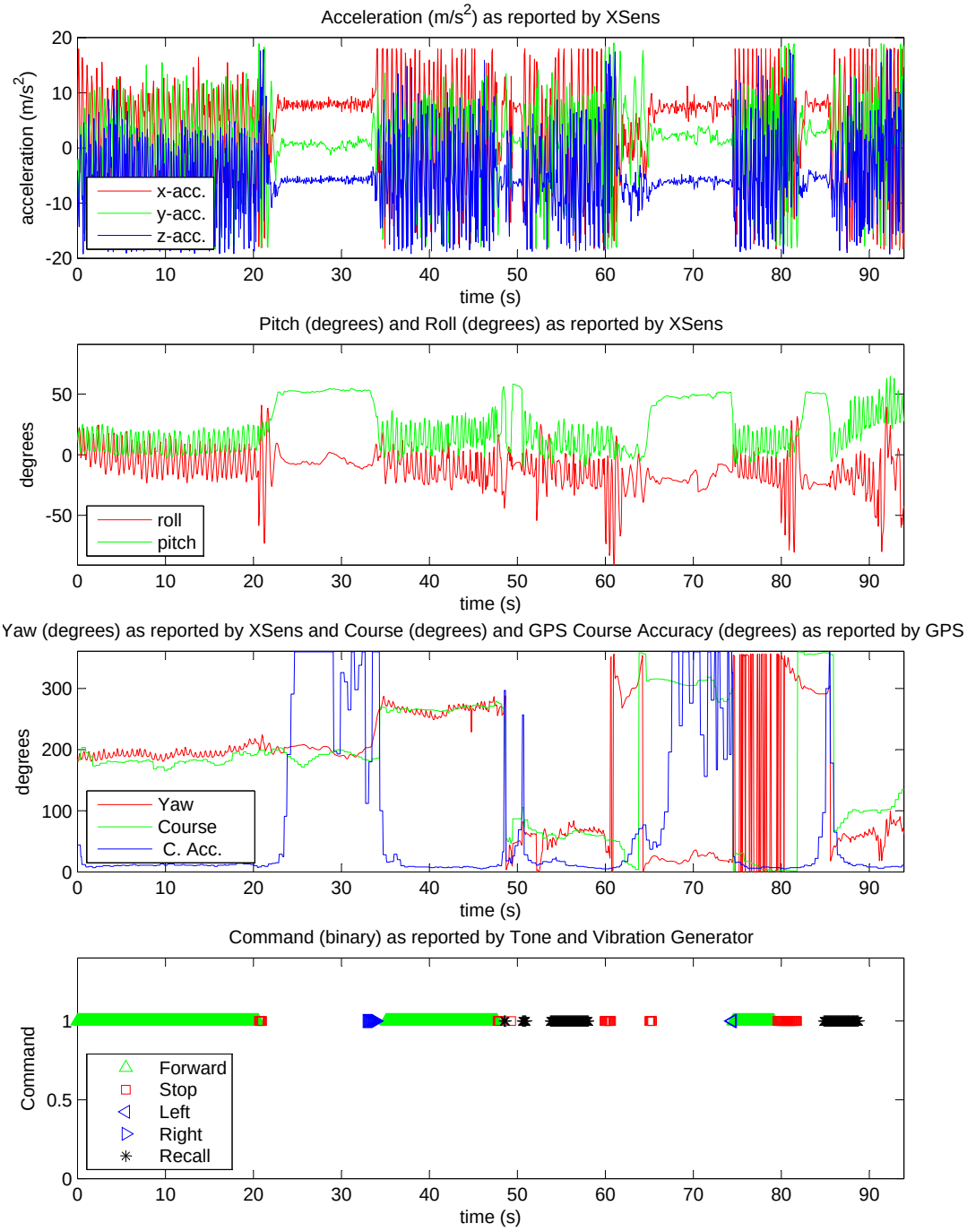


Figure 5.6: Subplots of acceleration, pitch, roll, yaw, course, and active command in time for the sample canine trial

The trainer starts the canine at waypoint S and then issues forward until the canine arrives near waypoint A and is stopped. In the position plot, the stop command is shown in proximity to the waypoint A. The very large accelerations shown in the acceleration plot () settle to the components of gravity ($9.81 m/s^2$) during a stop. The canine sits, which causes pitch to stabilize at approximately 50 degrees in this case The value of pitch depends on the orientation of the canine and the slope of the terrain. In the case of Major, this motion indicates sitting which could be used to identify a response to odor detection (when a stop command was not issued, which also causes him to sit). The observation of roll when the canine is moving indicates both a natural roll associated with the quadrupedal motion and a roll associated with the vest twisting slightly as it slides on the canine's fur. Rapid changes in acceleration, roll, and pitch drop off as the canine sits for the stop.

After approximately 8 seconds during which the canine is stopped, the trainer issues a right turn command in order to reorient the canine toward waypoint B. This change in the canine's facing is rapidly reflected in the yaw data issued by the XSens (see subplot 3 in Figure 5.6), but GPS course lags behind since the canine has very low velocity at this point in time. The measure of Course Accuracy indicates (in degrees) how close the reported course value is to the actual course. In other words, a 0 Course Accuracy indicates a perfect measurement whereas 360 indicates no certainty regarding course at all. The canine is issued a forward and travels near waypoint B, which was located on the corner of a building. After a movement of searching around the edge of the building, the canine arrives at point B.

Since the straight-line between waypoint B and the start position S would take the canine to waypoint C, the trainer issued a recall command followed by a stop. This stems

from the historical observation that the canine has a much higher accuracy in responding to recall commands than he does for the left or right directional commands. The canine begins traveling back to the start position, but then is stopped near waypoint C. This time a left command is issued to reorient the canine toward waypoint D (also a building). Again, yaw's correction for the change in facing of the canine precedes GPS course. The trainer issues a forward and the canine travels to waypoint D. This concludes the trial and the canine is recalled.

This trial data provides insights into several items of interest to both canine trainers and operators of canines in search missions. Roll and pitch can be analyzed in order to determine canine pose (e.g. sitting). Accelerometers indicate when the canine is still. Observing canine position and pose coupled with commands allows the trainer to analyze trials in a precise way after the trial has passed.

5.5.3 Data Aggregation with a Simulated GPS Outage

Section 5.5.3 on Techniques for Compensating for Short-Term GPS Loss is used with expressed permission from the author [Miller and Bevly 2007; 2009]. This is provided here to give the reader an understanding of how this system could be applied in situations where there are short term GPS losses.

> One of the benefits of having additional sensors beyond GPS alone is that a reasonable estimate of the canine's position and orientation can be maintained even if brief GPS outages occur. This is particularly important in the case of a search canine since it is possible that the dog will enter buildings or move into areas where the line of sight for some of the satellites is lost.

The Kalman Filter is an effective tool that can be used to integrate acquired measurements from a GPS receiver (e.g., position, velocity, and course) with acquired measurements from magnetometers and an Inertial Measurement Unit (IMU), comprised of accelerometers, which measure acceleration in a given direction, and gyroscopes, which measure the rate of turn in a given direction. Although GPS measurements prove to be relatively accurate, the rate at which they are taken is much slower than the rate at which an IMU takes measurements. Thus, if there are higher frequencies in the movement of the subject that is being tracked than the sampling rate of the GPS receiver, such as in the case of canine motion, aliasing will occur. Also, although the GPS signal can be lost from time to time, magnetometers and inertial sensors will be able to continue providing measurements. However, an IMU tends to to have inherent bias. These factors can be reduced or eliminated by using the Kalman Filter to estimate them. So, integrating the measurements from the different sensors with a Kalman Filter can help to achieve more accurate results during GPS outages [Bevly 1999, Godha 2006, Ryu and Gerdes 2004, Bevly 2004].

The Kalman Filter is used for linear systems. However, the Extended Kalman Filter (EKF) allows filtering of non-linear systems, such as those found in typical navigation filtering scenarios, and is described in detail elsewhere [Stengel 1994]. The EKF was used in our estimation of the position and course of canines, and the details have been presented elsewhere [Miller and Bevly 2007; 2009].

The unique motion characteristics of living biped and quadruped species have proven to require specialized tuning of the Process Noise Covariance (Q) Matrix

in the Extended Kalman Filter, in comparison to typical vehicle applications [Miller and Bevly 2007; 2009]. GPS/INS integration has been effectively utilized in tracking the motion of pedestrians and horses [Upadhyay et al. 2003, Gabaglio 2003, Ladetto et al. 2001, Hill et al. 2007], but very little work has been done in using this technology to track the position and orientation of canines, which intuitively have different motion characteristics, such as higher frequency, erratic movements and higher rate of turns and stops. As was mentioned above, EKF estimation algorithm improvements for canine applications have been made [Miller and Bevly 2007; 2009] and further improvements are currently underway to improve the canine position and orientation estimates, especially during GPS outages.

Figure 5.7 illustrates an example of the current EKF algorithm course results for a canine. The blue dots represent GPS course measurements. The red line represents the EKF course estimate after integrating GPS with the magnetometers and IMU. The black line represents the EKF course estimate after a 20 second simulated GPS outage halfway through the test.

A drift from the EKF solution (without GPS loss) and GPS course measurements is apparent. However, as reported in [Miller and Bevly 2009], a comparison of the mean difference between the EKF course estimate (with the GPS outage in place) and GPS course measurements for this test revealed a 6.3% improvement in the course estimate when magnetometers are used along with gyroscopes to estimate course. Magnetometers keep the course estimate from diverging

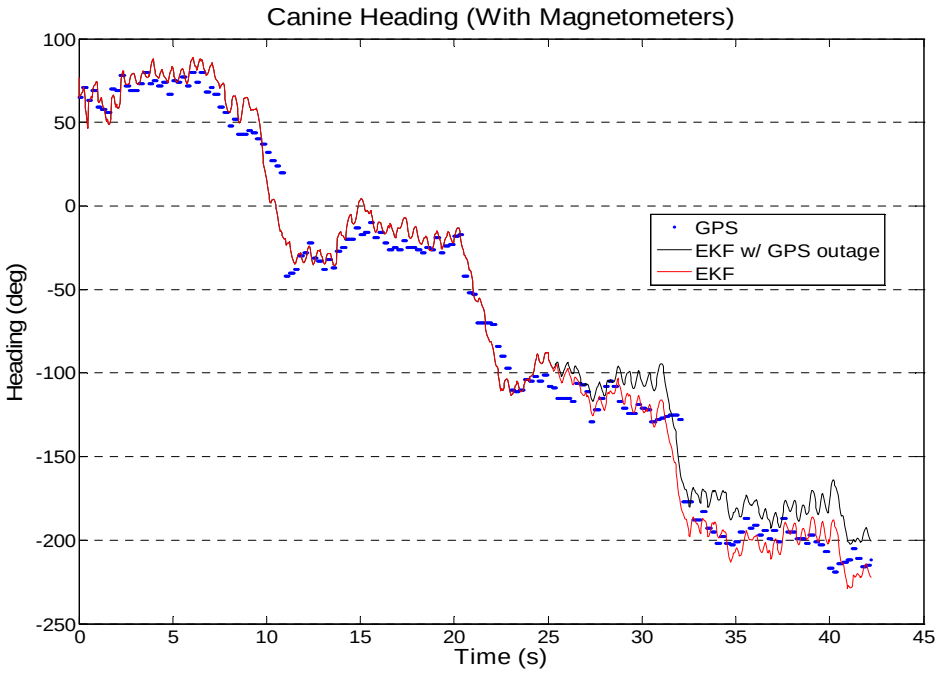


Figure 5.7: GPS course measurements, EKF course estimate, and EKF course estimate with a simulated GPS outage for a canine.

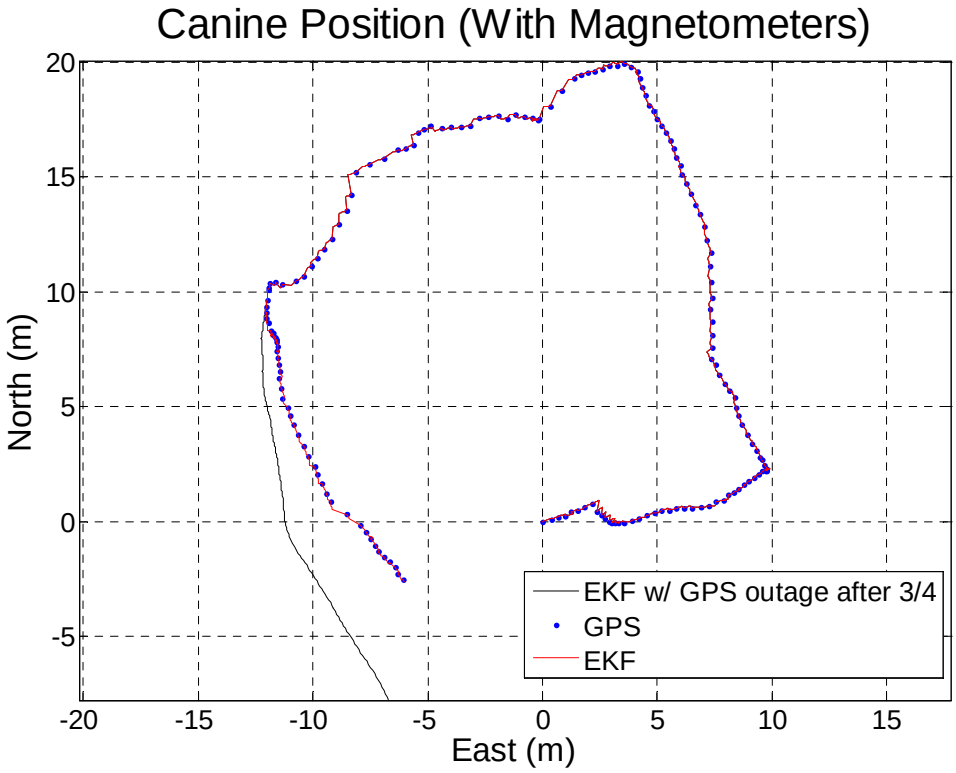


Figure 5.8: GPS position measurements, EKF position estimate, and EKF position estimate with a simulated GPS outage for a canine.

from the EKF and GPS course measurements as quickly as calculating a course estimate based on gyroscope and GPS course measurements alone.

The effects on the course estimates due to sensor pitch changes, like those found in the rocking motion of a canine while running, are currently being investigated. Preliminary data indicates that pitch changes severely affect the course estimate, especially for magnetometers.

Figure 5.8 exhibits the EKF position estimate for a canine. As before, the blue dots represent GPS position measurements, the red line represents the EKF

position estimate, and the black line represents the EKF position estimate after a 10 second GPS outage.

In this test, after ten seconds of GPS outage, the final position estimate drifted only about 5 meters from the actual position. However, results varied, dependent on velocity, inclination/pitch, and course changes during the outage. Some position estimate errors were significantly greater with such a long GPS outage. It is believed that this is due primarily to a corrupt velocity estimate because of the lack of incline and pitch change modeling in the EKF algorithms. Although these effects are often negligible in automobile applications, they can be significant in canine motion. Pitch affects the accelerometer measurements, which in turn, affect the velocity estimates, especially during GPS outages. The EKF algorithms are currently being adjusted to include sensor measurements that will allow for pitch estimates and an investigation of their effect on canine position and course estimates. Preliminary data indicates that the EKF position estimates are improved when incline and pitch are estimated and included in the sensor models.

5.6 Chapter Conclusion

The research in this chapter has shown the following:

1. The ability to command and track a trained canine in real-time using a sensor suite and a tone generator. This provides an advantage both to trainers and in a mission setting.

2. The ability to post-process, filter, and visualize canine tracking data to provide a useful tool for a canine trainer, experts in canine movements, and for those interested in modeling the response of the canine to actuation commands.

3. A case study which demonstrates the feasibility of a human commanding a trained canine using non-invasive techniques.

4. An analysis of the real-time sensor data and discussions of real world applications of said data.

5. An analysis of a significant number of trials with a human commanding a trained canine to verify the canine's capabilities and the system's consistent ability to perform tracking. This provides a data vehicle for research on the modeling of the command of canines (which will be used for developing the autonomous control algorithm in Chapter 6) as well as data to help enhance the tracking of canines.

6. Demonstrations of the ability to perform sensor aggregation using the data recorded from these trials for the purpose of dead-reckoning and improved position, velocity, and course tracking.

As the project continues, efforts will be made to continue to improve the ability to command and track the dog effectively to bring down the canine and sensor error rates. With further refinement of filtering techniques, the tracking will become even more robust, accurate, and reliable. Ultimately, this work serves as the platform for the autonomous canine phase of the project, presented in the next chapter.

Chapter 6

An Autonomous Control Algorithm

6.1 Chapter Introduction

This chapter will develop the automated guidance of the trained canine discussed in Chapter 5. The automated guidance is made possible given a canine capable of following directional commands, the ability to track the position of the canine, the equipment with the ability to issue directional commands via hardware, and a control algorithm sophisticated enough to correctly identify canine behavior through sensors and then to calculate and issue commands. The control algorithm will have the capability to detect anomalous behaviors and respond appropriately (either by re-issuing commands in order to get a correct response or by ending trials when all attempts at anomaly correction have proven unsuccessful).

The canine control algorithm described in this chapter is a parameterized state machine which bases its state transitions on sensor readings and heuristics. The expert knowledge introduced into the system is the result of considerable analysis of real canine trials, extensive discussion with canine trainers, and the results of control algorithm field trials. The control algorithm for the guidance of the canine must meet the following (prioritized) criteria:

1. Given correct responses from the canine (i.e. the canine turns left when given the command to go left), the control algorithm should always be able to direct the canine in such a way as to successfully complete any mission.

2. The control algorithm should be able to tolerate reasonable deviations in canine behavior, as long as the behavior is generally correct. Deviations occur due to inherent

noise in sensor data, the inherently erratic movements of the canine, the canine's reaction to events unobservable by the sensors, and natural deviations from a "perfect" course. These conditions should not cause the control algorithm to issue correcting commands since this will serve to confuse the canine and reduce the mission success rate.

3. The control algorithm should be able to quickly identify obvious anomaly conditions (i.e. turning right when the Left command was issued) and correct the canine.

4. The control algorithm should be able to recognize when the canine is no longer responding to correcting commands and safely recall the canine.

The remainder of this chapter is organized as follows. A detailed discussion of the changes to the system architecture between Phase II (described in Chapter 5) and the Autonomous Canine Phase appears in Section 6.2. The applicable sensor data used by the control algorithm is shown in Section 6.3. The state machine control algorithm is developed in detail in Section 6.4, and its mechanisms for detecting and correcting anomalies are described in Section 6.5. The experimental setup is described in Section 6.6 and results and discussion are provided in Section 6.7. Finally, some concluding remarks are given in Section 6.8.

6.2 Changes to System Architecture from Phase II to Autonomous Canine Phase

The Training Phase II platform described in Section 5.3 was used as the baseline hardware and software system, with several critical modifications to facilitate the command of the canine. These modifications are outlined in the following sections.

6.2.1 The Command Module

The Rabbit Microprocessor itself was unchanged between Phase II and the Autonomous Canine Phase, but was reprogrammed to issue the commands to the Tone and Vibration Generator rather than to merely record the tones being issued by a human being. Additionally, some minor synchronization was performed between the embedded software and the control algorithm located on the laptop in order to indicate the start and stop of missions. This allowed the system to minimize transmission by the radios outside of a mission state, saving power.

6.2.2 Tone and Vibration Generator

The tone generator described in Section 5.2.3 was connected via a serial connection to the command module and is reconfigured to accept a small data packet indicating the start and stop of a command as determined by the control algorithm. This allows for the real-time control of the canine.

The available tone commands provided by the Tone and Vibration Generator are summarized in Table 6.1 and their meaning to the canine was described in greater detail previously in Chapter 5. Throughout the text, the capitalized word indicates the command and

Command	Actuator	Desired Reaction
FORWARD	unique tone	move forward
STOP	unique tone	stop immediately
RECALL	unique tone	return to initial trial position
LEFT	vibration	turn left
RIGHT	vibration	turn right

Table 6.1: Summary of Tone and Vibration Commands

non-capitalized versions indicate the conventional usage of the word. For example, "Left" indicates the Left command and "left" indicates left according to the normal interpretation in the English language.

6.2.3 Sensor Suite

This section discusses the basics of the sensor suite used to autonomously guide the canine.

GPS Receivers and Antenna

All control experiments were performed using the larger GPS antenna (Nov Atel 702-L) discussed in Section 5.3 due to its higher accuracy over smaller alternative GPS antennas and due to its resistance to radio frequency interference.

XSens

The XSens was not used in these control experiments in order to reduce the computational power required on the rabbit to process the large amounts of data produced by the XSens. Removing the XSens simplified the computational load on the embedded system, reduced the complexity of radio communications, and reduced the interference produced by the radios.

6.2.4 Data Sink

The Data Sink was a Windows-based laptop system which collects sensor data from the Xbee radio modem through a second Xbee connected to the laptop via a Maxstream USB Xbee Development Kit. The role of the laptop evolved since it now not only collects the sensor data being sent by the embedded system, but also performs the calculations for the control algorithm, and logs all sensor and control information on the laptop for future inspection. The control software was a stand-alone C++ program compiled using the free Borland 5.5 compiler. A detailed description of the canine control algorithms follows in Section 6.4.

6.2.5 Xbee Radio Modem

In the Autonomous Control Phase, the Xbee radio modems provide two-way communications between the laptop and the embedded system. In other words, the modems transmit the sensor information from the embedded system on board the canine to the laptop, then the control algorithm on the laptop calculates any necessary command changes, and then the Xbee connected to the laptop transmits the command back to the embedded system.

6.3 Sensor Data Used for Control

6.3.1 Raw Data from the GPS Receiver

Using the GPS Receiver as the only sensor produces a subset of the metrics discussed in Section 5.4.3. For convenience of the reader, the updated list of raw metrics is provided below in Table 6.2. Notably absent are the XSens data and the command (which is now being produced by the control algorithm rather than being recorded from the human being).

Measurement	Units	Device
longitude	degrees	GPS Receiver
latitude	degrees	GPS Receiver
height	mm	GPS Receiver
height above mean sea level	mm	GPS Receiver
North velocity	cm/s	GPS Receiver
East velocity	cm/s	GPS Receiver
Up velocity	cm/s	GPS Receiver
3D ground speed	cm/s	GPS Receiver
2D ground speed	cm/s	GPS Receiver
course	degrees	GPS Receiver
horizontal accuracy (HACC)	mm	GPS Receiver
course accuracy (CACC)	degrees	GPS Receiver
current command	n/a	Rabbit Microprocessor

Table 6.2: Sensor Measurements from the GPS Receiver and Embedded System usable for the control algorithm. The GPS receiver produces measurements at 4Hz and reports the current command at that time.

It should be noted that the reading of "current command" is no longer a reading from the tone generator, but rather an indication of the what command the embedded system is currently issuing to the tone generator.

6.3.2 Derived Metrics Used for Control

There are a number of static and time-based metrics that are useful for the control of the canine that can be calculated using the raw sensor data and a simple buffer of sensor readings. These metrics are provided below.

Distance to Next Destination

The Distance (D) is the distance between the most recent latitude and longitude position of the canine (as reported by the GPS receiver) and the known desired destination (marked prior to the beginning of the trial). The calculation for determining distance

comes from [*u-blox: Support, Tools and Help: FAQ : About GPS* 2008]. The calculation is as follows:

$$dLat = latitude_{current} - latitude_{destination}$$

$$mLat = latitude_{current} + latitude_{destination}$$

$$dLon = longitude_{current} - longitude_{destination}$$

$$dMLat = dLat * 111199.233$$

$$dMLon = dLon * (111199.233 * \cos(mLat * \pi/180.0)$$

$$D = \sqrt{dMLon^2 + dMLat^2}$$

This method for calculating distance is considered accurate for distances less than kilometers, which is reasonable for the distances of the trials performed in this research. If the control algorithm were to be used for waypoints with great distance between each other, the Great Circle algorithm would be preferable. It is worth noting that D in this chapter is calculated in meters, unlike in Chapter 4 when it was calculated in raw degrees. For the purpose of writing and parameterizing the control algorithm, meters are more intuitive than raw degrees.

Difference Between Actual Course and Desired Course to the Current Destination

Difference Between Actual Course and Desired Course to Current Destination (DDC) is used for both anomaly detection and to calculate the correct vibrations for turns. In order to have some indication of the canine's course in relation to the canine's desired course, the signed difference is taken. Actual Course is the canine's current course as reported by the

GPS receiver. Desired Course (DC) is calculated based on the navigation formula provided in [Williams 2008]. The calculation is as follows:

$$x = (longitude_{destination} - longitude_{current}) * \cos(latitude_{destination})$$

$$y = \cos(latitude_{current}) * \sin(latitude_{destination}) - \sin(latitude_{current}) * \cos(latitude_{destination}) * \cos(longitude_{destination} - longitude_{current})$$

$$DC = mod(atan2(x, y, 2\pi))$$

In the above formula, the atan2 function denotes a version of arctan which takes into account the quadrant in the cartesian plane in which the calculation is currently being made. From DC, DDC can be calculated as:

$$DDC = course_{current} - DC$$

Since DDC is signed, it automatically provides knowledge not only of the magnitude of the deviation, but also the direction in which to turn the canine in order to correctly guide him to the goal location. For example, a negative DDC indicates the error is due to the canine being oriented left of the DC, indicating a needed right correction.

Difference Between Actual Course and the Desired Course to the Next Destination

The Difference Between Actual Course and the Desired Course to the Next Destination (DDC2) is simply the DDC to the next waypoint in the destination list after the current destination. It is calculated in the same way as DDC otherwise. The need for this metric stems from the fact that GPS's ability to accurately indicate course diminishes (and ultimately becomes undefined) as velocity approaches zero (since GPS uses the components of velocity in order to calculate course). DDC2 is used to maintain an estimate of anticipated

turns when the canine is entering a stopping condition which will make GPS course inaccurate. For example, if the canine is currently headed due North to a waypoint and the next destination will require a left turn, DDC2 should be calculated before the canine is stopped at the next waypoint (so that it will be based on a course obtained while the canine is moving). If there is no next destination after the current destination (i.e. the current destination is the final destination), DDC2 is set to the sentinel value of -999.

Readings Since Distance Has Shown Improvement

The Readings Since Distance has Shown Improvement (RSIDIST) measures whether or not progress is being made toward the current destination over time. Each time a new sensor reading is made available, RSIDIST is calculated by comparing the current D to the D metric from a given number of prior readings. The number of readings to look back is determined by a system parameter (LOOK BACK AMOUNT, described in Section 6.4.1). The rationale for looking back a number of packets instead of looking at the immediately preceding reading is to compensate for the inherent noise in GPS sensor readings. In other words, while it is possible that sensor inaccuracies could lead to misleading changes in D from reading to reading, it is less likely to concur when looking back several readings since the magnitude of the distance is likely larger and less likely to be obscured by sensor error.

Bad Course Count

Bad Course Count (BADCOURSE) is the number of readings where the magnitude of DDC exceeds some threshold (COURSE TOLERANCE, described in Section 6.4.1), which is a parameter of the control algorithm. If DDC exceeds MAX CORRECTABLE DDC, then BADCOURSE is increased by three steps (instead of just one) to penalize extreme

deviations from course. This serves as an indicator of anomalous canine behavior since it indicates a trend of moving away from the next waypoint.

6.4 The Control Algorithm

The control algorithm is a parameterized, state machine solution to perform canine control given the actuators available for controlling the canine. The primary goal of the control algorithm is to be able to effectively determine which of the five commands should be issued to the canine at any given time, based on the sensor information available to it.

6.4.1 Control Algorithm Parameters

A number of system parameters determine the effectiveness of the control of the canine. The initial parameter values suggested here are the result of extensive field trials with the canines, observation of human-guided canine trials, and discussions with expert canine trainers. These parameters, presented in Table 6.3, will be referred to throughout the remainder of the control algorithm discussion.

Parameter	Units	Data Type	Value
MAX STOP ATTEMPTS	readings	integer	20
WAIT PACKETS	readings	integer	8
TURN WAIT PACKETS	readings	integer	2
MAX ANOMALIES	n/a	integer	2
MAX SUCCESS DISTANCE	meters	double	2.0
SPEED COEFFICIENT	n/a	double	0.75
SUCCESS CEILING	meters	double	10.0
STOP SPEED	cm/s	integer	30.0
LOOK BACK AMOUNT	readings	integer	4
TOLERANCE	n/a	double	0.25
COURSE TOLERANCE	degrees	integer	20
MAX CORRECTABLE DDC	degrees	integer	135
MINOR RSIDIST ERROR	readings	integer	3
MAJOR RSIDIST ERROR	readings	integer	6
MINOR BADCOURSE ERROR	readings	integer	3
MAJOR BADCOURSE ERROR	readings	integer	6

Table 6.3: Key control algorithm parameters and their current values.

In addition to the control algorithm parameters, there are a number of additional state variables. In particular there is a buffer of sensor readings and a list of ordered destinations associated with the current mission. At a minimum, a destination has a known latitude and longitude. The remaining state variables (such as buffer indices and variables used as flags for state transitions) are used for the internal workings of the implementation and are unnecessary to discuss to understand the control algorithm.

6.4.2 Control States and Transitions

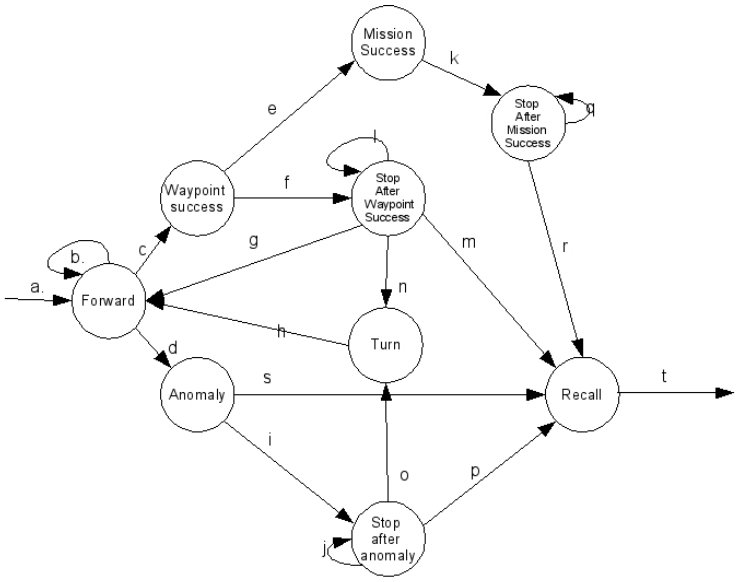
This section provides details on the key components of the control algorithm. A high-level view of the state diagram and state transitions is provided in Table 6.4. The states themselves were informed primarily by observing the distinct behaviors exhibited by the canine. The state transitions indicate the desired actuation of commands required in order to guide the canine to his next waypoint destination.

New Sensor Reading

Since the control algorithm is driven by receiving sensor readings, the controller performs this task before the other states (with the only exception being the Entry State). In other words, the data produced by a New Sensor Reading is what initiates the conditions for state transitions. After each sensor reading becomes available, the sensor data is updated and the following process begins:

1. Verify that the data has not been corrupted during radio transmission using simple heuristics. If it is corrupted, then discard this packet.

2. Update the derived metrics (see Section 6.3.2) based on the new sensor data.



Label	Source State	Condition
a	entry state	sensors okay, canine at start position
b	Forward	not anomaly, not waypoint success
c	Forward	canine is close enough to waypoint to call success
d	Forward	anomaly condition detected
e	Waypoint Success	waypoint success for the last waypoint in mission
f	Waypoint Success	canine is issued stop command
g	Waypoint Stop	next command is calculated to be forward
h	Turn	after Left or Right is issued for WAIT PACKETS readings
i	Anomaly	Stop command is issued
j	Anomaly Stop	canine speed is greater than STOP SPEED
k	Mission Success	Stop command is issued
l	Waypoint Stop	canine speed is greater than STOP SPEED
m	Waypoint Stop	canine failed to stop in MAX STOP ATTEMPTS readings
n	Waypoint Stop	canine speed is less than STOP SPEED
o	Anomaly Stop	canine speed is less than STOP SPEED
p	Anomaly Stop	canine failed to stop in MAX STOP ATTEMPTS readings
q	Mission Stop	canine speed is greater than STOP SPEED
r	Mission Stop	canine speed is less than STOP SPEED
s	Anomaly	detected anomalies exceeds MAX ANOMALIES
t	exit state	mission complete and canine is at start position

Table 6.4: A state diagram and transition table describing the control algorithm at a high level of abstraction.

3. Verify that the current command being issued by the embedded system is the desired current command. If it is not, then the correct command will be reissued immediately.

4. Buffer the new packet for use in determining state transitions.

5. Calculate and issue new command.

Entry State

The operator verifies that the vest and sensor pack are safely attached to the canine and that the canine is ready to begin. The waypoints have been marked and added into the waypoint list (the "path"), the canine is positioned at the start waypoint by the operator. The control algorithm waits for the GPS system to obtain sufficient high quality satellite signals in order to have a horizontal accuracy measure better than 10 m. If this level of accuracy cannot be obtained, the control algorithm waits. Typically waiting is only necessary if the system has just been powered on and a sound GPS lock is still being acquired. This process typically takes no more than 45 s from power on. Once the trial begins, the system transitions to the Forward state.

Forward State

During the Forward State, the canine hears the continuous Forward tone indicating that he is moving in the correct direction toward his next destination. Two possibilities exist for exiting the Forward state: arriving at the current destination waypoint or triggering the anomaly detection algorithm.

If the canine arrives sufficiently near his current destination waypoint, this state transitions to the Waypoint Success state. "Sufficiently near" is defined by the condition described in Equation (6.1) :

$$D < (MAXSUCCESSDISTANCE) + HACC + (SPEEDCOEFFICIENT) * gSpeed \tag{6.1}$$

In this equation, MAX SUCCESS DISTANCE (m) and SPEED COEFFICIENT are the control system parameters, HACC is the horizontal accuracy (m), and gSpeed is the current ground speed (m/s). The rationale for allowing what essentially constitutes a radius of success around the target is that in an operational setting it would not be necessary to direct the canine exactly to a searchable target. A trained canine brought within a relatively small distance (several meters) of a target would be able to search automatically at that distance. Including the HACC component allows for some additional tolerance due to sensor noise. Finally, the gSpeed component causes the canine to be stopped farther away from the target as his speed increases. The intended purpose of this is to prevent the canine from passing by the desired waypoint when he is moving very quickly.

The other possibility for exiting the Forward state is triggering the anomaly detection algorithm (discussed in Section 6.5), which will cause a transition to the Anomaly state. At the beginning of the Forward command, anomaly detection will not be performed for the first WAIT PACKETS readings in order to guarantee that sufficient velocity is in the system for course measurements to be accurate. The exception to this is if the Forward is following a Turn, TURN WAIT PACKETS (a much smaller number of readings) will be

used instead. This scheme is referred to as "rapid correction" for the remainder of the text. The rationale for this is described later in Section 6.5.

Waypoint Success

This state indicates that the canine has successfully arrived at a waypoint. Provided that there are more destinations to continue to, the current destination waypoint will now be updated to the next destination in the path. The control algorithm will transition to Stop After Waypoint Success. If the canine has arrived at his final waypoint, the control algorithm will transition to the Mission Success.

Mission Success

This state indicates that the canine has successfully arrived at the final waypoint. The next destination will be set to the starting location, but instead of normal directional commands, the controller will move into Recall. The canine's response to the Recall command is so consistently correct that directional commands are not necessary for a recall. This could change if missions were to become very long and complicated, such that the canine could no longer navigate on his own back to the starting location. In this case, a new mission with a reversed path could be initiated in order to autonomously return the canine to the original starting location.

Waypoint Stop

Following a Waypoint Success, the canine is issued the Stop tone. The control algorithm will remain in the Stop state until the canine's measured speed (cm/s) falls below the STOP SPEED parameter (cm/s). The reason that there is a speed threshold is to account for

sensor error (i.e. even when the canine is still, there is still a very small speed measured by GPS). Prior to issuing the Stop command, the control algorithm "freezes" the values of DDC and DDC2, since the GPS receiver depends on the components of velocity in order to calculate course. When velocity becomes very small, the GPS course measurements become inaccurate. Since the control algorithm always needs to calculate the next command from a stop state, these course-based metrics must be saved from before course becomes inaccurate.

If the canine fails to stop in MAX STOP ATTEMPTS readings, the control algorithm transitions to Recall. In practice, the canine responds extremely well to the Stop command, so this is very seldom necessary. When the canine responds correctly to the Stop command, the control algorithm transitions to the Turn state.

Anomaly Stop

Anomaly Stop operates in much the same way as Waypoint Stop, but also includes a logical condition to check if the number of total detected anomalies in this trial exceeds MAX ANOMALIES. If this occurs, the canine is recalled and the mission is unsuccessful. Otherwise, once the canine stops, the controller transitions to Turn.

Mission Success Stop

Mission Success Stop operates in much the same way as the Waypoint Stop. Once the canine stops, the control state transitions to Recall with the mission marked as successful.

Turn

Following a waypoint success, the canine will need to be issued a new command indicating the general direction of the next destination. In some cases, the next destination

waypoint is generally forward, in which case the control algorithm will transition to Forward. The Turn state can also be reached following an anomaly detection, but the result of Turn will always be either a Left or Right turn (if Forward was the correct command, there would not have been an anomaly). Calculating the command is described by the set of rules describe in Algorithm 2.

Algorithm 2 Calculating a New Command from a Stop

```
if WAYPOINTSUCCESS then
  if DDC2 ≤ 20.0 then
    command ← FORWARD
  else
    if DDC2 ≥ 0.0 then
      command ← LEFT
    else
      command ← RIGHT
    end if
  end if
else
  if DDC2 ≥ 0.0 then
    command ← LEFT
  else
    command ← RIGHT
  end if
end if
```

All directional commands (Forward, Left, or Right) are held for a number of readings equal to WAIT PACKETS and then the control algorithm transitions to the Forward state. To be clear, issuing a turn is a two step process: first issuing and holding the Left or Right vibration (for WAIT PACKETS readings) and then issuing the Forward command.

Anomaly

This state indicates that the canine has deviated from a correct course in such a way as to require stop and correction. A discussion on Anomaly Detection is given in Section 6.5.

After the detection of an Anomaly, a stop is issued and the control algorithm transitions into Anomaly Stop.

Exit

Once the controller issues the recall and the canine arrives back near the start position, the algorithm terminates.

6.5 Anomaly Detection and Anomaly Correction

A considerable amount of effort was spent on the Anomaly Detection problem using a neural network approach based on the encouraging results of a literature review and initial experimental results (as discussed in Chapter 4). However, several discoveries were made regarding the nature of anomaly detection and anomaly correction as the project progressed. In addition, between the Training Phase I and Training Phase II, hardware changes relating to the command of the canine made the original decision to use a neural network less reasonable.

The most obvious change was that the actuation mechanisms for the canine had become considerably more fine-grained by Training Phase II. In Training Phase I, the canine was controlled simply by indicating a reinforcing tone or the absence of one. The control mechanisms was strictly binary in nature. In Training Phase II, there were five commands (Forward, Stop, Left, Right, Recall) giving more direct control over the canine. Increasing the number of possible classifications increases the complexity for a neural network. This increased complexity could possibly be overcome with an extremely large number of canine trials to use as training data. Essentially, for the neural network to be able to learn this

new and more complex rules set, it would need a set of a full range of demonstrable canine behaviors (both typical and anomalous). This is similar to the problem with neural networks identified and discussed in [Moafipoor et al. 2008], in which the authors attempt to model pedestrian motion characteristics using a neural network. Experimentally, it is difficult to coerce the canine into demonstrating specific types of anomaly behavior and would come with the added hindrance that the canine would in effect be trained to demonstrate such behaviors more often in the future! An additional problem is that neural networks are difficult to "tune" manually, even if expertise is available (such as the canine trainer).

However, another problem emerged from field trials: the issue of false positives (incorrectly classifying correct behavior as an anomaly, which would lead to corrections). False positives have a high penalty in terms of mission success since they lead to stopping the canine and issuing a correction (which may then lead to more anomalous behavior). The results in Chapter 4 were fairly promising, but even if it were possible to adjust these algorithms to eliminate false positives (perhaps through implementing filters which required multiple firings of the network), it would come at the expense of not being able to quickly identify actual anomalies. Additionally, using a neural network as an anomaly detection mechanism runs the risk of violating the goal of never punishing the canine for correct behavior, or, with filtering, runs the risk of not being able to identify anomalous behavior quickly enough to react in time.

Ultimately, a rules-based approach which takes advantage of the "memory" created through the buffering of the sensor's data proved to be very effective at quickly identifying anomalies without significant false positives. Further, this approach is computationally

inexpensive (compact enough to eventually be implemented on an embedded device) in comparison to the Neural Network and can be tuned more readily.

6.5.1 Anomaly Detection Using a Rules-Based Approach

The two derived metrics used to determine anomalies were RSIDISTANCE and BADCOURSE. The rationale behind RSIDISTANCE is simply that if D is steadily increasing, the canine is moving away from the destination waypoint. If BADCOURSE becomes large, this simply means that canine is consistently deviating from a course that will take him to the destination waypoint. There is a built-in tolerance to allow for small deviations caused by the canine's normal movements and for small obstacle avoidance. In other words, a series of readings will never be called anomalous unless at least MINOR RSIDIST ERROR and MINOR BADCOURSE ERROR are exceeded. An anomaly can be triggered either when both metrics begin to indicate a trend away from the waypoint or if one metric grows very large. The algorithm conditions are shown in Algorithm 3.

Algorithm 3 Rules for determining if the canine is in an anomaly state based on trends in distance from target and deviations from desired course.

if $RSIDISTANCE > MINORRSIDISTERROR \&\& BADCOURSE > MINORBADCOURSEERROR$ **then**
  $anomaly \leftarrow TRUE$
end if
if $RSIDISTANCE > MAJORRSIDISTERROR || BADCOURSE > MAJORBADCOURSEERROR$ **then**
  $anomaly \leftarrow TRUE$
end if

6.5.2 Anomaly Correction

Detecting anomalies does not contribute to improved mission success unless a recovery can be made in those instances. There are two mechanisms for anomaly correction. Both strategies are described in mechanical detail in the control algorithm description, but this section highlights the rationale for these approaches.

The most common type of anomaly is the canine not taking a Right or Left turn command due to some object unobservable to the sensors. This results from the canine's desire and training to search promising targets which supersedes his training to listen to the commands issued by the tone generator. This is discussed in greater length in Chapter 5. In these cases, the desirable anomaly correction is a very quick stop and an issuing of a new directional command. This is in contrast to the typical anomaly detection/correction approach which gives some leeway during a Forward command. Following a Left or Right command, the amount of time allowed before anomalies can be detected is very small and an immediate correction follows. In practice, if the canine fails to make a turn and then is quickly stopped and told to make the turn again, he will often reverse course. If the canine is allowed to progress toward an incorrect target, many turns will be required to correct his course (there is no "Back" command to indicate "turn around") and these corrections will either be very slow or simply ignored after several are issued.

The alternate "correction" technique is to simply recall the canine and restart the mission. For repeated detected anomalies, the control algorithm recalls to the starting position.

6.6 Experimental Setup

6.6.1 Initial Validation of the Control Algorithm

Prior to any experiments with a live canine, two different simple simulation strategies were undertaken in order to guarantee that the algorithm would behave appropriately on the canine.

Simulation Using Recorded Canine Trials

In these tests, sensor data collected from live canine trials (discussed in Chapter 5) was used instead of real-time sensor data. A data file containing sensor measurements was passed in to the control algorithm. The results from the control algorithm were logged and inspected for a wide variety of trials to guarantee that the control algorithm was stable and could appropriately issue all the commands necessary to guide the canine. These simulations were not particularly useful for parameterizing the control algorithm at this stage since the "simulated" canine was not not reacting to the commands being issued by the control algorithm in this case. Rather, the sensor data being fed in corresponded to the canine's reactions to human commands - not the control algorithm commands. Nonetheless, this helped provide system testing to correct semantic errors in the software.

Simulations in Hardware Using the Author as a Canine

Once the software was considered stable enough from software simulations to test in real-time on the hardware, the components were placed in a small plastic tray (without the physical vest used for the canine) and hand-carried by the author in a marked field under simulated mission scenarios. This served to identify some basic parameter changes

and additional logical conditions. An attempt was made to exercise all of the commands in many different ways. Once there was confidence that the control algorithm was correct under normal conditions, anomalies were added to guarantee their detection and appropriate correction.

Once these tests were completed, the control algorithm in hardware was demonstrated for the expert trainers to gain their feedback. Again, their recommendations were invaluable in making modifications to parameters and improving the algorithm. Specifically, they provided suggestions to increase the waypoint stop distance and to minimize the anomaly detection time following turns (since the canine tended to only respond to corrections issued quickly in these circumstances). Once all of these phases of testing were complete, the system was deployed to autonomously guide the canine himself.

6.6.2 Live Canine Trial Setup

All of these trials were performed under the direct supervision of a qualified canine training professional and under the same regulations described in Chapter 5. Given the nature of these trials, it is not possible (or even desirable) to hold all of the conditions constant (for example, repeated trials in the exact same area would lead to a bias in the canine since he would learn the waypoints). However, every attempt was made to perform similar types of trials as the ones performed in the human-guided experiments. A combination of simple (one point) and complex (multiple point) trials were performed over the course of two months on four different trial dates. Environments were both open fields and areas with sparse and dense buildings.

6.7 Results and Discussion

6.7.1 Preliminary Blind Trials

This section describes a small set of trials performed to guarantee that the canine could be guided to a waypoint without either an initial course toward the first waypoint or without any distinguishing characteristics of the waypoint itself (i.e. no physical marker). Particularly, these trials demonstrate the feasibility of guiding a canine to an arbitrary point in a featureless landscape. This is significant because the trials demonstrate the most basic capability to direct the canine to a location using only the tones and vibrations produced by the control system. In order to perform this trial set, the canine was taken to a wide open field with no observable objects (i.e. a plain, flat field). Three distinct waypoints in the middle of the field were measured with software (but not physically marked). The control algorithm was used with parameters as described previously, but allowing a total of four corrections (MAXANOMALIES) instead of the usual two. The canine was initially oriented such that his course was straight ahead (but not toward any of the waypoints) and then the control algorithm issued corrections appropriately. In each trial, a different point was chosen such that no two trials had the same waypoint as the goal. The key metric in this case is how close the canine came to arriving at an arbitrary waypoint.

In Table 6.5, the results of these five preliminary trials are shown. The "Initial Distance" column describes how far away the canine was at the beginning of the trial from the arbitrary waypoint. The "Closest Distance" column describes how close he arrived to that waypoint before being recalled due to reaching the maximum number of corrections. The "Corrections" column indicates how many times the canine was stopped and issued a new turn. Finally, the "Strict Success" column simply indicates whether this trial would have

been considered a successful mission following the same standards applied to traditional trials with physical waypoints.

Trial Number	Initial Dist. (m)	Closest Dist. (m)	Corrections Issued	Strict Success
1	47.95	15.67	4	N
2	56.75	5.72	1	Y
3	68.46	6.95	4	Y
4	48.13	10.79	4	N
5	67.70	28.36	4	N
avg.	57.80	13.50	3.4	40%

Table 6.5: Blind Field Trial Result Summary

These trials are more difficult for the canine since he does not have any objects to view to add meaning to the Left or Right commands. Additionally, since there is nothing to search, nothing "interests" the canine. With that said, the data shows the ability to get relatively close (average of 13.5 m) to arbitrary unmarked waypoints. Trial 2 is shown in Figure 6.1. In order to achieve this distance, it usually required all of the available corrections (average 3.4 corrections issued). Under the same terms of success applied to human-guided trials (under 7m to the destination waypoint) where there were physical waypoints, 40% of these blind trials would have been considered successful.

6.7.2 Live Canine Trial Results and Discussion

The control algorithm was held to the same standard of success or failure as described in Section 5.5. In determining statistical performance, the p-values reported in this section are the results of a two-tailed Fisher's test, with $p < 0.05$ being considered statistically significant. The results of repeated trials with the canine are shown in Table 6.6. The trials run on 05-08-09 were run without rapid correction (described in Section 6.4.2) following turns. This improvement was included in all future trial results.

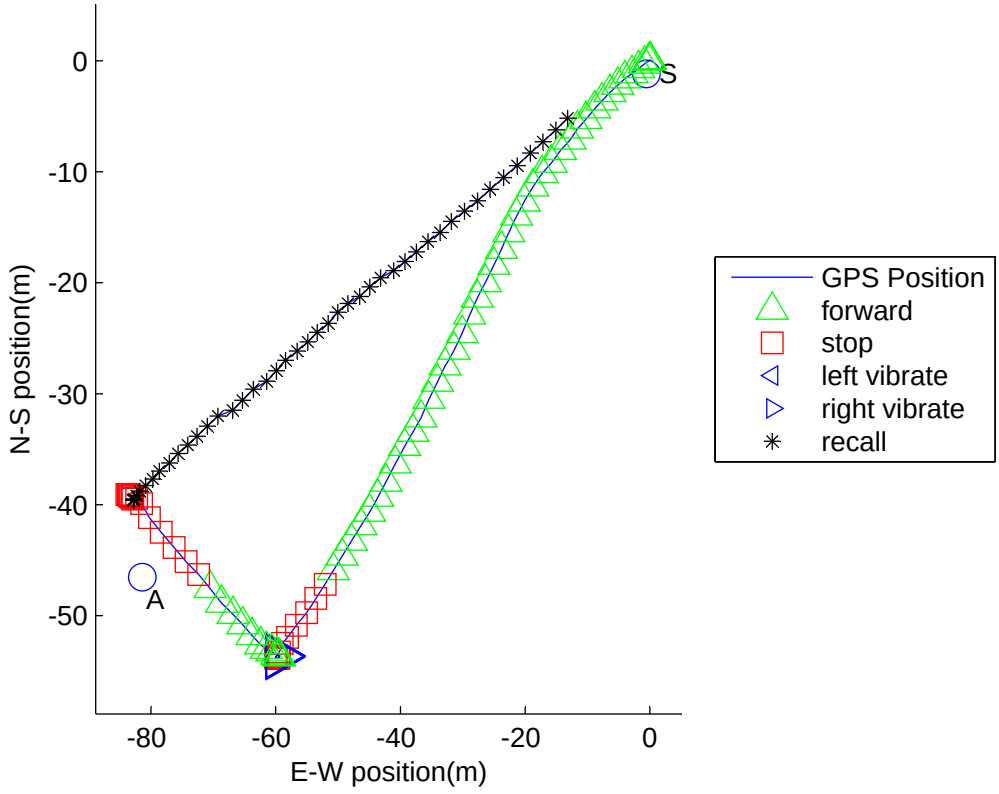


Figure 6.1: This plot shows the canine being autonomously commanded to an arbitrary, unmarked point in the middle of an open field. The canine is commanded forward until he begins to deviate from the desired course too significantly. He is stopped, issued a Right, and then arrives near A. He is stopped and then recalled since he arrived within 7 m of the desired waypoint. This is both a command success and a canine success.

Date	Total	1PT S.	1PT F.	1PT SR	MPT S.	MPT F.	MPT SR
05-08-09							
field	10	4	1	0.80	3	2	0.60
dense buildings	5	1	1	0.50	1	2	0.33
05-14-09							
field	8	4	0	1.00	2	2	0.50
sparse buildings	3	0	0	n/a	2	1	0.66
05-21-09							
sparse buildings	5	1	0	1.00	2	2	0.50
dense buildings	6	1	2	0.33	3	0	1.0
06-17-09							
sparse buildings	4	0	0	n/a	3	1	0.75
Summary	41	11	4	0.73	16	10	0.62

Table 6.6: A Breakdown of Single Point and Multi-Point Trial results taken over a two month period using the Autonomous Canine Control Algorithm.

At the present, these results indicate sufficient repetition to demonstrate the feasibility of autonomously controlling the canine. Although the control algorithm's success rate on single point trials is slightly higher than that achieved by the human operator (73% vs. 71%), this result does not indicate a statistically significant difference ($p = 1.0$) between the trial sets. Similarly, while the results achieved on complex trials are lower (62% vs. 63%), these results also do not demonstrate a statistically significant difference ($p = 1.0$). It should be noted that the goal of this research is not necessarily to outperform (or even equal) the performance of the human operator, but rather to establish the feasibility of this approach for the autonomous control of a canine and to identify means for system refinement. Hence, results that are not statistically significantly lower are sufficient. An example of a successful simple trial is shown in Figure 6.2, a multi-point trial is shown in Figure 6.3, a multi-point trial containing an anomaly detection and attempted anomaly correction cycle is shown in Figure 6.4.

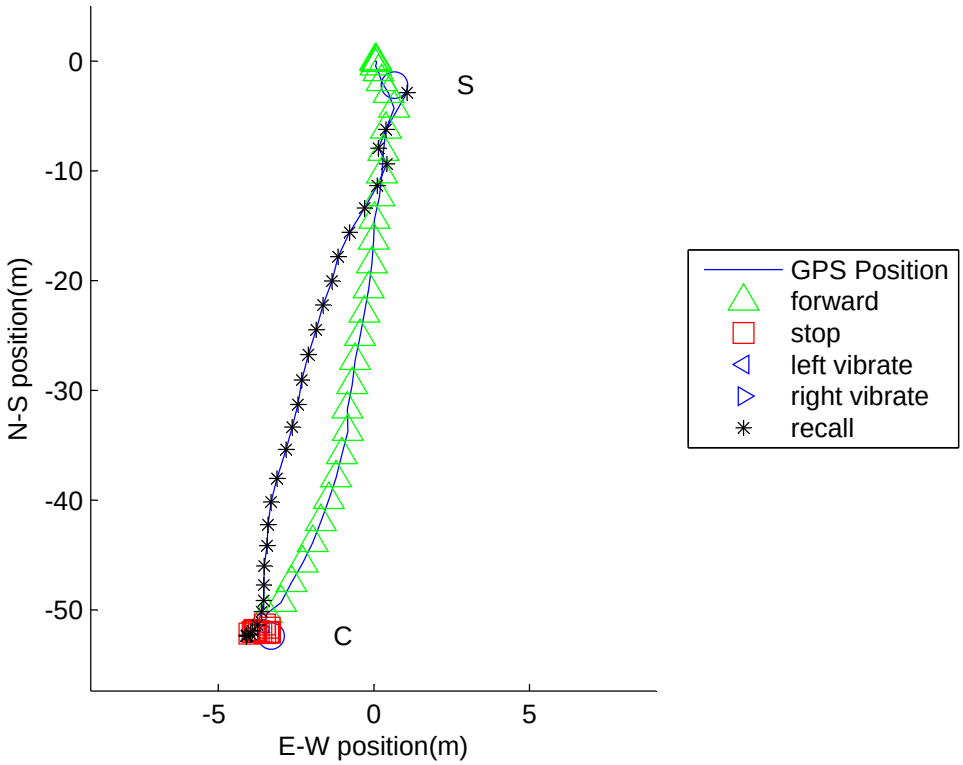


Figure 6.2: A successful single-point canine trial. The control algorithm issues Forward until the waypoint success in proximity to waypoint C, then the mission is completed and a Recall is issued.

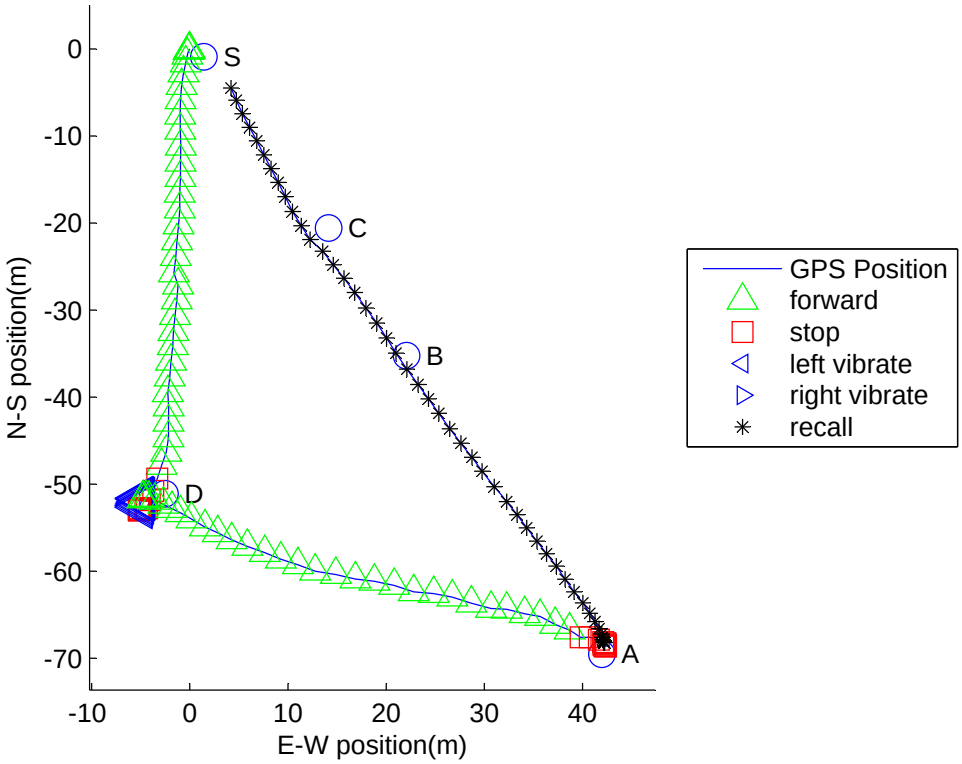


Figure 6.3: A successful multi-point canine trial. The control algorithm issues Forward until the waypoint success in proximity to waypoint D, then stops the canine and reorients him using the Left command. The canine is then commanded Forward until the waypoint success at A. The mission is completed and a Recall is issued. In this trial, waypoints B and C serve as "foil" waypoints.

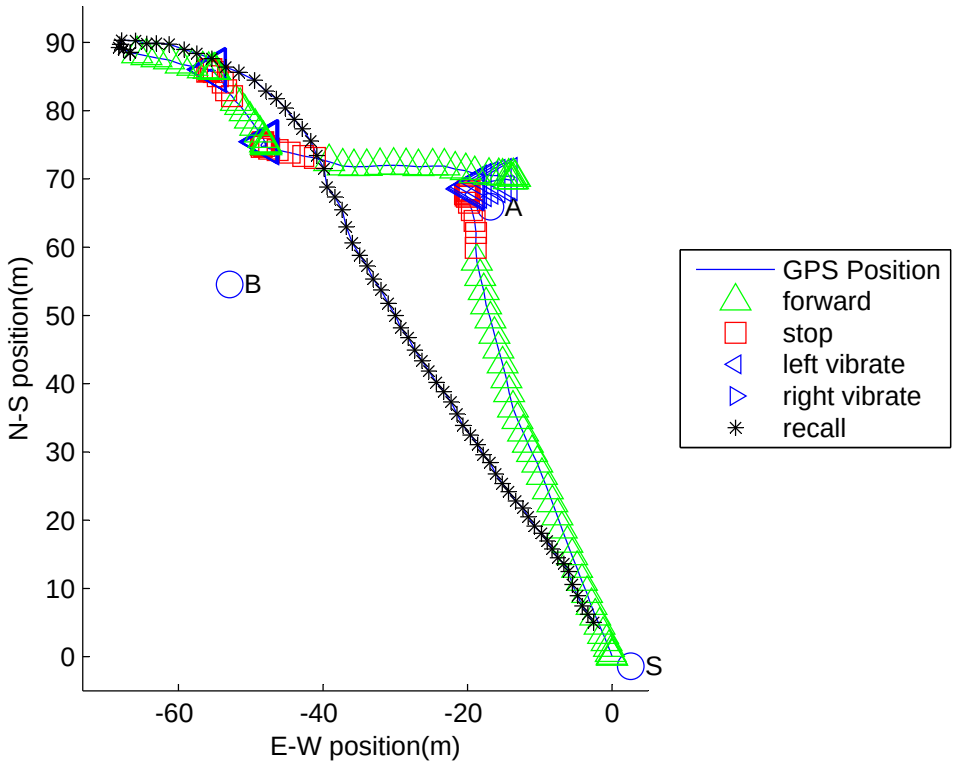


Figure 6.4: A multi-point trial that would be considered a canine failure, but a control algorithm success. The control algorithm issues Forward until the waypoint success in proximity to waypoint A, then issues a Left command to move the canine toward B. The canine begins to deviate from B after approximately 35 m triggering anomaly detection. The canine is stopped and issued a Left. The canine ignores the Left, a new anomaly is quickly detected and the canine is stopped a second time. The canine continues to ignore the issued Left command and the control algorithm Recalls due to the anomaly threshold being reached.

With respect to the control algorithm, all trials were logged and the control algorithm was verified to guarantee that observed successes were within the appropriate proximity to the waypoints and that all commands issued were correct. There are only two circumstances in which the system itself had errors leading to mission failure. The first was in a very hilly environment where the start position and the end position had roughly 100 m of distance and a large hill between them. This led to a radio failure with the Xbee Radio modems and the canine had to be recalled. A second issue was that on rare occasion, the canine would stop at a waypoint and then suddenly change his yaw (heading) without significant velocity. In this case, the canine's velocity was near zero (meaning course estimates from GPS were inaccurate). Recall that the canine's course measurements are saved at the beginning of a Stop, causing the incorrect tone (a Left instead of a Right, for example) to be issued. An example of this behavior along with a successful correction is shown in Figure 6.5. Otherwise, the control algorithm appropriately issued commands.

With respect to environment, the canine tends to move toward "searchable" targets (buildings, cars, etc.) rather than arbitrary and artificial waypoints (flags, cones, etc.). This presents a problem in dense environments where searchable targets like buildings may distract the canine from the desired waypoint.

6.7.3 Scenarios in Which the Autonomous Control Algorithm Overcomes Limitations of a Human Operator

The previous trial sets focused on situations in which the human operator and the control algorithm were tested in circumstances favorable to successful control by the human. In other words, the human had the full advantage of line of sight with the canine. However,

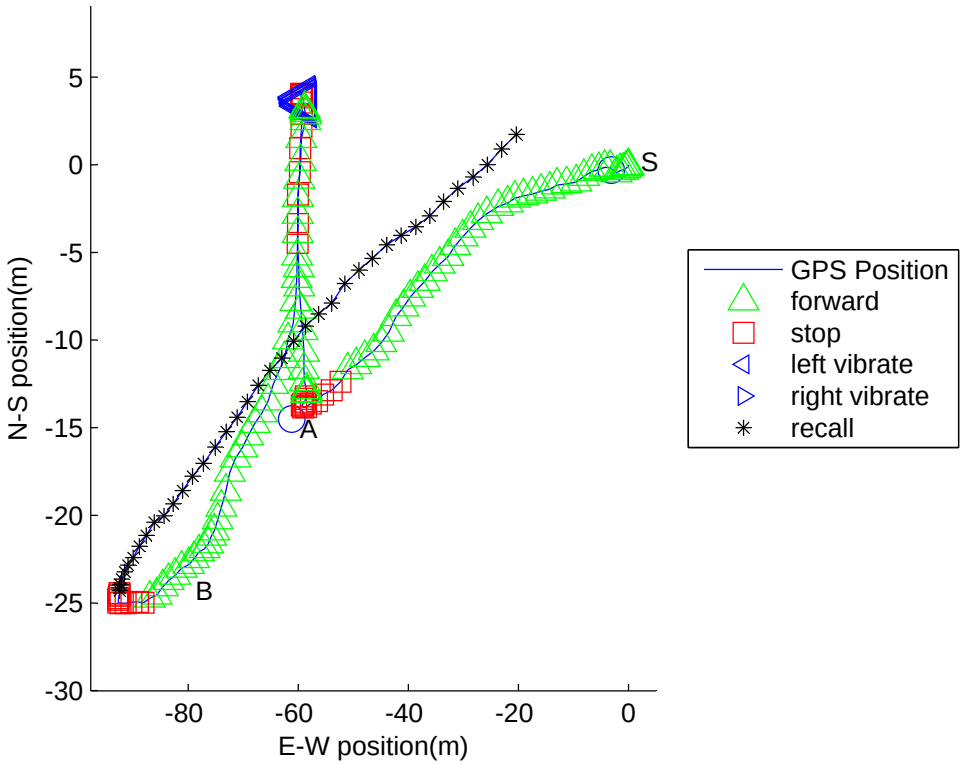


Figure 6.5: A multi-point trial that would be considered a canine success and a control algorithm success. The control algorithm issues Forward until the waypoint success in proximity to waypoint A. However, after stopping the canine turns to his right to face a nearby shed. GPS does not detect this change in course, so the algorithm issues a Forward to attempt to move the canine toward waypoint B. The canine follows the Forward command, creating an anomaly which is detected. The canine is stopped and then commanded Left. The canine takes the correction and begins moving to his left, eventually reaching waypoint B. He is recalled successfully.

a significant contribution of the system would be to overcome the need to have the operator within light of sight. The scenarios illustrated here are some simple examples where the autonomous control algorithm can succeed using sensors when a human operator would need to resort to guessing since they would be guiding blind.

For example, in both Figure 6.6 and Figure 6.7, the ultimate destination is out of the line of sight of the canine operator due to large structures. In the Figure 6.6, the control algorithm successfully commands the canine to waypoint A, stops the canine, then issues a Right. The canine successfully arrives near waypoint B, is stopped, and then a Recall is issued. Since the trial is performed in an open air environment, GPS is available and the sensors have no difficulty. For very long distance trials of this type or for trials where there are significant physical obstructions, a more powerful radio than the Xbee Radio Modem would be required. In the Figure 6.7, the control algorithm successfully commands the canine to waypoint A, stops the canine, then issues a Left. The canine moves out of line of sight of the starting position. The canine successfully arrives near waypoint B, is stopped, and then a Recall is issued. Again, the human being would be forced to resort to "guessing" to guide the canine in this instance or would need to physically move around the building, which could be undesirable in operating environments.

In a third trial (see Figure 6.8), the canine was autonomously guided around the perimeter of a building. The canine is commanded to waypoint A, issued a Stop followed by a Left command. The canine moves toward waypoint B and then is stopped again rapidly. Another Left command is issued and the canine continues toward C. The canine is stopped and recalled. In this case, the canine saw the starting position from waypoint C and chose

to take the most direct route (alongside the remaining side of the building rather than retracing his previous path).

The key result of this section is to identify some scenarios where the autonomous canine control actually exceeds the sensory limitations of a human handler.

6.7.4 Canine Behavioral Changes Over Time

An important realization regarding the trials performed is that the autonomous canine system consists of both equipment and the canine himself. While system improvements are being made, the canine is also learning how to respond effectively to the system. Since the canine is rewarded after successful trials with petting or the opportunity to play with the toy, he is conditioned to follow the instructions of the autonomous system. One anecdotal observation from field trials is that canine began to develop a better response to the commands Left and Right after being exposed to the autonomous canine system more frequently. In other words, human guidance often takes advantage of the canine's posturing and heading in a way that the autonomous canine system does not. The human may simply watch and wait until the canine is facing the correct direction and then issue a Forward (rather than relying solely on the Left and Right commands). The autonomous canine system does not operate in this fashion, since it precomputes turns using the DDC2 metric. However, over time, the canine began to respond better to the autonomous canine directional commands even though they had slightly different meanings than they did in the human guided trials. Ultimately, continued consistent training with the autonomous system would likely yield higher performance in terms of mission success.

6.7.5 Suggestions for Improvement

Given the specific issues encountered in the experimental trials, the following potential solutions are proposed:

1. In order to address radio connectivity issues, use a more powerful radio than the Xbee Radio Modems. There are many more powerful radios that would have extended range and robustness in the face of geography. However, this comes with a penalty in terms of the amount of weight the canine must carry and in the battery life of the command pack.

2. In order to improve the canine's response to commands, use a dedicated canine rather than a canine trained for multiple tasks. This would improve the accuracy of the canine's response to commands. This comes with the disadvantage of requiring a dedicated canine, which is an expensive proposition. Further, if those other trained skills (search, for example) are desirable, then multiple training modes would still be required. A possible solution would be to integrate additional stimuli (such as mild electrical stimulation) into the system to improve anomaly correction capabilities.

3. In order to deal with the situation where the canine changes heading when velocity is near zero (and thus course becomes undetectable to GPS), additional sensors such as the XSens demonstrated in Chapter 5 could be read and aggregated in real-time rather than in offline simulations. The downside of this approach is additional weight on the canine and the added complexity of interfacing additional software and hardware components.

6.8 Chapter Conclusion

This chapter provides a detailed description and demonstration of an algorithm to automate the guidance of a trained canine. The algorithms approach to issuing commands, handling sensor data, detecting and correcting anomalies, and its response to failure conditions are explained. The control algorithm is integrated with an autonomous canine system and then its feasibility and performance are demonstrated through repeated field trials with a trained canine. An analysis of these results has been provided as well as some suggestions for future improvements.

Figure 6.6: A successful autonomous canine trial in which portions of the trial take place out of the line of sight of the canine operator. Although the perspective of the Google Earth imagery makes the canine's path appear to go "inside" the structure, he is actually passing around the side of the building.

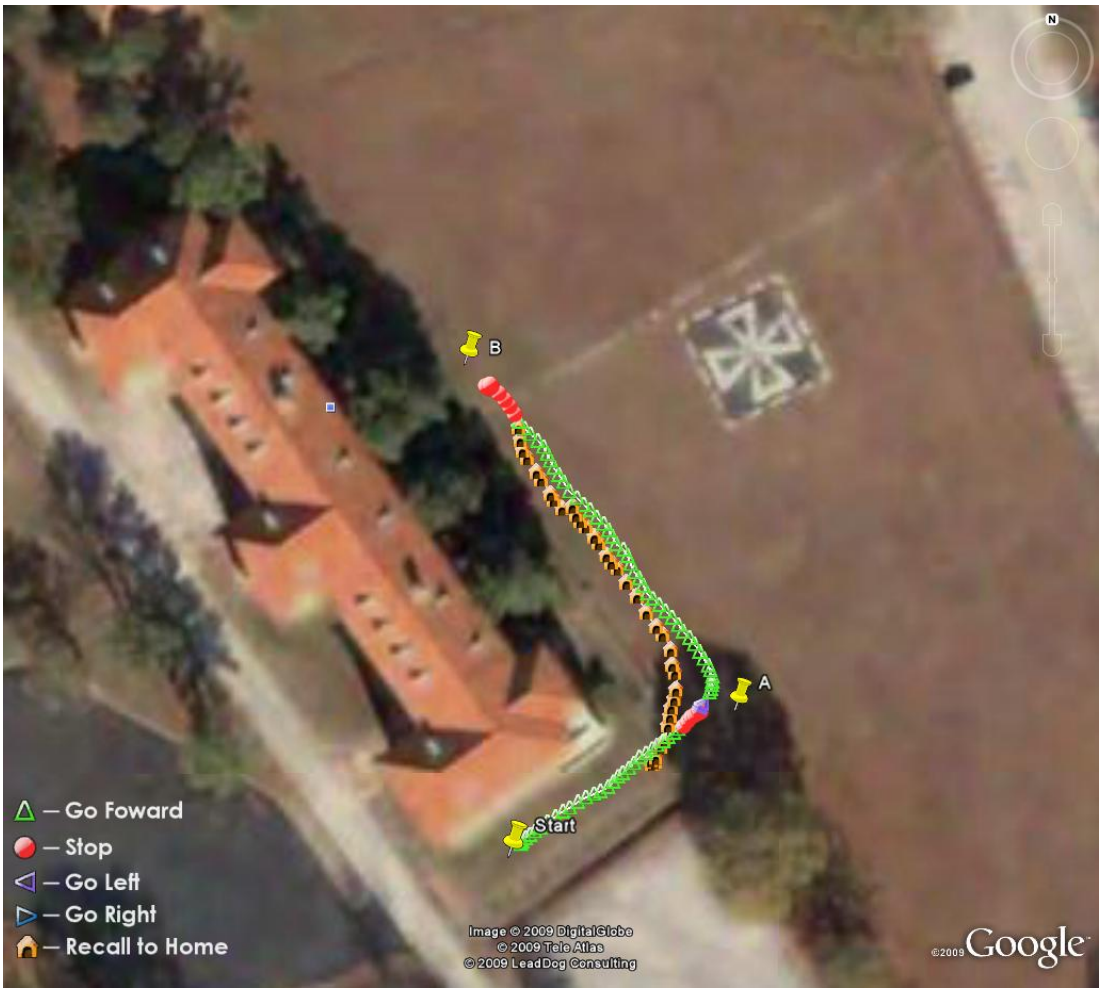


Figure 6.7: A successful autonomous canine trial in which most of the trial take place out of the line of sight of the canine handler.

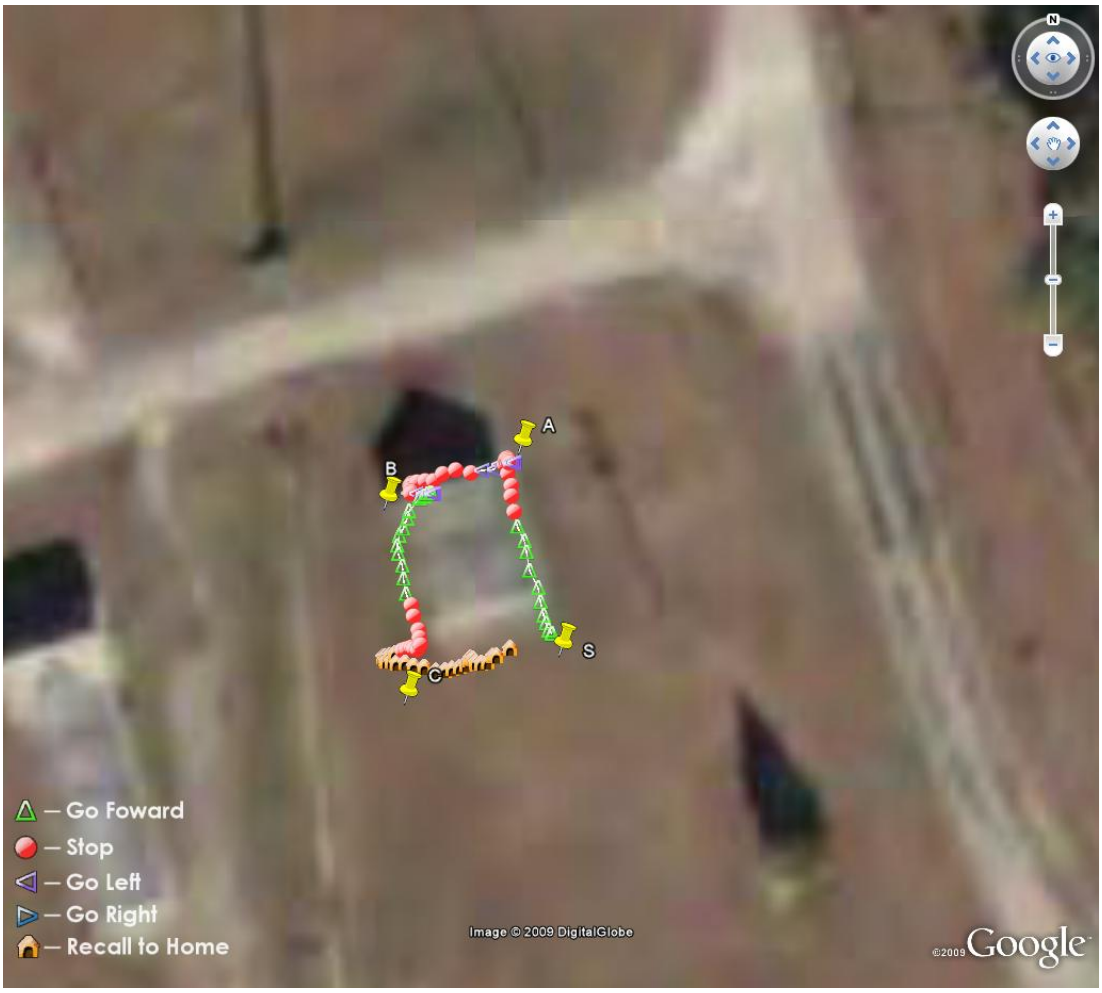


Figure 6.8: A successful autonomous canine trial in which the canine is guided around the perimeter of a building. Most of the trial occurs out of line of sight for the canine handler.

Chapter 7

Conclusion

This chapter provides the key accomplishments of the dissertation, some concluding remarks, and some directions for future work in autonomous guidance.

7.1 Key Contributions

- Applied evolutionary algorithms to optimize General Regression Neural Networks for the modeling of the binary tone guidance of a trained canine.
- Developed and interfaced hardware with GPS Satellite Receivers and Inertial Measurement Unit sensors to gather data on the canine's position, velocity, acceleration, and heading. Developed software to extract that information in a usable form to an external source (a laptop).
- Demonstrated the ability to command and navigate a canine under the guidance of a human operator for the purpose of extending the operational range of a human trainer and to provide training data to inform a model of human guidance.
- Provided a means to record and visualize canine trials showing both commands issued and sensor measurements in an intuitive format (annotated Google Earth displays).
- Implemented a state machine model in order to autonomously guide a canine on multiple waypoint missions and verified that system through field trials.
- Demonstrated through a significant number of field trials with a canine the effectiveness of a strategy for the autonomous command and navigation of a trained canine.

7.2 Concluding Remarks

The key goal of this work was to demonstrate the autonomous command of a trained canine in missions of one or more waypoints. Pursuant to this goal, a multi-disciplinary effort was undertaken to bring together the components needed for an autonomous canine solution. Specifically, a canine was trained to respond to tone and vibration directional commands. A command module was built and programmed that could perform navigation for the canine and transmit that data to a data sink where it could be recorded, visualized, and analyzed. This command module was rigorously tested in conjunction with the trained canine in order to gauge both the effectiveness of the sensors in canine tracking and to analyze the accuracy of the trained canine in following commands.

Human-guided canine trial data was used to develop both an evolved Machine Learning approach to canine Anomaly Detection and a state machine approach. Ultimately, the rules-based approach was chosen due to its advantages with respect to false positives, efficiency, and intuitive design. A state machine control algorithm was developed and tested extensively with a variety of canine simulation techniques. This algorithm included both canine anomaly detection and correction capabilities. The control algorithm was then integrated into the command module and field tested with the trained canine in repeated field trials in order to not merely establish feasibility of this approach, but to demonstrate its capability and effectiveness. Non-trivial autonomous canine missions were accomplished in a variety of environments with reasonably consistent results.

This system developed in this dissertation provides a foundation for autonomous canine units to be used with limited to no supervision in a wide variety of environments for bomb and narcotic detection, reconnaissance, and training. Law enforcement officers, the armed

forces, and travel security officers all stand to benefit from trained, autonomous canine units. Further, this work will pave the way for future non-invasive autonomous animal applications. Although this dissertation has shown the first automated guidance of a trained canine, there are many opportunities for future improvements. Some of these are described in the next section.

7.3 Future Work

This section describes several areas for future work within the autonomous canine guidance system. Suggestions for future work include, but are certainly not limited to:

- The system presented here could be made more robust with real-time sensor filtering techniques which could eliminate some of the weaknesses associated with GPS (as discussed in Chapter 6).

- Given more canines with similar training, the algorithm could be tested on multiple canines in order to evaluate its robustness and ability to generalize. Given enough canines, the original ML approach might become practical since the ability to collect large amounts of training data could be satisfied. Similarly, if other types of animals could be trained to respond to non-invasive stimuli in this fashion, autonomous control could be extended to animals other than canines.

- This system could be combined with other actuators (such as mild electrical stimuli) to improve canine response and mission success. Additionally, the system could be adapted to automate the training of canines by adding positive and negative stimuli as a reaction to sensed canine behavior.

- Improved wireless radios could extend the range of the system from the home base.
- The state machine control algorithm could be housed on the embedded system instead of the laptop to allow for autonomous missions outside of radio range. This would allow for autonomous missions outside of radio range (even with minimal, primarily syntactic, changes to the control algorithm itself).
- Analysis could be performed on the state machine to guarantee stability in scenarios that may not have been considered in this dissertation.
- Canine pose analysis using the sensors described in Chapter 5 could be used to perform "Detection Detection" or the automatic ability to recognize when the canine has detected something of interest by recognizing changes in canine pose.
- Evolutionary Algorithms could be used to discover control algorithm parameters by automatically analyzing large numbers of human guided canine trials.
- Additional investigation could be performed in using the directional commands in different ways in order to get better mission performance. For example, in situations where the canine repeatedly ignores Left and Right commands, a partial recall could be issued to "reset" the canine before continuing the trial.
- A control algorithm could be developed which adapts to changing canine behavior over time by changing system parameters as the canine learns from the autonomous control system. An interesting comparison would be to see if such a dynamic control algorithm would outperform a static algorithm.

Bibliography

Araujo, R. and de Almeida, A. T. [1999], 'Learning Sensor-Based Nagivation of a Real Mobile Robot in Unknown Worlds', *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* **29**(2), 164 – 178.

Back, T., Hammel, U. and Schwefel, H. [1997], 'Evolutionary Computation: Comments on the History and Current State', *IEEE Transactions on Evolutionary Computation* **1**(1).

Baluja, S. [1996], 'Evolution of an Artificial Neural Network Based Autonomous Land Vehicle Controller', *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* **26**(3).

Barnard, E. [1992], 'Optimization for Training Neural Networks', *IEEE Transactions on Neural Networks* **3**, 232–240.

Bevly, D. M. [1999], Evaluation of a Blended Dead Reckoning and Carrier Phase Differential GPS System for Control of an Off-Road Vehicle, *in* 'ION GPS', pp. 2061–2069.

Bevly, D. M. [2004], 'Global Positioning System (GPS): A Low-Cost Velocity Sensor for Correcting Inertial Sensor Errors on Ground Vehicles', *Journal of Dynamic Systems, Measurement, and Control* **126**, 255–264.

Billings, S. and Zheng, G. [1995], 'Radial Basis Function Network Configuration Using Genetic Algorithms', *Neural Networks* pp. 877 – 890.

Britt, W., Bevly, D. M. and Dozier, G. [2008], Automated Modeling of the Guidance of a K-9, *in* 'Proceedings of the American Control Conference', pp. 2467–2474.

Britt, W., Cunningham, H. and Dozier, G. [2006], A Comparison of Evolutionary Protocols for Solving Distributed Constraint Satisfaction Problems, *in* 'IEEE Congress on Evolutionary Computation'.

Britt, W., Gopalaswamy, S., Hamilton, J. A., Dozier, G. and Chang, K. [2007], Computer Defense Using Artificial Intelligence, *in* 'Proceedings of SpringSim 2007: Symposium on Simulation Software Security', Vol. 2, pp. 2201–2207.

Brown, S. [2006], 'Stealth Sharks to Patrol the High Seas', *New Scientist* pp. 30–31.

Bureau of Diplomatic Security [2004], 'A Nose for Trouble: How the State Department Uses Bomb Detection Dogs', Web Citation. www.state.gov/m/ds/rls/33087.htm.

Burges, C. J. [1998], 'A Tutorial on Support Vector Machines for Pattern Recognition', *Data Mining and Knowledge Discovery* **2**, 121–167.

Calvert, D., Bajar, E., Stacey, D. and Thomason, J. [2003], Analysis of Equine Gait Through Strain Measurement, *in* 'Proceedings of the 25th International Conference of Engineering in Medicine and Biology Society', Vol. 3, pp. 2370–2373.

Chen, S., Cowan, C. F. N. and Grant, P. M. [1991], 'Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks', *IEEE Transactions on Neural Networks* **2**, 302–309.

Chowdhury, F., Wahi, P., Raina, R. and Kaminedi, S. [2001], A Survey of Neural Networks Applications in Automatic Control, *in* 'Proceedings of the 33rd Southeastern Symposium on System Theory', pp. 349–353.

Cook, D., Ragsdale, C. and Major, R. [2000], 'Combining a Neural Network with a Genetic Algorithm for Process Parameter Optimization', *Engineering Applications of Artificial Intelligence* **13**, 391–396.

Cornou, C. and Lundbye-Christensen, S. [2008], 'Classifying Sows' Activity Types from Acceleration Patterns: An Application of the Multi-Process Kalman Filter', *Applied Animal Behavior Science* **111**, 262–273.

Correll, N., Schwager, M. and Rus, D. [2008], Social Control of Herd Animals by Integration of Artificially Controlled Congeners, *in* 'Proc. of the 10th International Conference on Simulation of Adaptive Behavior (SAB)', pp. 437–447.

Cortes, C. and Vapnik, V. [1995], 'Support Vector Networks', *Machine Learning* **20**, 273–297.

Curtis, P. and Cupp, D. [1989], *Dogs on the Case: Search Dogs Who Help Save Lives and Enforce the Law*, Lodestar Books.

Cymbalyuk, G. S., Borisyuk, R. M., M'ueller-Wilm, U. and Cruse, H. [1998], 'Oscillatory Network Controlling Six-Legged Locomotion. Optimization of Model Parameters', *Neural Networks* **11**, 1449–1460.

Daily, R. and Bevly, D. [2004], 'The use of gps for vehicle stability control systems', *IEEE Transactions on Industrial Electronics* **51**(2), 270–277.

Davis, I. L. [1995], Sensor Fusion for Autonomous Outdoor Nagivation Using Neural Networks, Technical Report CMU-RI-TR-95-05, The Robotics Institute at Carnegie Mellon University.

Davis, L. [1991], *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

Dean, P. and Porrill, J. [1998], 'Pseudo-Inverse Control in Biological Systems: A Learning Mechanism for Fixation Stability', *Neural Networks* **11**, 1205–1218.

Dozier, G., Britt, W., SanSoucie, M., Hull, P., Tinker, M., Unger, R., Bancroft, S., Moeller, T. and Rooney, D. [2006], 'Evolving High-Performance Evolutionary Computations for Space Vehicle Design', *IEEE Congress on Evolutionary Computation* pp. 2201 – 2207.

Dozier, G., Cunningham, H., Britt, W., Wang, Y., Seals, C. and Zhang, F. [2007], 'Distributed Constraint Satisfaction, Restricted Recombination, and Genetic Protocols', *Applied Soft Computing* (3), 1005–1011.

Engelbrecht, A. [2002], *Computational Intelligence*, John Wiley & Sons, Ltd, West Sussex, England.

Ferentinos, K. P. [2005], 'Biological Engineering Applications of Feedforward Neural Networks Designed and Parameterized by Genetic Algorithms', *Neural Networks* **18**, 934–950.

Ferworn, A., Sadeghian, A., Barnum, K., Ostrom, D., Rahnama, H. and Woungang, I. [2007], Rubble search with canine augmentation technology, pp. 1–6.

Fierro, R. and Lewis, F. L. [1998], 'Control of a Nonholonomic Mobile Robot Using Neural Networks', *IEEE Transactions on Neural Networks* **9**, 589–600.

Fix, E. and Hodges, J. [1951], Discriminatory Analysis - Nonparametric Discrimation: Consistency Properties, Technical Report 21-49-004, USAF School of Aviation Medicine.

Floreano, D. and Mondada, F. [1998], 'Evolutionary Neurocontrollers for Autonomous Mobile Robots', *Neural Networks* **11**, 1461–1478.

Fogel, D. [1995], *Toward a New Philosophy of Machine Intelligence*, IEEE Press.

Forrest, S. and Mitchell, M. [1993], Relative Building-Block Fitness and the Building-Block Hypothesis, *in* L. D. Whitley, ed., 'Foundations of Genetic Algorithms 2', Morgan Kaufmann, San Mateo, CA, pp. 109–126.

Gabaglio, V. [2003], GPS/INS Integration for Pedestrian Navigation, PhD thesis, Institute of Geomatics of the Swiss Federal Institute of Technologye in Lausanne.

Gillette, R. L. and Angle, T. C. [2008], 'Recent Developments in Canine Locomotor Analysis: A Review', *The Veterinary Journal* **178**, 165–176.

Glasius, R., Komoda, A. and Gielen, S. C. A. M. [1994], 'Neural Network Dynamics for Path Planning and Obstacle Avoidance', *Neural Networks* **8**(1).

Godha, S. [2006], Performance Evaluation of Low Cost MEMS-Based IMU Integrated with GPS for Land Vehicle Navigation Application, Master's thesis, University of Calgary.

Golberg, D. [1989], *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Gomes, W., Perez, D. and Catipovic, J. [2006], 'Autonomous Shark Tag with Neural Reading and Stimulation Capability for Open-Ocean Experiments', *Eos Transactions, American Geophysical Union* **87**(36).

Han, J., Moraga, C. and Sinne, S. [1996], 'Optimization of Feedforward Neural Networks', *Engineering Applications of Artificial Intelligence* pp. 109 – 119.

Haykin, S. [1999], *Neural Networks: A Comprehensive Foundation*, 2nd edn, Prentice Hall, Upper Saddle River, New Jersey.

Helweg, D., Roitblat, H. and Nachtigall, P. [1993], Using a Biomimetric Neural Net to Model Dolphin Echolocation, *in* 'Proceedings of the First New Zealand International Conference on Two-Stream Artificial Neural Networks and Expert Systems', pp. 247–251.

Hill, A., Slamka, A., Morton, Y., Miller, M. and Campbell, J. [2007], A Real-Time Position, Velocity, and Physiological Monitoring and Tracking Device for Equestian and Race Training, *in* 'Proceedings of the ION GNSS'.

Holland, J. [1975], *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press.

Islam, M. M. and Murase, K. [2005], 'Chaotic Dynamics of a Behavior-Based Miniature Mobile Robot: Effects of Environment and Control Structure', *Neural Networks* **18**, 123–144.

Joachims, T. [1999], *Making Large-Scale SVM Learning Practical: Advances in Kernel Methods - Support Vector Learning*, MIT Press.

Joachims, T. [2005], A Support Vector Method for Multivariate Performance Measures, *in* 'Proceedings of the International Conference on Machine Learning'.

Joachims, T. [2006], Training SVMs in Linear Time, *in* 'Proceedings of the ACM Conference on Knowledge Discovery and Data Mining'.

Kalkkhul, J., Hunt, K. J. and Fritz, H. [1999], 'FEM-Based Neural-Network Approach to Nonlinear Modeling with Application to Longitudinal Vehicle Dynamics Control', *IEEE Transactions on Neural Networks* **10**(4).

Kohavi, R. and John, G. [1997], 'Wrappers for Feature Subset Selection', *Artificial Intelligence* **97**(1-2), 273 – 324.

Kohonen, T. [1989], *Self-Organization and Associative Memory*, 3 edn, Springer-Verlag.

Kohonen, T. [1997], *Self-Organizing Maps, Second Edition*, Springer.

Koza, J. [1992], *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press.

Ladetto, O., Gabaglio, V. and Merminod, B. [2001], Combining Gyroscopes, Magnetic Compass and GPS for Pedestrian Navigation, *in* 'Proceedings of the International Symposium on Kinematic Systems in Geodesy, Geomatics, and Navigation', pp. 205–213.

Leung, F. H. F., Lam, H. K., Ling, S. H. and Tam, P. K. S. [2003], 'Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm', *IEEE Transactions on Neural Networks* **14**(1), 79 – 88.

Ma, Z.-Q. and Yuan, Z.-R. [1995], 'Real-time Nagivation and Obstacle Avoidance Based on Grids Method for Fast Mobile Robots', *Engineering Applications of Artificial Intelligence* **8**(1).

Mahadevan, S. and Theocharous, G. [1998], 'Rapid Concept Learning for Mobile Robots', *Machine Learning* **31**, 7 – 27.

Maniezzo, V. [1994], 'Genetic Evolution of the Topology and Weight Distribution of Neural Networks', *IEEE Transactions on Neural Networks* pp. 39 – 53.

Meyer, D., Leisch, F. and Hornik, K. [2003], 'The Support Vector Machine Under Test', *Neurocomputing* **55**, 169–186.

Miller, G. F., Todd, P. M. and Hegde, S. U. [1991], 'Designing Neural Networks', *Neural Networks* **4**, 53–60.

Miller, J. and Bevly, D. M. [2007], Position and Orientation Determination for a Guided K-9, *in* 'Proceedings of the 20th International Technical Meeting of the Satellite Division of the Institute of Navigation ION GNSS 2007', pp. 1768 – 1776.

Miller, J. and Bevly, D. M. [2009], Guided K-9 Tracking Improvements Using GPS, INS, and Magnetometers, *in* 'Proceedings of the ION ITM'.

Moafipoor, S., Grejner-Brzezinska, D. A. and Toth, C. K. [2008], 'A Fuzzy Dead Reckoning Algorithm for a Personal Navigator', *Navigation: The Journal of the Institute of Navigation* **55**(4), 241–254.

Moody, J. and Darken, C. J. [1989], 'Fast Learning in Networks of Locally-Tuned Processing Units', *Neural Computation* **1**, 289–303.

Moreau, M., Siebert, S., Buerkert, A. and Schlecht, E. [2009], 'Use of a tri-axial accelerometer for automated recording and classification of goats' grazing behaviour', *Applied Animal Behaviour Science* **In Press, Corrected Proof**, –.
URL: *http://www.sciencedirect.com/science/article/B6T48-4WBB6TP-1/2/af33be442803a939109fbd72dfdb505c*

Mudra, R. and Douglas, R. J. [2003], 'Self-correction Mechanism for Path Integration in a Modular Nagivation System on the Basis of an Egocentric Spatial Map', *Neural Networks* **16**, 1373–1388.

Muggleton, S. [1991], 'Inductive Logic Programming', *New Generation Computing* **8**(4), 295–318.

Quinlan, J. [1993], *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Rabbit Semiconductor [2008], 'Rabbit Products', World Wide Web electronic publication.
URL: *{http://www.rabbit.com/products/}*

Ribeiro, C., Ferworn, A., Denko, M., Tran, J. and Mawson, C. [2008], Wireless estimation of canine pose for search and rescue, pp. 1–6.

Rosenblatt, F. [1957], The Perceptron: A Perceiving and Recognizing Automaton, Technical Report 85-460-1, Cornell Aeronautical Laboratory.

Rumelhart, D., Hinton, G. and Williams, R. [1986], *Parallel Distributed Processing*, MIT Press, Cambridge MA.

Russell, S. and Norvig, P. [2003], *Artificial Intelligence: A Modern Approach*, 2nd edn, Prentice Hall, Upper Saddle River, NJ.

Ryu, J. and Gerdes, J. C. [2004], 'Integrating Inertial Sensors with GPS for Vehicle Control', *Journal of Dynamic Systems, Measurement, and Control* **126**, 243–254.

Schwager, M., Detweiler, C., Vasilescu, I., Anderson, D. M. and Rus, D. [2008], 'Data-Driven Identification of Group Dynamics for Motion Prediction and Control', *Journal of Field Robotics* **25**(6-7), 305–324.

Shandong University of Science and Technology [2007], 'SDUST Created Remote-Controlled Pigeon', World Wide Web electronic publication.
URL: *{http://www.sdkd.net.cn/en/news_show.php?id=65}*

Shinchi, T., Tabuse, M., Todaku, A., Tokushige, Y. and Kitazoe, T. [2001], Evolutionary Robot Simulations with Competitive-Cooperative Neural Network and Adaptive Synaptic Couplings, *in* 'IEEE International Workshop on Robot and Human Interactive Communication', pp. 280–285.

Simmons, J. A., Saillant, P. A., Wotton, J. M., Haresign, T., Ferragamo, M. J. and Moss, C. F. [1995], 'Composition of Biosonar Images for Target Recognition by Echolocating Bats', *Neural Networks* **8**(7/8).

Song, W., Chai, J., Han, T. and Yuan, K. [2006], 'A Remote Controlled Multimode Microstimulator for Freely Moving Animals', *Acta Physiologica Sinica* **58**(2), 183–188.

Spears, W., Back, K. A., Fogel, D. B. and de Garis, H. [1993], An Overview of Evolutionary Computation, *in* 'The Proceedings of the European Conference on Machine Learning', pp. 442–459.

Specht, D. [1991], 'A General Regression Neural Network', *IEEE Transactions on Neural Networks* **2**, 568–576.

Sporns, O. and Alexander, W. H. [2002], 'Neuromodulation and Plasticity in an Autonomous Robot', *Neural Networks* **15**, 761–774.

Stengel, R. [1994], *Optimal Control and Estimation*, Dover Publications.

Talwar, S. K., Xu, S., Hawley, E. S., Weiss, S. A., Moxon, K. A. and Chaplin, J. K. [2002], 'Rat Navigation Guided by Remote Control', *Nature* **417**(May 2), 37–38.

Tani, J. and Fukumura, N. [1996], 'Self-Organizing Internal Representation in Learning of Nagivation: A Physical Experiment by the Mobile Robot YAMABICO', *Neural Networks* **10**(1).

Tao, K. [1993], A Closer Look at the Radial Basis Function Network, *in* 'Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers', pp. 401–405.

u-blox: Support, Tools and Help: FAQ : About GPS [2008], u-blox FAQ.

URL: *{http://www.u-blox.com/customersupport/faq_generic/distance.html}*

United Nations University [2001], 'Researchers Develop Robo-Roach', *UNU-MERIT: I&T Weekly* **7**.

URL: *{http://www.merit.unu.edu/i&tweekly/&tweekly_previous.php?issue=0107&issue_show=7&year=2001}*

Upadhyay, T., Cotterhill, S. and Deaton, A. [2003], 'Autonomous GPS/INS Navigation Experiment for Space Transfer Vehicle', *IEEE Transactions on Aerospace and Electronic Systems* **29**, 772–785.

van de Ven, P. W., Flanagan, C. and Toal, D. [2005], 'Neural Network Control of Underwater Vehicles', *Engineering Applications of Artificial Intelligence* **18**, 533–547.

van Heijst, J. J., Vos, J. E. and Bullock, D. [1998], 'Development in a Biologically Inspired Spinal Neural Network for Movement Control', *Neural Networks* **11**, 1305 – 1316.

van Rijsbergen, C. J. [1979], *Information Retrieval*, Butterworth.

Vapnik, V. [1995], *The Nature of Statistical Learning Theory*, Springer-Verlag.

Vapnik, V. [1998], *Statistical Learning Theory*, John Wiley and Sons, Inc.

Waggoner, P. [2009], The Training of a Canine to Perform Search and to Respond to Audio and Vibration Commands. unpublished manuscript.

Watanabe, S., Izawa, M., Kato, A., Ropert-Coudert, Y. and Naito, Y. [2005], 'A New Technique for Monitoring the Detailed Behaviour of Terrestial Animals: A Case Study with the Domestic Cat', *Applied Animal Behavior Science* **94**, 117–131.

Whitehead, B. and Choate, T. [1996], 'Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction', *IEEE Transactions on Neural Networks* **7**(4), 869 – 880.

Widrow, B. and Hoff, M. E. [1960], Adaptive Switching Circuits, *in* 'WESCON Convention', Institute of Radio Engineers, New York, pp. 96 – 104.

Williams, E. [2008].

URL: *{http://williams.best.vwh.net/avform.htm}*

Wolpert, D. and Kawato, M. [1998], 'Multiple Paired Forward and Inverse Models for Motor Control', *Neural Networks* **11**, 1317–1329.

XSens [2009], 'XSens 3D Motion Tracking', World Wide Web electronic publication.

URL: *http://www.xsens.com/en/general/mti*

Xu, X. and He, H.-G. [2002], Neural-Network-Based Learning Control for the High-Speed Path Tracking of Unmanned Ground Vehicles, *in* 'Proceedings of the 2002 International Conference on Machine Learning and Cybernetics', Vol. 3, pp. 1652–1656.

Yang, S. X. and Meng, M. Q. [2003], 'Real-Time Collision-Free Motion Planning of a Mobile Robot Using a Neural Dynamics-Based Approach', *IEEE Transactions and Neural Networks* **14**, 1541–1552.

Yao, X. [1992], A Review of Evolutionary Artificial Neural Networks, Technical report, Commonwealth Scientific and Industrial Research Organization, Victoria, Australia.

Ye, J. [2008], 'Adaptive Control of Nonlinear PID-based Analog Neural Networks for a Nonholonomic Mobile Robot', *Neurocomputing* **71**, 1561–1565.