MACHINE LEARNING APPLIED TO HUMAN MOTION AND BEHAVIOR ANALYSIS

by

Richard J. Parker

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematical and Computer Sciences).

Golden, Colorado

Date November 15, 2004

Signed: Richard J. Parker
Richard J. Parker

Approved: Michael D. Colagrosso
Dr. Michael D. Colagrosso
Thesis Advisor

Golden, Colorado

Date Nov 16, 2004

for DP Newstead
Dr. Graeme Fairweather
Professor and Head
Department of Mathematical and
Computer Sciences

ABSTRACT

Understanding behavior from motion is simple for people to do, yet the procedure is difficult to codify. It is impossible to enumerate all possible motion patterns which could occur, nor to dictate the explicit behavioral meaning of each motion. We apply machine learning to the classification of motion into behavioral labels. Issues include segmentation of motion tracks, feature selection, and classifier construction.

The process of segmentation consists of converting motion tracks to segments of uniform length. We examine the impact of segment length selection on other areas of the experiment. Segment length impacts the number of data points which are used, the density and dimensionality of the feature space, and fluctuations in validation error rates. Selection of segment length must balance these considerations.

To counter the loss of data as segment size increases, a window of information from segments adjacent to a given segment is implemented and evaluated. For segments which are insufficiently long to utilize this window, values are left missing. Thus we gain the benefit of the greater information content of longer tracks while shifting the task of handling short tracks to a missing value problem.

Once the segment length is determined, construction of machine learning models to classify behavioral labels from motion tracks is possible. Models are constructed to classify human motion based solely on features describing the motion itself. We examine both motion-based classifications and behavioral classifications.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

Mike Colagrosso, for providing the initial spark that started me into machine learning and bountiful opportunities to apply my interest in the field, and for the direction and patience necessary to serve as my advisor for two years.

Alan Norton and Bill Hoff, for inviting me to assist with the research project which motivated this thesis work. Support for the research project is provided through University IRAD funding from Lockheed Martin Corporation

Desi, for inspiration, support, devotion, and love, without whom this thesis may never have been completed.

Our Father in Heaven, for granting inspiration and guidance in answer to the many prayers offered over the course of this project.

CHAPTER 1: HUMAN MOTION AND BEHAVIOR

Understanding behavior from motion is simple for people to do, yet the procedure is difficult to codify. It is impossible to enumerate all possible motion patterns which could occur, nor to dictate the explicit meaning of each motion. Even more difficult is assignment of behavioral labels to a given motion track. We apply machine learning to the classification of motion into behavioral labels. Issues include feature selection, segmentation of motion tracks, and classifier construction.

1.1 Fingerprinting

Everyone has unique characteristics that set them apart from others. Some of these are hereditary, such as hair color or height. Others develop as a result of the random growth patterns of our cells, as is the case with our fingerprints. Fingerprinting is the art of identifying key attributes or aspects from an information database which can be used to identify distinct conclusions about the source of the information. Often, selected attributes do indeed serve as keys opening doors to greater understanding regarding the individual or individuals about whom the information pertains.

Many techniques have been developed which are compared to the process of fingerprinting. Some examples include handwriting analysis, speech recognition, and retinal scans. Less robust methods include face recognition and gait recognition. Gait recognition consists of analyzing key features about how a person walks which are distinctive or unique, which brings us to the field of analyzing motion of a person to identify them.

Much research has been performed on motion of parts of objects, as in gait recognition and gesture analysis. Noticeably less effort has been exerted on analysis of

the overall motion of an object. For example, how a person travels from point A to point B can provide insight into who the person is or the underlying activities of the person. Methods which identify patterns and behaviors of motion are applicable in areas beyond human motion analysis. The methodologies we explore in analyzing motion are applied specifically to human motion and group behaviors.

We now examine in more depth the work of Johansson in modeling biological motion, as it provides a stark parallel to our work.

1.2 Biological Motion Analysis

In 1973, Johansson performed experiments in which the motion of key points of a human body was captured, and analysis of the motion alone was sufficient to determine both individual point labels and the overall activity of the person (1973). This is analogous to our work, though at a simpler level.

The work of Johansson explores visual perception of biological motion, with specific application to human motion (1973). The work provides a strong foundation for this thesis in that Johansson explores interpretation and understanding of an object based on the motion of the key components. We move to a more abstract level, attempting to identify behaviors of entire groups of objects based on their overall motion patterns.

Johansson extracted motion tracks of the joints of a person, stating that "the motion tracks of the main joints were chosen as representative motion events" (1973). An example of this can be seen in Figure 1. The motion of the foot, ankle, wrist, hip, etc., is not only identifiable by their individual motion, but also contributes to a higher identification of the actions of the moving person. Johansson notes "the geometric structures of body motion patterns in man ... are determined by the construction of their skeletons" (1973). Further, "the time relations in the series of iterated pendulum motions as a rule are highly regular. This interaction brings about a body displacement with a rather constant speed" (1973).

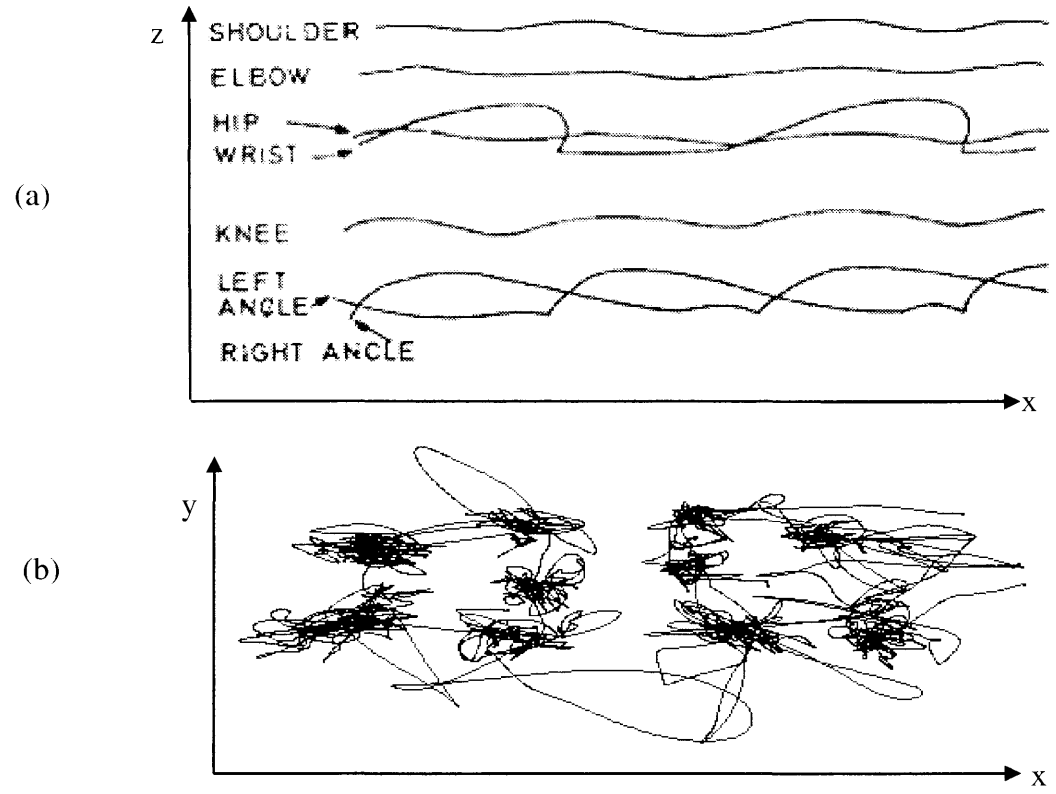

Figure 1: Comparison of biological motion to structured sports motion

(a) Johansson's figure depicting motion paths of key joints of a walking human (1973). Note that the motion of the wrist and ankle carry a distinct signature which is unique from the other tracks. The tracks in this figure are periodic, and knowledge of the structure of the human form assists in determining the underlying figure who generated the motion tracks. (b) We generalize the concepts of Johansson to more abstract motion track analysis. Our motion tracks are generated by the paths people take through a scene. As Johansson's motion tracks emit structure and order because of the underlying skeletal constraints imposed by the human form, so do motion tracks in general. The structure which emerges is due to the goals and limitations in place on the individuals in the scene.

More generally, the motion of a single element could be used to identify that element; the collective motion of a set of moving elements could also be used to identify the collection as a whole. As Johansson's motion tracks exhibited structure and periodicity due to the underlying structure and motion of the human skeleton which generated the motion (1973), our motion tracks also exhibit distinct features which we use to infer details regarding the generating structure. Thus, groups of moving objects can be analyzed.

We now move from the specific example of Johansson's work to discuss motion tracks in general.

1.3 Information Content of Motion Tracks

Biological motion analysis, though inspired by Johansson's experiments, has taken an intriguing route in influencing the efforts of other researchers since 1973. We review several of the projects which have strong bearing on our work. Individual projects are dealt with in greater depth in Chapter 2.

Much study has been done in areas related to analyzing periodic motion of objects (Polana & Nelson, 1994) and the motion of components of objects (gait—Niyogi & Adelson, 1994; gestures—Bobick & Ivanov, 1998). We focus on characterizing the motion of an object at a higher level, analyzing the path taken as the object travels through a scene. This constitutes a shift from the motion of sub-parts of an object to the overall motion of the object.

Few projects have focused directly on the overall motion tracks of objects in a scene. However, several other fields have comparable behavior-type analysis methods. In analyzing behavior of users on a network, the system administrator can potentially identify malicious users prior to their causing significant harm (Cannady, 1998). Similarly, new antivirus applications add monitoring of program activities to attempt to identify infected computers and limit their access to the rest of the network (Martin,

Nelson, & Sewani, n.d.). As an example of the information content of human traffic to the surveillance world, we note the following: "During the 1991 Gulf War: a pizza parlor near the Pentagon told newsmen that it could always tell when something was about to happen because large numbers of people stayed late at the Pentagon and ordered pizza" (Diffie & Landau, 1998).

From the field of computer vision, attempts to model human traffic yield promising results. One reason cited by Bennewitz, Burgard, and Thrun is that "as people move through their environment, they do not move randomly. Instead, they often engage in typical motion patterns" (2002). In other words, analysis of the motion patterns of objects may infer information regarding the causes which generate the motion.

Our work melds the analysis efforts of motion and behaviors with the flexibility and power of machine learning. In the next section, we review the motivational applications for this work.

1.4 Motivation

As discussed in the previous section, we integrate machine learning models with motion and behavior analysis techniques. We discuss here several applications for such technology.

We work toward developing a system which can be used for group activity recognition. Identification and classification of groups of objects based on their collective motion is a promising approach. This can best be described as behavioral classification of moving objects.

A primary motivating application of a behavioral classification system is video scanning. As a direct result of the ready availability of video capture systems, the quantity of stored video data is swiftly outpacing the resources necessary to process it. With our behavioral classification system, a user would identify an example of the activity which they wish to locate. The system would generate a model of the activity

and then locate intervals in the available video data where a matching motion fingerprint occurs. Results from the behavioral classification experiments of Chapter 5 show the strength of this approach.

Motion is often a simple metric to evaluate on target objects. In the absence of more concrete data which might be used to classify the object, motion classification is still possible. As is discussed in Chapter 2, sources of motion information are varied, from radar systems to GPS locators and even laser range-finders. In military applications, often motion detection systems (e.g., radar) are the first contact with an object. Rather than failing, a motion-based classification system could still perform.

Group behaviors are of great interest especially in view of recent terrorist activities. Security surveillance primarily consists of classifying a monitored person as either "normal" or "abnormal." Ideal systems identify dangerous individuals in a population before they engage in a dangerous act. Surveillance systems augmented by behavioral modeling tools would have improved ability at identifying such persons.

Current modeling techniques of activities and interactions between people are discussed in more depth in Chapter 2. The system described by Patterson, Liao, Fox, and Kautz analyzes human activities in a top-down manner (2003). A human counterpart defines the structure of a multi-person interaction, and the system works from the high-level activity definition down to the corresponding low-level events which serve as the activity fingerprint. Our goal is a system which performs in a bottom-up approach. Low-level events are recognized, and then proposed to a human user for verification of usefulness. The processes outlined herein may contribute toward such a system.

Applications drive the use of technological advances. Further uses of the processes outlined in this thesis are certain to arise. Rather than exhaustively enumerating them, we move to introduce the overall methodology of our work.

1.5 Methodology

The previous section outlines applications for the work of this thesis. We now fly through our approach, explaining at a high level how the various portions fit together.

The applications of Section 1.4 focus on motion and behavioral analysis uses. In quantifying the data contained by a single motion track, it would be ideal if the length of a track and the richness of the features representing said track were independent of each other. However, such is not the case. In reality, the quality of the features depends strongly on the length of the track. We discuss this problem in greater detail, and propose a solution to it in segmenting the motion tracks, in Chapter 5.

Our approach to motion and behavioral analysis consists of classifying human behaviors based solely on the motion itself. We develop machine learning models to classify features extracted from the motion track to a label. We show that the resulting low-level classifiers perform well across widely varying scenes. Also, we show that motion behavior analysis is improved by providing a wider window of information to each activity segment. The experiments are discussed in Chapters 5 and 6.

1.5.1 Problem Formalization

Formally, our problem is this. Moving objects can be tracked via many methods, producing a motion track t of each object of interest. In order to be of use in any of the applications outlined in Section 1.3, it is necessary that there be some way of classifying a single motion track based on how it moves. Furthermore, a collection of motion tracks obtained from a set of interacting, cooperating objects are classified based on characteristics of the motion of all the objects in the set. Thus, our problem breaks into two parts: (1) individual motion and behavior, and (2) group behavior.

1.5.2 Individual Motion and Behavior

Our goal is to identify individual behavior based on the motion properties of the individual. Let us assume we have a set of motion tracks which have had their behaviors identified and labeled. This problem now becomes a supervised learning problem, and consists of generating a model of the mapping from the motion properties to a behavioral label.

More formally, given a motion track $t = \langle p_1, p_2, p_3, \mathrm{K}, p_{|t|} \rangle$, where $p_j = (x, y)$, classify the motion of t with respect to a training set of motion tracks and their labels $T = \{(t_1, y_1), (t_2, y_2), \mathrm{K}, (t_{|T|}, y_{|T|})\}$, where t_i is a motion track defined as is t, and y_i is the class label.

We first identify activities within each motion track in the set. The set of activities is then segmented at a uniform activity segment length. Selected features are computed from the resulting activity segments. The resulting feature vectors are then classified by trained machine learning models, mapping an input track t to a label y. Chapter 4 outlines activity labeling and feature selection. Chapter 5 examines how the segment length is determined. The specifics of the model construction and evaluation are outlined in Chapter 6.

1.5.3 Future Work: Group Behavior

Our long-term goal is to identify group behavior based on the motion properties of the objects which make up the group. For convenience, we refer to a group of objects as a scene. As is discussed in Chapter 3 regarding machine learning theory, many of the machine learning algorithms depend on a reliable distance metric for comparison of the

input instances. Instances which are similar should have a small distance, while dissimilar instances should have a large distance.

It is anticipated that features extracted from the motion tracks of the objects in the group can be augmented with the output of the set of classifiers developed in this thesis.

1.6 Thesis Overview

This thesis is organized as follows. Chapter 2 provides background on related work being performed in the field of motion analysis. This establishes the framework for discussion of both motion tracks and group behavior. Chapter 3 describes the field of machine learning, and stands as the foundation for our approach to motion track analysis. Chapter 4 covers the preprocessing techniques used to prepare the motion track data of a scene for use by the machine learning tools, namely activity labeling and feature selection. Features from Chapter 4 are scaled by the size of the activity segments. Chapter 5 determines the manner in which segment lengths are selected. Chapter 6 covers the development and evaluation of low level motion track classifiers. Chapter 7 projects into future work spawned by this thesis.

CHAPTER 2: RELATED WORK

We provide an overview of the work which has been performed to date in the field of motion track analysis. This provides a backdrop for our work, and establishes the key terminologies necessary for adequate discussion of our project.

A motion track *t* consists of a sequence of points $p_i = (x_i, y_i)$, in our case residing in two-dimensional space, as follows:

$$t = \langle p_1, p_2, p_3, \mathrm{K}\ , p_{|t|} \rangle \tag{2.1}$$

where $|t|$ is the length of the sequence *t*, or the number of points in *t*. This chapter discusses work done by others in the field of analysis of motion tracks. Section 2.1 reviews the foundation of low level approaches. Sources of motion data and several applications are discussed in Section 2.2. Section 2.3 outlines research being performed specific to human motion and behavior analysis. Section 2.4 reviews approaches taken in modeling human motion tracks. Section 2.5 discusses related methods of analysis.

2.1 Applicability of Low Level Approaches

The work of Polana & Nelson stands as one of the first efforts in analysis of human motion by automated methods (1994). It constitutes a shift from attempting to identify a model of the human form to recognizing the activity of the person with very coarse information. The focus of the work was periodic activities, such as walking or

running, where a basic motion was repeated with a fairly steady frequency.[1] One of the most powerful ideas generated by the work of Polana and Nelson is that approaches which focus on low level features "apply equally well to any source of motion" (1994). Though the source of the motion tracks may change, methodologies of analysis which apply well to any source or method of acquiring the motion tracks should still apply.

Many works (Bobick & Ivanov, 1998; Ivanov, Stauffer, Bobick, & Grimson, 1999; Nevatia, Zhao, & Hongeng, 2003; Patterson, Liao, Fox, & Kautz, 2003) utilize low level sensor data as the preliminary inputs to the system. Methods for preprocessing the low level input data have been developed. These include event generation (Bobick & Ivanov, 1998; Ivanov, Stauffer, Bobick, & Grimson, 1999), where the event stream is then parsed into a parse tree according to a context-free grammar; extraction of features of interest, such as motion data and other properties (Nevatia, Zhao, & Hongeng, 2003); and combination of the low level data with human-generated rules to classify the underlying action (Patterson, Liao, Fox, & Kautz, 2003).

2.2 Motion Data Acquisition and Applications

Some of the impact of Polana and Nelson's work can be seen in the varied methods for acquiring motion track data, as well as the range of target applications for which methodologies are developed. Methods for acquiring motion data tend to fall into two categories: one where the moving objects provide their positional data and the other where the positional data is extracted from the moving objects. The first category primarily consists of mobile objects which are equipped with GPS sensors, generating a stream of positions (Patterson, Liao, Fox, & Kautz, 2003). More common are the approaches which fall into the second category, relying on the ability of the system to extract the positional data via some other medium. Some examples of this include video

[1] In a later work, Efros, Berg, Mori, and Malik note that this specific approach has its limits. "Methods based on periodicity are restricted to periodic motion" (2003).

surveillance (either from a single camera or through multiple cameras) (Aggarwal & Cai, 1999; Ben-Arie, Wang, Pandit, & Rajaram, 2002; Bobick & Ivanov, 1998; Efros, Berg, Mori, & Malik, 2003; Hongeng, Bremond, & Nevatia, 2000; Ivanov, Stauffer, Bobick, & Grimson, 1999; Mohan, Papageorgiou, & Poggio, 2001; Nevatia, Zhao, & Hongeng, 2003; Niyogi & Adelson, 1994; Polana & Nelson, 1994; Sato & Aggarwal, 2001a; Sato & Aggarwal, 2001b; Tan & De Silva, 2003; Tucakov & Ng, 1998; Viola, Jones, & Snow, 2003), laser range-finder systems (Bennewitz, Burgard, & Thrun, 2002; Cielniak, Bennewitz, & Burgard, 2003; Panangadan, Matarić, & Sukhatme, 2004a; Panangadan, Matarić, & Sukhatme, 2004b), and unmanned aerial vehicles (UAVs) (Haritaoglu, Harwood, & Davis, 1998). Proving the statement by Polana and Nelson about the source of the motion tracks being independent of the method for processing them (1994), Yan and Matarić validate their motion track processes using robot positioning systems and then test the motion track processes on human motion extracted from video (2002).

Often, the motion track source is chosen in part to coincide with the target application. For example, the GPS system of Patterson, Liao, Fox, and Kautz is intended for use as a "cognitive compass system" for travelers (2003). Many of the laser range-finder systems mount the range finders on mobile robots (Bennewitz, Burgard, & Thrun, 2002; Cielniak, Bennewitz, & Burgard, 2003) while others (Panangadan, Matarić, & Sukhatme, 2004a) seem to use the range finders to guarantee that no information can be gathered which could be used to identify the people who are tracked. Video applications often are used where the analysis technique targets the tracked motion of specific features of the human body (or other tracked objects). These include gait detection and recognition, where the motion of the feet and other portions of the human figure are identified as key aspects of the person's gait (Niyogi & Adelson, 1994); activity recognition, where such activities include sitting, standing, walking, etc., and are identified solely by tracking of the person's head (Tan & De Silva, 2003); human gesture recognition, experimenting on identifying rhythm conducting (Bobick & Ivanov, 1998).

2.3 Human Motion Analysis

While the works discussed in Section 2.2 can be applied to any motion data, many applications target human motion analysis. One reason for this concentration is the difficulty in explicitly modeling human motion. Patterns in human motion exist. Bennewitz, Burgard, and Thrun note that "as people move through their environments, they do not move randomly. Instead, they often engage in typical motion patterns" (2002). Thus, for a given scene, certain motion patterns become much more likely. These patterns are identified and modeled in the human motion analysis projects. Many systems utilize low level features as the input space, so as to allow added flexibility in the applicability of the developed methodologies. For human motion analysis, approaches combine the low-level data with higher level analysis. In comparing detection of low level features such as positional data to analyzing human motion, Aggarwal and Cai note that "human activity recognition is based on successfully tracking the human through image sequences, and thus is considered to be a higher level task" (1999).

The most common video-based motion track analysis application is video content and surveillance analysis[2] (Aggarwal & Cai, 1999; Hongeng, Bremond, & Nevatia, 2000; Ivanov, Stauffer, Bobick, & Grimson, 1999; Nevatia, Zhao, & Hongeng, 2003; Oliver, Rosario, & Pentland, 1998). It is due to the ready availability of raw video footage from surveillance cameras and the difficulty in processing said footage.

Two common recognition tasks in analysis of video footage include activities and interactions. A video sequence is referred to as a *scene*, and the people who are observed over the course of a given scene are *actors*. In general, anything which actors can do are *activities*, though the specific scope of the term is often limited to a subset of all possible activities. Nieto uses the term to refer to anything a person may do (1997). Polana and

---

[2] The work of Panangadan, Matarić, and Sakhatme could also be applied in a surveillance manner, though it involves laser range-finders instead of video as the motion source (2004a).

Nelson base their approaches on periodic activities (1994). Tan and De Silva concentrate on activities which might be observed in an office, such as sitting and walking. Also, activities which involve more than one actor are *interactions*. Panangadan, Matarić, and Sukhatme note that "two interacting people generate [motion tracks] from similar distributions" (2004b). Common approaches to interaction analysis consist of two-person interactions, as in (Aggarwal & Cai, 1999; Hongeng, Bremond, & Nevatia, 2000; Ivanov, Stauffer, Bobick, & Grimson, 1999; Oliver, Rosario, & Pentland, 1998; and Panangadan, Matarić, & Sukhatme, 2004b). Developing methodologies apply to multi-person interactions (Nevatia, Zhao, & Hongeng, 2003).

2.4 Approaches to Motion Track Analysis

Many modeling techniques have been explored in the area of motion track analysis. The first approaches utilized hard-coded rules or thresholds to identify clusters of motion tracks. Contemporary approaches utilize statistical modeling methods and machine learning algorithms, often in conjunction with one another.

Aggarwal and Cai, in summarizing some of the techniques, note that most approaches concentrate on modeling human behaviors as a state-space (1999). Modeling techniques which fit in this area include Hidden Markov Models (HMMs) (Cielniak, Bennewitz, & Burgard, 2003; Oliver, Rosario, & Pentland, 1998) and grammars for motion languages, an approach borrowed from natural language processing (Bobick & Ivanov, 1998; Hongeng, Bremond, & Nevatia, 2000; Ivanov, Stauffer, Bobick, & Grimson, 1999; Nevatia, Zhao, & Hongeng, 2003).

Alternative methods match templates of a given activity to motion tracks (Aggarwal & Cai, 1999). The activity templates are often generated via machine learning algorithms. Some algorithms which have been used include k-Nearest Neighbors (Efros, Berg, Mori, & Malik, 2003), Expectation-Maximization (EM) clustering (Bennewitz,

Burgard, & Thrun, 2002; Cielniak, Bennewitz, & Burgard, 2003), Bayesian networks (Patterson, Liao, Fox, & Kautz, 2003), and AdaBoost (Viola, Jones, & Snow, 2003).

In keeping with the concept of constructing multilayered approaches to analyzing human behaviors, recent approaches utilize techniques from both machine learning and state spaces in implementing the overall solution. For example, Cielniak, Bennewitz, and Burgard use EM clustering to identify common motion paths through the environment, and then derive HMMs to model the paths (2003). Panangadan, Matarić, and Sukhatme opt for a hierarchical clustering algorithm to generate activity types, and then model those using Poisson distributions (2004a).

Our implementation has similarities to the template methods, in that we develop machine learning templates of human behaviors. The main difference between our approach and previous template methods is that we do not store explicit templates; rather, the activity template is stored in the implicit representation of the generated machine learning models. Our approach is very flexible in modeling behaviors, as the feature space discards position-specific information in favor of relative features, such as speed and turning angle. Chapter 4 introduces our selected feature set, and Chapter 5 discusses further implementation details of the construction of said models.

Having completed our overview of the field of motion and behavior analysis, we now turn to other foundational necessities for our project, beginning with machine learning theory.

CHAPTER 3: MACHINE LEARNING THEORY

Chapter 2 explores related work in the field of motion and behavior analysis. We now proceed to establish the framework of machine learning theory.

Section 3.1 introduces learning theory. Different areas of machine learning are outlined in Section 3.2. Section 3.3 outlines in brief the major modeling algorithms which are referred to in this thesis. Section 3.4 delves into the bias and variance trade-off in hypothesis selection. Feature selection and the curse of dimensionality are discussed in 3.5.

3.1 Learning

Learning is a difficult concept to grasp. A preliminary definition might consist of "transfer of knowledge," though such a definition is riddled with ambiguities all its own. For one, what is the knowledge source? Can learning occur without a teacher or other provider of knowledge? Of course. We can recognize *learning* when it happens, but to quantitatively establish some metric to determine how it has occurred is somewhat more difficult. Machine learning often does not tackle the problem of disambiguating *learning* prior to measuring it. The metric most often utilized is this:

Learning is evidenced by improvement over time (Witten & Frank, 2000).

Thus, by evaluating the performance p_0 of the learner at time t_0, and then again measuring the performance p_1 at time t_1, where $t_1 - t_0 > 0$, if $p_1 - p_0 > 0$, *learning* has

taken place (Mitchell, 1997). *Machine learning* is the study of algorithms by which computer models improve after experience.

Machine learning algorithms operate on data. Given a set of data *S*, various tasks could be performed on it. Each element **x** of the set *S* is an *instance* of data residing in N-dimensional space. Each dimension can be defined as binary valued, nominal valued, or real valued. Thus, $\mathbf{x} \in ¡^{N}$ with elements as follows:

$$\mathbf{x} = \{x_1, x_2, \mathrm{K}, x_N\} \tag{3.1}$$

The goal is to develop a model which represents the information contained in the set *S*, which model shows improved performance on representing *S* after the model has examined the elements $\mathbf{x}_i \in S$ for all *i* such that $1 \le i \le |S|$.

3.2 Learning Methods

Reviewing the question we raised earlier, can learning occur without a teacher or other provider of knowledge? The answer is most assuredly in the affirmative. The branch of machine learning dealing with independent learning is *unsupervised learning*. In this branch, a model is generated to identify natural patterns, groupings, or clusters in the input data. Clustering algorithms evaluate some metric of similarity to determine whether a pair of input instances should be grouped together or not.

An alternative approach to learning is to have a teacher or expert on the data set *S* provide labels for each element **x** in *S*. Thus, *S* consists of paired elements:

$$S = \left\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \mathrm{K}, \left(\mathbf{x}_{|S|}, y_{|S|}\right)\right\} \tag{3.2}$$

where each vector $\mathbf{x}_i$ is paired with a label y_i. This branch of machine learning is *supervised* learning, since a supervising expert provides the labels. When $y_i \in ¡$, the learning method is *regression*; when y_i is an element of a finite set of elements, the model technique is *classification* – mapping the input space to one of the available class labels.

In general, rather than attempting to discern natural clusters which exist in *S*, the task of supervised learning is to develop a mapping function *f* as follows:

$$\begin{array}{l} f : ¡^{N} \text{ a } ¡ \\ f(\mathbf{x}_i) \text{ a } y_i \end{array} \tag{3.3}$$

S is referred to as *training data* or a set of *training examples*. Additional samples from the distribution which generated *S* can be obtained to provide *validation* or *test examples*.

In order for a machine learning model to truly be of use, it is generally the case that the training data represents only a sample from the true population of the domain of the target function *f*. The task of a learning algorithm is not to enumerate a perfect mapping for all possible elements of the domain, but rather to determine which of a set of possible functions represents the best fit for *f* given the training data *S*. A possible function is a *hypothesis*. For a given set of training data *S*, a multitude of hypothesis functions exist.

In the next section, we discuss several specific machine learning algorithms. The material is not intended to be comprehensive. If further information on a specific algorithm is desired, such sources as (Mitchell, 1997) or (Witten & Frank, 2000) might be of use.

3.3 Basic Modeling Techniques

We have covered the preliminary concepts necessary to understand any machine learning algorithm. Now we examine particular algorithms which will be utilized in this thesis. Specific models discussed are one rule, decision tables, decision stumps, decision trees, and SVMs.

3.3.1 One Rule

The one rule algorithm is very straightforward. Of the attributes of the input space, a single attribute is selected using information gain, an entropic measure of the spread of the data set. The algorithm then generates a set of thresholds to divide up the chosen input attribute. The output class of a bin is the most frequently occurring class of that portion of the training data which falls into the bin.

Models generated with the one rule algorithm tend to be simplistic in the types of hypotheses which can be represented.

3.3.2 Decision Tables

The decision table algorithm is closely related to the one rule algorithm. It generates a set of attributes with thresholds on each attribute to map the input space to the output labels.

Because of the added complexity of the decision table algorithm over the one rule algorithm, the space of hypotheses which can be represented is much greater.

3.3.3 Decision Stumps

The decision stump algorithm has similarities to the structure of the one rule algorithm. With decision stump, we move from rules to a tree structure. Leaves of the tree represent class labels. Interior nodes of the tree are decisions, very similar to the rules of one rule and decision tree. The one major difference lies in that a decision tends to be a single threshold rather than a set of thresholds for the given attribute.

The decision stump algorithm generates at most one interior node, a single decision. Thus the hypotheses which can be represented are limited to those which can be determined by a single threshold on a single attribute.

3.3.4 Decision Trees

The decision tree algorithm is a generalization of a decision stump, removing the depth limit previously imposed. The generated decision tree has much greater representational power, and hence can generate a much larger portion of the hypothesis space than the decision stump algorithm. The implementation we use is the J4.8 algorithm.

3.3.5 Support Vector Machines

Support vector machines (SVMs) are discussed in fine detail in Appendix A. Here, we note that the resulting models represent decision boundaries in a high dimensional space. The mapping from the input attribute space to the decision boundary space is accomplished by means of a kernel function. The flexibility of the decision boundary is determined in large part by the selection of the kernel function. Polynomial kernels and Radial Basis Function (RBF) kernels are used later in this thesis.

In the next section, we discuss how a given learning algorithm selects among the many available hypotheses, namely bias and variance.

3.4 Hypothesis Selection and Model Complexity

As discussed in section 3.2, for a given domain, many potential hypotheses exist. Each machine learning algorithm performs the task of searching through hypothesis space for the best hypothesis to fit the training data set *S*. An added challenge is that the hypothesis should not only fit *S*, which is a sample from some unknown distribution, but should also generalize to match as yet unobserved examples from the same distribution which generated *S*. In other words, "Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples" (Mitchell, 1997).

Herein is the crux of machine learning. Since hypothesis space is vast, and the machine learning algorithm is only permitted to examine a sample of the target function, each learning algorithm must select among multiple hypotheses. The method by which this is most readily accomplished is bias/variance trade-off. We first examine bias, then variance. We conclude this section by discussing methods for adjusting the complexity of some of the machine learning models of section 3.3. See Figure 2.

3.4.1 Inductive Bias or Capacity

In order to select from the available hypotheses, each machine learning algorithm introduces an inductive bias in selecting certain hypotheses over others (Mitchell, 1997). A given machine learning algorithm has limits on which portions of the potentially available hypothesis space it can represent. For example, decision stump and one rule each select a single attribute from the input space, and establish one or more thresholds

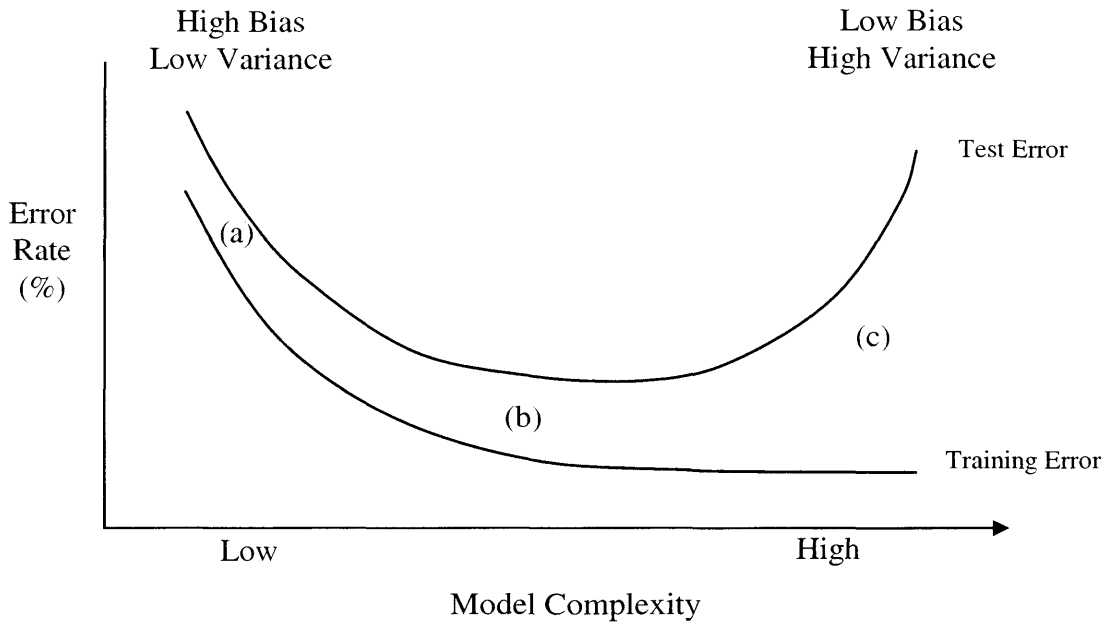

Figure 2: Bias/Variance trade-off

As model complexity varies, so do the bias and variance of the model. In general, low complexity models (models with high bias and low variance) tend to have higher error rates on both training and unseen test data (a). As the complexity of the model increases, performance tends to improve (b) on both the training data and the test data. High complexity models often reach a point of overfitting the training data (c), where improvements in training performance are not reflected in test performance. Model validation attempts to identify potential test data performance for a model of given complexity, as evaluation solely on the training performance tends to lead to overfitting.

on the values of that attribute. Models developed by these algorithms have a bias against hypotheses which depend on the value of multiple attributes.

In other words, bias[3] is a measure of how well a given learning algorithm is suited for matching the target function *f* (Witten & Frank, 2000). Modeling algorithms such as one rule and decision stumps are ill-suited for many target functions; hence, they have *high bias* toward specific hypotheses. Decision tables and decision trees have *low bias* toward a specific hypothesis, as they are able to represent fairly complex hypotheses.

Some machine learning algorithms can vary the bias of generated models in proportion to the complexity of the model generated. Models with a high bias tend to *overfit*, or memorize, the training data *S*, but fail to generalize well to the target function, and thus to testing data (Mitchell, 1997).

3.4.2 Model Variance

A balancing metric for measuring how well a model performs on unseen data is *variance* (Witten & Frank, 2000). From statistics, variance is a measure characterizing width or dispersal around a mean (Mitchell, 1997). In discussing the variance of a learning algorithm, the concept is similar. The variance measures how much a given model changes when trained on different data sets generated by the same target function *f*. Models with high variance tend to perform poorly on unseen test examples.

Of the modeling techniques from section 3.3, one rule and decision stumps tend to have fairly stable models for new training data, meaning the resulting hypothesis will not tend to change much. Conversely, decision tables and decision trees often will vary greatly as the training data is altered. Thus, one rule and decision stumps have *low variance*, while decision tables and decision trees have *high variance*.

---

[3] Cristianini and Shawe-Taylor refer to this measure as the *capacity* of the learning algorithm (2000).

3.4.3 Bias/Variance Trade-off

It is not difficult to notice a general trend. A model with high bias tends to have low variance, and a model with low bias tends to have high variance. The two metrics are in opposition to each other. In general, Witten and Frank note that the expected error of a given model on test examples can be predicted by the sum of the bias and variance (2000). This leads to the bias-variance decomposition or trade-off – the need to find a balance between bias and variance.

An added control in determining the bias and variance of a model consists of adjusting the model complexity. As noted in Section 3.3, one rule and decision stumps are simplistic models. They both have high bias and low variance. Decision tables and decision trees have the potential of generating much more complex models. These exhibit low bias and high variance. There is a strong connection between model complexity, and bias and variance. By controlling the model complexity, the balance point between bias and variance can be found.

As mentioned in Section 3.2, a secondary goal of all machine learning algorithms is to generate a model that both represents the training data well and generalizes to new data. See Figure 2. In general, a low-complexity model tends to perform poorly on both the training and testing data. As the model complexity increases through training, performance on both the training and testing data improves (as is our definition of learning from Section 3.1). At some point, the testing performance begins to degrade even though the training performance is improving. At this point, the model is *overfitting* the training data. The model has low bias and high variance. Its complexity is too high.

Machine learning algorithms incorporate a method for limiting or adjusting the model complexity. One rule can limit the number of thresholds permitted for an attribute, as can decision tables for the set of attributes. Decision trees can limit the tree depth or perform validation pruning to remove overfit branches.

We have not discussed the complexity, bias, nor variance of SVMs as yet. SVMs have much flexibility in the decision surface generated. The complexity of a surface is influenced in large part by the kernel function selected. Burges states that a radial basis function kernel "of sufficiently small width can classify an arbitrarily large number of training points correctly" (1998). This translates to a very low bias model. Polynomial kernels of high degree have low bias, as their representational power is high. Lower degree polynomial kernels move toward higher bias. For example, the linear kernel can only represent a linear decision boundary.

For SVMs, an added control over complexity is found in the regularization constant C. This is discussed in finer detail in Appendix A. For our purposes, it is sufficient to note that C serves as an upper bound on the amount a given α_i can increase to. The result is a smoothing effect on a decision surface, limiting the complexity of the resulting hypothesis.

In terms of variance, Cristianini and Shawe-Taylor state that SVMs models tend toward low bias, low variance models (2000). This claim is founded in the use of support vectors. A direct result of this is that changes to non-support vector instances of the training data will have very little effect on the hypothesis.

We have discussed bias and variance as they relate to hypothesis selection, as well as some specific methods incorporated into some of our machine learning algorithms for controlling the model complexity. We now discuss additional methods by which the ideal balance can be found between bias and variance.

3.4.4 Validation Methods

Methods for estimating the optimal model complexity exist, and collectively are referred to as validation methods. Several validation methods of importance for later

discussion include use of a training and validation set, cross validation, and leave-one-out validation.

Since a primary goal of a trained model is generalization to unseen data, a straightforward approach is to reserve a portion of the available training data. This reserve is not used for further training of the model, but rather for practice runs on unseen data from the original distribution. Model performance can then be optimized for this validation data set, rather than for the training data set. In this manner, overfitting of the training set tends to be avoided. A prime disadvantage of this approach, however, is that the training data is drastically reduced.

Cross validation overcomes the disadvantage of using a validation set by cycling through all instances from the training data set for use in a validation set. The training data set is divided into n subsets of equal size. Each subset is then used once as the validation set, and the remaining $n-1$ subsets are merged and used as the training set. At the completion of the validation phase, n models have been generated, but more important, the performance of the entire data set has been evaluated once as though it were 'unseen' by the trained models. In this manner, potential generalization can be better gauged. Ten-fold cross validation is usually used, where $n=10$. Additionally, to provide a measure of statistical confidence, ten-fold cross validation is performed ten times, with a different random number seed to mix the training data prior to splitting.

Cross validation is impractical for small data sets, especially where the number of instances is less than ten. In such cases, a viable alternative is leave-one-out validation. As the name suggests, one instance is not used for training a model, and then the trained model is evaluated on the withheld instance.

We use ten-fold cross validation to construct models for the experiments in Chapter 5. Understanding of bias and variance is important for Chapter 5, where we discuss the impact of model complexity on motion track classification. Prior to that, we examine general information regarding feature selection.

3.5 Feature Selection

The task of machine learning as stated in Section 3.1 is to learn a target function *f* to map input instances from the input space to a class label. We are not limited to the input space of the function. A layer of preprocessing is often included. The raw input data is referred to as *attributes*, while any new dimension generated as a result of preprocessing is referred to as a *feature*. These terms are important for the discussion of SVMs in Appendix A.

We discuss the problem of the Curse of Dimensionality which arises with a high dimensional space, and preprocessing in general.

3.5.1 Curse of Dimensionality

It should be noted that the information content being learned by a given machine learning model increases exponentially as the dimensionality N of the input space increases. In other words, the number of potential hypothesis functions grows much faster than the dimensionality of the input space. Thus, calculating new features serves both to provide added information and to increase the number of possible hypotheses. The exponential increase in hypothesis space is referred to as the *curse of dimensionality* (Cristianini & Shawe-Taylor, 2000; Mitchell, 1997). With higher dimensional spaces, modeling the decision function *f* becomes increasingly difficult. In general, this translates to an increase in training and evaluation time.

3.5.2 Preprocessing

Once the curse of dimensionality is understood, it becomes apparent that increased quantity of available input data does not necessarily engender improved performance. In fact, Witten and Frank conduct an experiment in which a randomly

generated attribute is added to a data set, in order to better understand what effect poor attribute selection has on the performance of machine learning on a data set (2000). Witten and Frank note that inappropriate attributes potentially degrade performance from five to ten percent (2000).

Going beyond the initial selection of the input attributes is the *preprocessing* step. In preprocessing, additional attributes, referred to as *features*, are generated. Often, feature generation combines several attributes, such as computing an average, or performs other functional processes on the input attributes. Cristianini and Shawe-Taylor define feature selection as extracting pertinent information from a given set of attributes (2000). Even if the input attributes are not well suited for modeling with any available machine learning algorithm, a strong set of features may be extracted, drastically improving performance.

In Chapter 4, we discuss preprocessing steps taken to prepare our motion tracks for modeling. Specifically, we will outline the set of features which are extracted from the positional information contained in the motion tracks.

3.5.3 Missing Attributes

In machine learning applications, training instances of a data set S need not be consistent, or even complete. Mitchell discusses approaches for dealing with input instances which are missing attribute values (1997). The first method replaces missing attributes with the mean value of that attribute over all other instances in the training data. For our experiments, if necessary, we will utilize this approach.

Alternative methods discussed by Mitchell include computing the mean of the attribute given the output class, and splitting the instance according to the frequencies of the attribute values, and allowing the instance to contribute fractionally to all possible values (1997). This last method becomes impractical for real-valued attributes.

Other approaches exist for solving the problem of missing attribute values in a data set. In analyzing our segmentation algorithm in Chapter 5, we will reduce the original problem of comparing motion tracks of disproportionate lengths to a missing attribute problem.

This brings our theoretical discussion of machine learning to a close. In Chapter 4, we examine the preprocessing methods used to prepare the specific data sets for use in the machine learning algorithms we covered in this chapter.

CHAPTER 4: PREPROCESSING

The discussion of machine learning theory from Chapter 3 covers preprocessing techniques and their usefulness. This chapter outlines specific preprocessing tasks we perform on our motion track data sets. The major goal of the preprocessing process is to ease the burden of the machine learning algorithm in learning the target function *f*. Section 4.1 introduces the specific motion track data sets we have at our disposal. Section 4.2 presents the method of activity labeling. Section 4.3 delves into the process of feature selection, with specific effort spent outlining the selection of our feature set.

4.1 Motion Track Data Sets

Perhaps the most key aspect of any machine learning application is the data set itself. Thus far, we have discussed foundational concepts (Chapter 1), related work and background efforts (Chapter 2), and the theory of machine learning (Chapter 3). We now have the necessary tools to apply to the data.

We consider two major data sets of motion tracks, both extracted from video captured by a single video camera at approximately 45 degree angle above the plane of action. Actors in each video scene consist primarily of pedestrians, though at times cars and bicyclists are also active in the scenes. All scenes are outdoor sequences. The first set of motion tracks consists of staged traffic with a person assigned the task of passing out flyers. The second set of motion tracks is made up of sports games, namely Frisbee, volleyball, and soccer.

Our goal is to identify objects based on how they move, or rather, on their behavior. To this end, most of our activities are staged. We examine first the flyer scenes, followed by the sport scenes.

4.1.1 Flyer Scene Data Sets

As noted above, the first of our data sets consists of a staged flyer activity. See Figure 3 for images of the flyer scenes and the motion tracks extracted from them. One individual has a handful of flyers, and is attempting to distribute them to other people who travel through the scene. Reactions from travelers are varied, from pausing to chat for a few seconds to outright avoidance of the flyer person.

As we wish to model our scenes from a behavioral standpoint, we examine the behavioral content of such a scene. The motion patterns consist of a collection of individuals who travel through the scene at fairly constant velocity, much as normal scene traffic tends to do. Occasionally, an individual will pause or stop, perhaps in reaction to intervention from the flyer person. The flyer person, on the other hand, meanders about the scene. Their motion of meandering is a mixture of standing with slow walking and frequent direction changes (often, referred to as walking aimlessly). At times, the flyer person moves to intercept a traveler. In such instances, the motion is direct and may be swift, as the flyer person is attempting to either meet or catch up with a traveler. The motion track of the flyer person is distinctive from the other individuals in the scene, and hence serves as a basic example of our goal, classifying a motion track based on its behavioral distinctiveness.

For added robustness in modeling the motion data, we have captured several sequences of the flyer activity at two separate locations. The first location is an intersection, the second is a plaza. Each location has its own distinct background traffic. Between scenes captured at the same location, variations naturally occur in the positions and paces of the actors. Any modeling technique must take into account the variations in similar activities, while differentiating between truly separate activities.

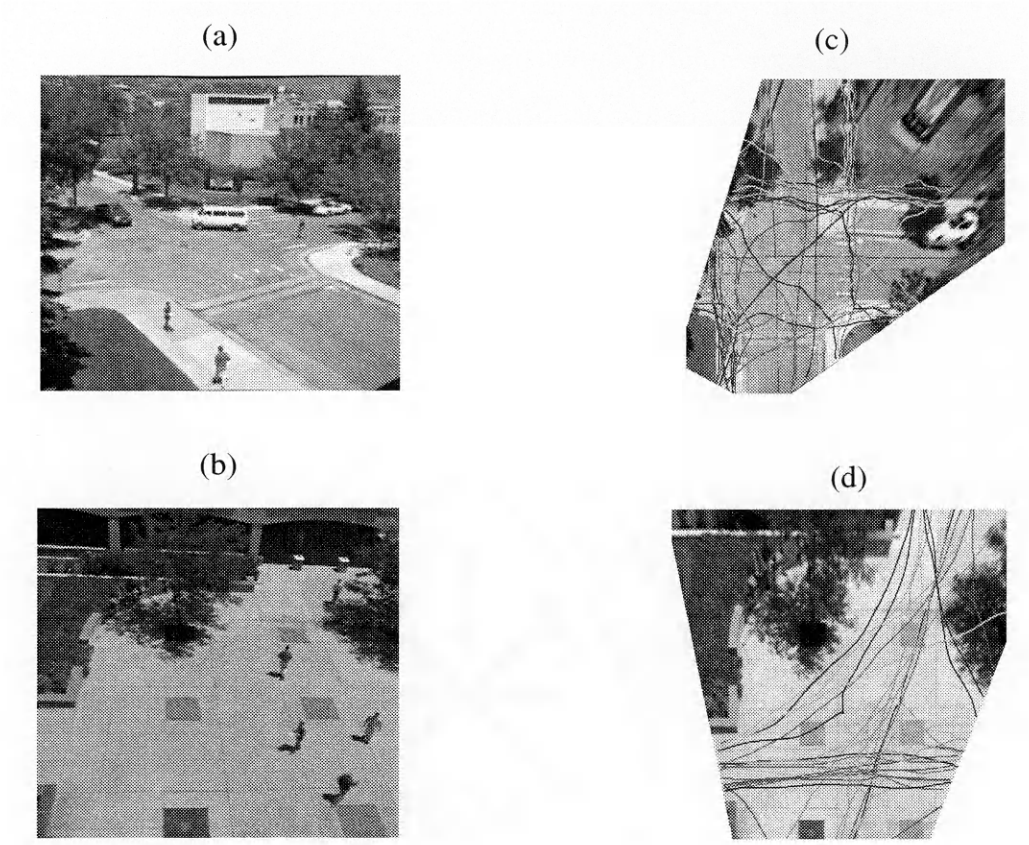

Figure 3 Flyer scenes and motion tracks

We compare data from two locations. The first location is an intersection (a). The second location is a plaza (b). In both locations, a person is assigned the task of distributing flyers to travelers. Motion tracks of moving objects in each scene are shown in (c) and (d), respectively.

4.1.2 Sports Scene Data Sets

We now turn to the second data set. As one of our more distant goals involves modeling of large-scale structured group behaviors, we model sports. For a given sport, activities in which players engage and the behaviors exhibited are governed by the nature of the game, its rules, and the opposing goals of the various teams. As each sport has different rules and the players have differing objectives, the motion tracks should exhibit different behaviors, and thus be distinguishable at the motion level.

Specific sports which we examine are ultimate Frisbee, volleyball, and soccer. Each is governed by different rules and goals; hence, the players exhibit different motion patterns. Figure 4 shows an example frame from each sport sequence, as well as the set of extracted motion tracks. We review the rules in brief here.

For a Frisbee game, each team attempts to move the Frisbee toward their goal at opposing ends of a field. The Frisbee is moved by throwing it from one team member to another. When a player has the Frisbee in hand, they are not permitted to take more than a limited number of steps (usually three) before throwing it. If the Frisbee is not caught, the opposing team takes possession of the Frisbee and begins moving it toward their goal.

The next sport we consider is soccer, which has many similarities to Frisbee. The team moves a ball toward their goal, again at opposite ends of a field. The ball may not be touched by hand, only by foot or other body parts (head is allowed). The ball is dribbled by a player with their feet, and passed to other players on the same team. No stopping is necessary as possession of the ball changes teams.

Volleyball differs from both soccer and Frisbee in its game structure, and thus is distinguishable from the other two. In volleyball, each team is separated by a net. The objective is to keep the ball from touching the ground by hitting it from one player to another. When the ball touches the ground on one side of the net, a point is awarded against the team on whose side the ball fell. A general strategy is for each team to spread

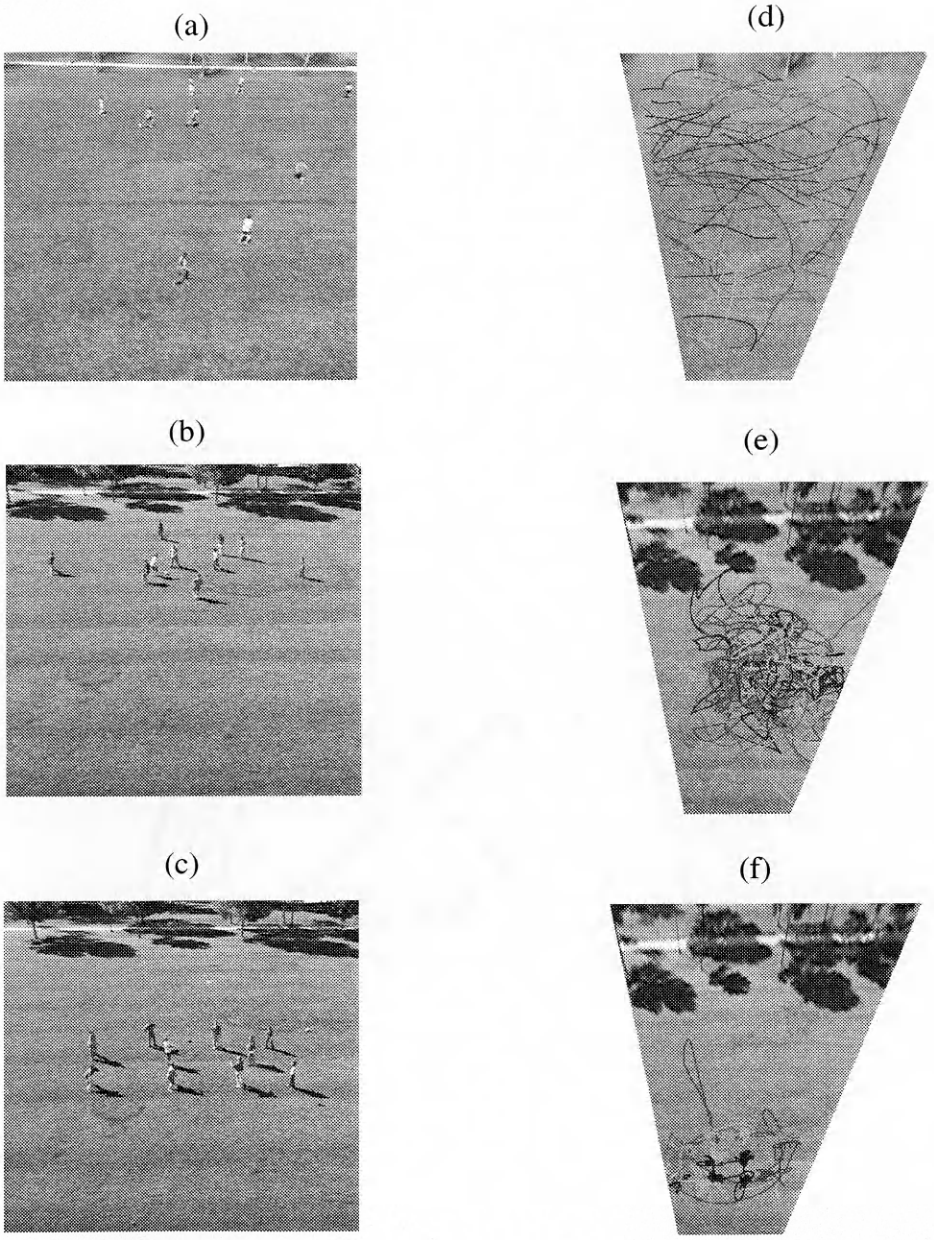

Figure 4: Sports scenes and motion tracks

The modeled sports are Frisbee (a), soccer (b), and volleyball (c). Extracted motion tracks are shown in (d), (e), and (f), respectively.

their players uniformly across their side of the net, and to allow each player to cover a section or zone of their territory. As a result, players are mostly stationary or move within a certain region. Such motion is governed strongly by the path of the ball.

In gathering the motion tracks of the players in each scene, the position of the ball or Frisbee is ignored. In future experiments, we may consider including such information as reference objects with which the actors in a scene interact. For now, we strive to identify the behavior of the group without consideration of the motion of the object of focus. Added information would only improve any classification or clustering algorithm.

4.1.3 Video Tracking Process

As noted earlier in this section, we capture video footage from a single video camera of each scene. The video footage is digitized, and then motion tracks are extracted via a tracking system. Many tracking systems have been proposed, each with its own strengths and flaws. For the purposes of this thesis, we are not attempting to evaluate tracking systems. Instead of an automated system, the motion tracks are identified by a person, as this provides accurate motion track information on the individuals in each video sequence while being robust to both noise and occlusion. Output from automated tracking systems would tend to add noise and errors in the production of motion tracks. By using a hand tracking system, we eliminate the extra noise and focus on the specifics of modeling the activities.

The results from the tracking process are the paths which the tracked objects followed while traveling through the scene. Rather than restricting the system to assigning a single label to each motion track, we allow multiple labels to be given to different portions of a track. Each label identifies an activity. Activity labeling is discussed in Section 4.2. We generate descriptive features which are calculated directly from the raw positional data to distinguish both motions and behaviors. Our feature set is presented in Section 4.3. For feature generation to be successful in producing a fixed-

length feature vector, we first perform segmentation of the data. The segmentation process is discussed in Chapter 5.

4.2 Activity Labeling

The result from the tracking process of Section 4.1.3 is a set of motion tracks. Panangadan, Matarić, and Sukhatme note that "a person's track may span more than one activity" (2004a). They establish *activity* in terms of periods of similar motion across all tracks in the set, as discussed in Chapter 2, and apply an entropic segmentation methodology for breaking motion tracks into variable-length activity segments. The resulting segments are then compared only to segments of similar size and location. We agree with the concept that a motion track may consist of multiple activities; however, from a behavioral analysis standpoint, an activity may also occur for differing lengths of time, and in vastly different locations of a scene.

Without loss of generality, we can assume that a motion track with multiple activities can be broken into smaller motion tracks, each consisting of a single activity. Thus, we consider a set of activities rather than motion tracks.

Chapter 5 discusses our methodology for generating segments of comparable length. The process uses as its base labeled activities within a motion track. The labels are assigned by hand, much as the tracking of objects is performed by hand. An activity label consists of a motion track, an index into the position list of the track for both the start and end of the activity, and a label of the activity. The task of Chapter 5 is to generate segments of uniform size which can be compared directly.

For the next sections, we assume the segmentation method of Chapter 5, and discuss generation of features from activity segments. The number of features generated depends on the length of an activity segment, and thus the exact dimensionality of the final input space for the machine learning classifiers will be determined by the segment length.

4.3 Feature Selection

We consider the task of generating a feature vector for use in construction of machine learning models from a given activity segment. The result of segmentation (Chapter 5) is a set of activity segments which are uniform in length. Each activity segment consists of a sequence of points from its source motion track. Formally, this is $s = \langle p_1, p_2, p_3, \mathrm{K}, p_l \rangle$, where l is the segment length selected as a result of the analysis of Chapter 5.

The original position information could be retained; however, for purposes of behavioral analysis, the position at which a behavior occurs is irrelevant in terms of describing the behavior. We discard positional data in deference to more descriptive features. Specific features included are the velocity (Section 4.3.1), acceleration (Section 4.3.2), and the speed and turning angle (Section 4.3.3) of the activity at each position along the segment. Speed and turning angle are aggregated to provide the minimum, maximum, and average values over the activity segment (Section 4.3.4).

4.3.1 Velocity Features

The velocity of the object can be computed at each point of a track segment by use of the forward difference (Farin, 2002):

$$\mathbf{v}_j = \Delta p_j = p_{j+1} - p_j \tag{3.4}$$

where $\mathbf{v}_j$ is the velocity vector at point $p_j = (x, y)$. Velocity is a vector in two-dimensional space. Figure 5 shows the velocity vectors $\mathbf{v}_0$ for the flyer scene with

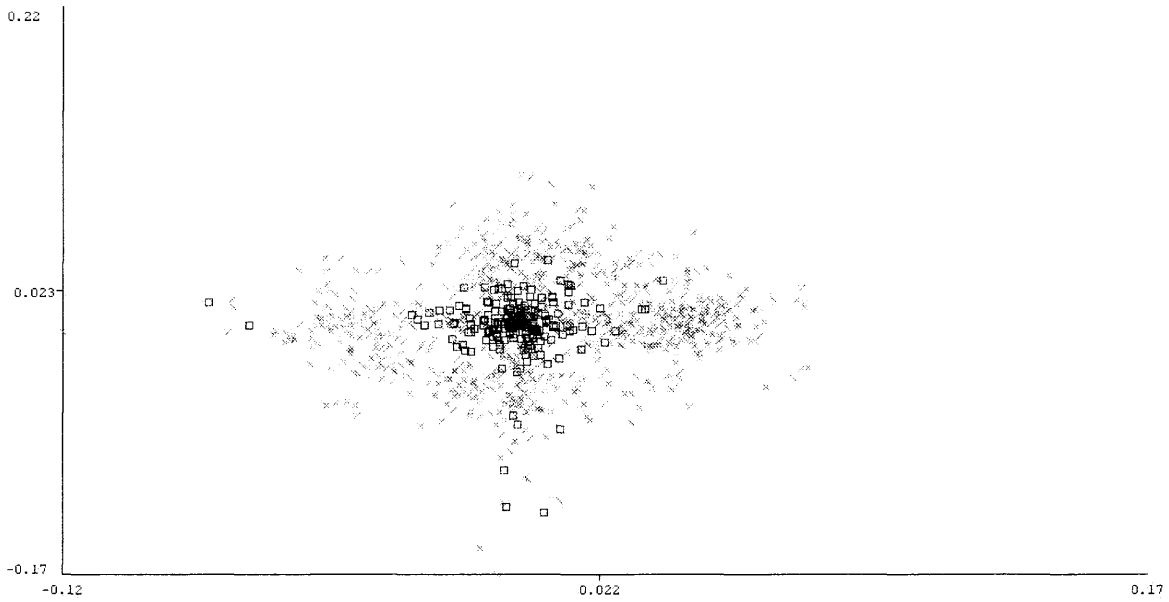

Figure 5: Initial velocity plot of flyer scene

For each activity segment in the flyer scene, the initial velocity vector in the x and y direction is considered. Labels are assigned by a human user. The squares represent activities which are labeled meandering, and the crosses represent activities which are labeled normal. The center of the plot corresponds to the vector (0, 0), or no velocity at all. The meandering activities are clustered tightly toward the center, while the normal activities tend to be spread out.

classifications of meandering and normal, where each velocity is represented by a point at its head for clarity.

4.3.2 Acceleration Features

The acceleration of an object can be computed at each point of a track segment by use of the second forward difference (Farin, 2002):

$$\begin{aligned}\mathbf{a}_j = \Delta^2 p_j &= \Delta p_{j+1} - \Delta p_j = \left(p_{j+2} - p_{j-1}\right) - \left(p_{j-1} - p_j\right) \\ &= p_{j+2} - 2p_{j-1} + p_j\end{aligned} \tag{3.5}$$

where $\mathbf{a}_j$ is the acceleration at point p_j. Figure 6 contains a graph of acceleration vectors $\mathbf{a}_0$ for the flyer scene with class labels of meandering and normal, with the head of each acceleration vector being noted by a point for simplicity.

It should here be noted that for an activity segment of length l, velocity can be computed for only $l-1$ of the points, and acceleration can only be computed for $l-2$ of the points of a given segment. For simplicity in calculations, we restrict calculated features to $l-2$ of the available points in a segment, as this is the shortest of all features.

We now consider the additional features of speed and turning.

4.3.3 Speed and Turning Angle Features

The speed s of an object at any given point j is the magnitude of its velocity $\mathbf{v} = [v_1, v_2]^T$ at j. Magnitude of a given velocity vector is computed thus:

$$s = \|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2} \tag{3.6}$$

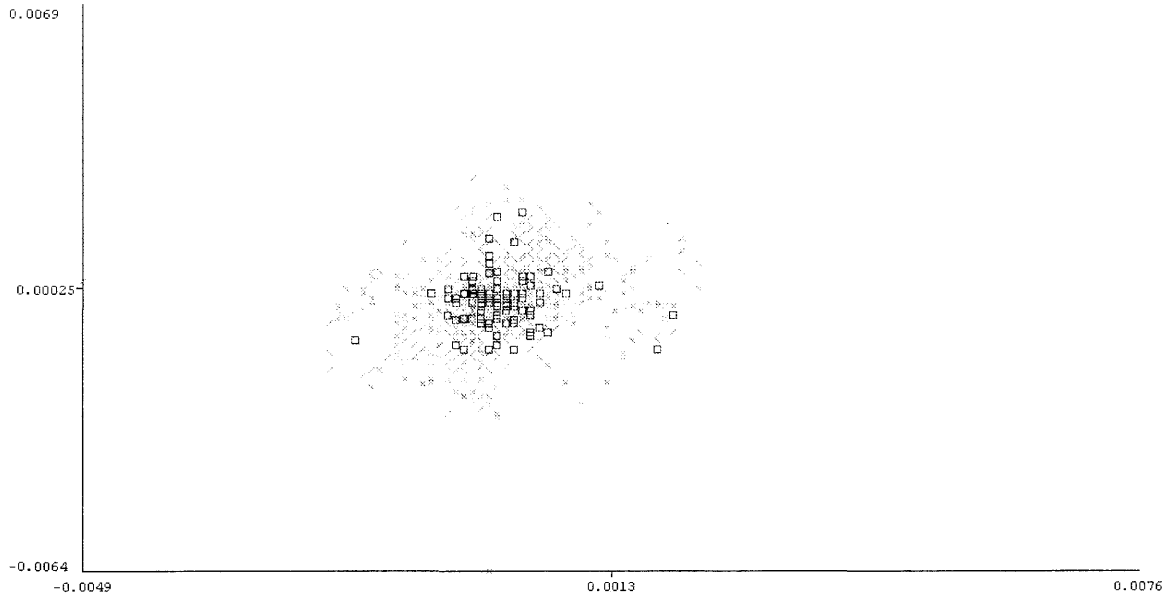

Figure 6: Initial acceleration plot of flyer scene

This plot shows the initial acceleration vector in the x and y directions. At the center of the plot is the vector (0, 0), corresponding to no acceleration. The labels are generated by a human user. Squares represent the acceleration of activity segments labeled meandering, and crosses represent activity segments labeled normal. In general, as with the velocity plot in Figure 5, the meandering activities show less spread than the normal activities.

The amount an object turns at a given point is the difference in angle between the velocity vector of the current point and the velocity vector of the next point. Thus, for a given point *j*, the turning angle θ is computed as follows (watko, 1998):

$$\theta = \cos^{-1}\left(\frac{\mathbf{v}_j \cdot \mathbf{v}_{j+1}}{\|\mathbf{v}_j\|\|\mathbf{v}_{j+1}\|}\right) \tag{3.7}$$

where $\mathbf{v}_j \cdot \mathbf{v}_{j+1}$ is the dot product between the two velocity vectors.

4.3.4 Aggregated Values

Our feature vector consists of velocity, acceleration, speed, and turning computed at every position of a given activity segment. Thus, the exact quantity of input features for a segment is proportional to the size of the segment l, determined in Chapter 5. As long as the segment sizes are uniform, the quantity of input features will be uniform. In addition to this set of features calculated from the position array of the segment, we also gather the minimum, maximum, and average values for speed and turning angle over the activity segment.

Figure 7 shows a plot of the speed versus the turning angle for each activity segment in the flyer scene, labeled as meandering or normal. As is noted in Chapter 5, the class labels are roughly defined by a human user. In this graph, the meandering behavior becomes apparent as both slow and turning often, especially as the turning angle is compared to the normal activities.

This chapter introduces the data sets we model in Chapter 6, as well as the preprocessing techniques which generate the feature vectors on which the machine

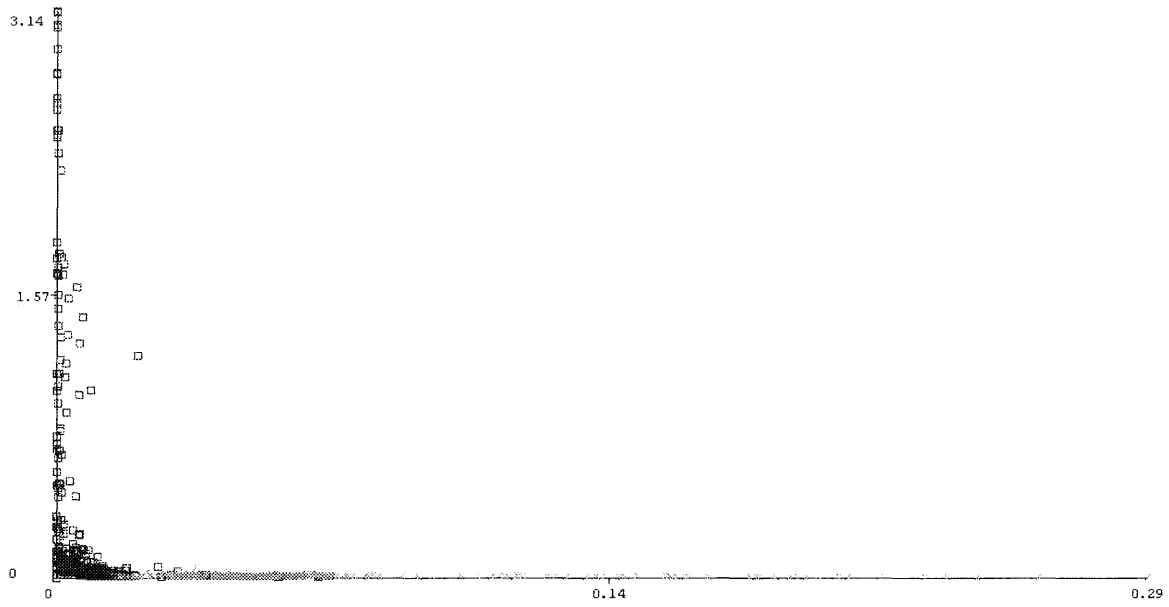

Figure 7: Average speed versus average turning angle of flyer scene

This plot considers the averaged speed (x axis) and the averaged turning angle (y axis) over a given activity segment. The labels are generated by a human user. Squares represent activities labeled meandering, and crosses represent activities labeled normal. Normal activities tend to be faster and straighter than meandering activities.

learning algorithms operate. The only factor which remains to be determined is the size of an activity segment. Chapter 5 discusses the impact of the choice of the segment length, and outlines the method through which we select this parameter.

CHAPTER 5: SEGMENTATION

Chapter 4 provides a brief overview of the path motion data takes from initial capture via a video camera to a feature vector ready for processing by a machine learning algorithm. Motion tracks are extracted from the video, labeled with constituent activities, broken into activity segments, and then converted to the feature vector. The key aspect which remains prior to performing classification of the activity segment is determining the segment length.

Section 5.1 reviews the need for segments to be of uniform length for classification. Section 5.2 introduces our method for determining segments from a motion track with activities labeled. Section 5.3 outlines the experiments performed to explore the overall impact of segment length. Section 5.4 discusses the impact of segment length on different information content metrics. Section 5.5 counters the segmentation drawbacks with an additional metric, windowing. The windowing method is shown to be a promising approach, worth added consideration for future development.

5.1 Comparing Track Data

In discussing machine learning algorithms in Chapter 3, we noted that machine learning algorithms assume all input instances to contain the same number of attributes. In order to successfully perform machine learning, we must ensure that all generated feature vectors for a collection of motion tracks are of identical length.

Chapter 4 touches on some of the segmentation work performed by Panangadan, Matarić, and Sukhatme (2004a). Some of the ambiguities of their approach were mentioned in Chapter 4, such as an assumption that an activity occurs at a specific location and for a fairly similar duration. With these assumptions, they replace an

activity with a set of twenty samples from an approximating cubic spline (Panangadan, Matarić, & Sukhatme, 2004a). Were we to follow this approach, we would lose the ability to compute such parameters as velocity and acceleration. Additionally, behavioral activities may occur over varying durations.

Due to the inapplicability of the assumptions made in their segmentation approach, we turn to a different method more suited to our specific application. The method consists of dividing an activity into uniform segments through time, and is described in greater detail in Section 5.2. The overarching ideal segmentation results in a minimum amount of discarded or lost data, and a maximum segment size.

5.2 Segmentation Method

The activity labeling process of Chapter 4 results in a set of activities from all motion tracks which occur in the scene of interest. Our task is to identify the optimal segment size to divide activities into, so that activity segments can be compared by a machine learning algorithm. Once the segment size is determined, the features outlined in Chapter 4 are generated for each segment. The set of segments is then used for model training and evaluation, discussed in Chapter 6.

For a given segment size, we generate segments as follows. First, in order to allow for transitional noise of an individual changing between activities, a predetermined number of data points surrounding the identified activity begin and end points are discarded. These data points cannot be recovered, nor are they influenced by the segment size; thus they represent a lower bound on number of data points which are discarded.

We now are left with a trimmed set of activities for the entire scene. There are no assumptions which can be made regarding the lengths of the activities. Rather than attempting to identify a common segment length to evenly divide into all activities

throughout the scene, we use the given segment length. For an activity of length m, and a segment length n, the number of segments s available in the activity is found by

$$s = \left\lfloor \frac{m}{n} \right\rfloor$$

The remaining points are not sufficient to constitute a segment of length n, and must be discarded.

We evaluate the impact of the selection of the segment size in Section 5.3 by generating a series of segment sets of varying segment sizes.

5.3 Experimentation

Section 5.2 introduces our segmentation approach. We consider several metrics for evaluating the suitability of a given segment size. The first is the number of data points which are lost for a given segment size. The second is the prior probabilities of the activities of the resulting set of segments. The third metric is a count of the number of segments produced, and the length of the feature vector, which represents the dimensionality of the input space. Our final metric consists of ten-fold cross-validation performance of a sample data set at varying segment sizes. With this final metric, we attempt to compare potential modeling performance on the output segment set.

In Chapter 3, we discussed the impact of model complexity on the performance of the model. In order for our findings to generalize to other machine learning algorithms, we use two training algorithms to evaluate performance on the various metrics. One algorithm is low bias, the other high. The selected modeling algorithms for testing segment length are decision stumps (high bias, low variance) and decision trees (low bias, high variance). We now examine the results of the segmentation analysis phase.

5.4 Segmentation Findings

Our task reduces to the problem of, given a set of motion track activities, identify an optimal segment size for generating activity segments. Ideal segment size should be large, to allow a rich feature space for model training, while also minimizing the amount of unusable motion track data points. This task is much more complex than at first glance.

Section 5.4.1 examines the impact of segment size on loss of motion track data points. Section 5.4.2 looks at the changes which occur in the prior probabilities of the activity breakdown as it is affected by segment size. Segment quantity is also affected, as is the dimensionality of the feature space. Both are covered in Section 5.4.3. Section 5.4.4 covers the results of ten-fold cross validation performed on the flyer scenes, with the labels of flyer track to normal track, in order to assess how model performance is affected by choice of segment size. Finally, Section 5.4.5 explains our process for selection of segment size.

5.4.1 Loss of Motion Track Data Points

Two sources of discarded data points exist in our current process. The first is due to allowance of several data points around the actual transition point between activities. This is motivated by the realization that when humans change from one activity, such as running, to another, there is a transitional state which often is difficult for labeling as either activity, as it represents a noisy version of both activities. By removing most of the points associated with this transitional state, we reduce the amount of noise in the overall activity data points. The size of this activity transition is currently very small, at approximately ten frames per transition. The total impact of the discarded points depends on the density of activities per motion track, but tends to be between 0.5% and 3% of the

total available data points. We consider this to be worthwhile, as the discarded points have higher probability of containing noisy data.

The choice of segment size cannot regain any of the discarded points from the activity transitions. Instead, we turn to the second source of lost data points. As mentioned earlier, for an activity of length m, and a segment length n, the number of segments s available in the activity is found by

$$s = \left\lfloor \frac{m}{n} \right\rfloor$$

The remaining points $(m \bmod n)$ are not sufficient to constitute a segment of length n, and must be discarded. As the size of the segment length n increases, the quantity of discarded points which cannot be utilized increases. See Figure 8 for the actual values from our flyer data set of points lost.

In examining the results of greatly increased segment size, it is obvious that the result is catastrophic loss of available data. If we are to select the segment size which minimizes the amount of lost data, we would tend to select extremely small segments, perhaps even of a single data point, if permitted. Data loss is an important aspect to consider in selection of segment size, but should definitely not be the sole factor.

5.4.2 Prior Probability Shift

Segmentation constitutes resampling the available data. In resampling problems, of utmost concern is whether the resulting distribution matches the original distribution. See Figure 9 for the graph of our flyer data set distribution between the flyer person activity segments and normal activity segments, with respect to the chosen segment size. The drastic shift of the distribution as the segment size changes is due in large part to the

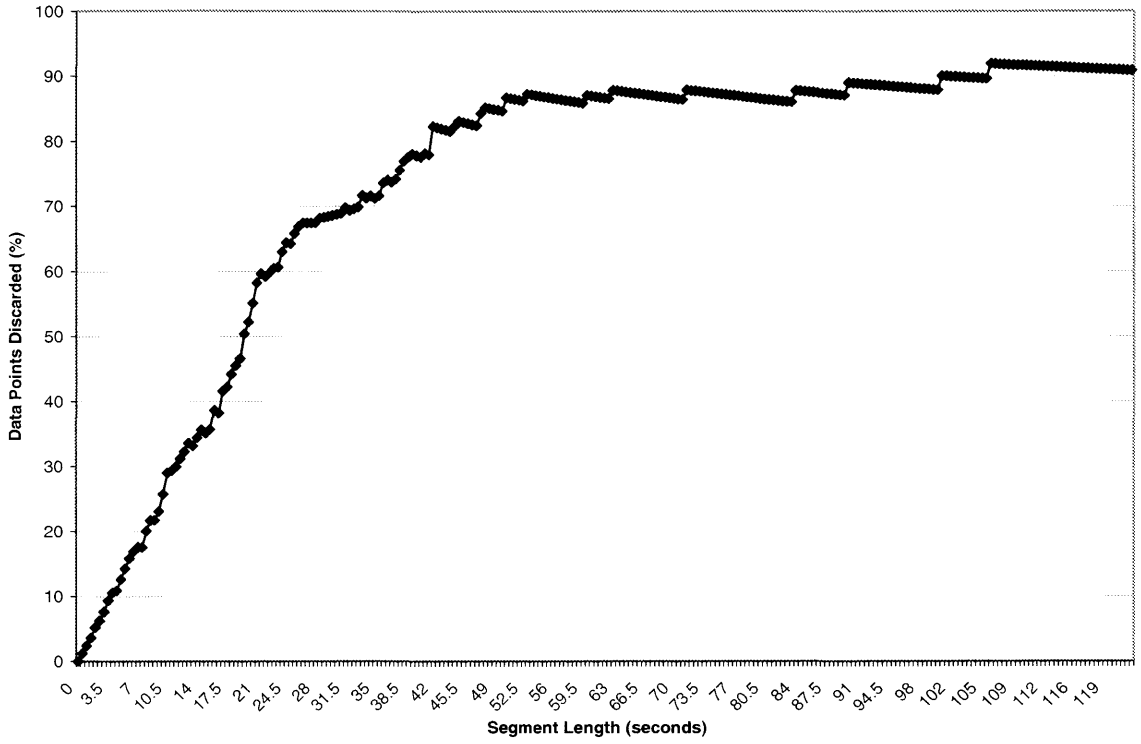

Figure 8: Segment length impact on data loss

For a given segment length, some number of track data points are not used, due to there being insufficient data points available to constitute a complete segment. The quantity of unused data points is positively impacted by segment length increases. Optimizing segment size solely with respect to minimizing amount of data lost would drive segment size down to zero. The longest track is approximately 200 seconds; for segment lengths longer than 200 seconds, 100% of the data must be discarded.

Figure 9: Segment length impact on prior probabilities

As segment length increases, the resulting distribution of the activity classes in the segment set shifts. At the smallest segment length, the distribution is approximately 85% Normal Person class, 15% Flyer Person class. At the largest segment length, the distribution has been reversed to 90% Flyer Person class, 10% Normal Person class. At a given segment length, the prior probabilities reported accurately represent the distribution of normal person segments to flyer person segments for the given time interval; hence, without a specific target distribution to match, prior probability shift is not an adequate metric for segment size selection.

fact that the flyer person is active for a longer duration than individuals who pass through the scene. Hence, at longer segment lengths, the flyer segments dominate the scene. Also, as the other individuals pass through the scene over shorter intervals, their activity tracks are discarded much more swiftly as the segment length exceeds the duration of the activity.

Although the shift of prior probabilities is indeed interesting to note, the question which arises is, which distribution of activities in the segmented set most accurately reflects the true distribution of activities? The answer is that it depends on the length of time being considered. If we truly are interested in activities of a minute length or longer, then activities with duration of less than a minute do not contribute to the distribution. Thus, while the prior probability shift may impact our decision at some point, we do not use it as a factor for optimizing segment length at this time.

5.4.3 Segment Quantity and Feature Dimensionality

The next aspect considered is the quantity of segments produced for a given segment size. We refer to Figure 10, which contains the plot of segment quantity and segment length in data points.

For small segment sizes, extremely large numbers of segments are produced. From the feature extraction methods of Chapter 4, the number of dimensions for representing a segment in feature space is proportional to the length of the segment. Thus, short segment size results in a very dense, compact feature space. Comparably few data points have been discarded, hence the set of segments contains much of the available data points.

As the segment size increases, the quantity of segments swiftly drops, while the dimensionality grows linearly with respect to the segment size. For large segment sizes, relatively few segments are produced, and these reside in a higher dimensional space. The space is much sparser than that of short segments.

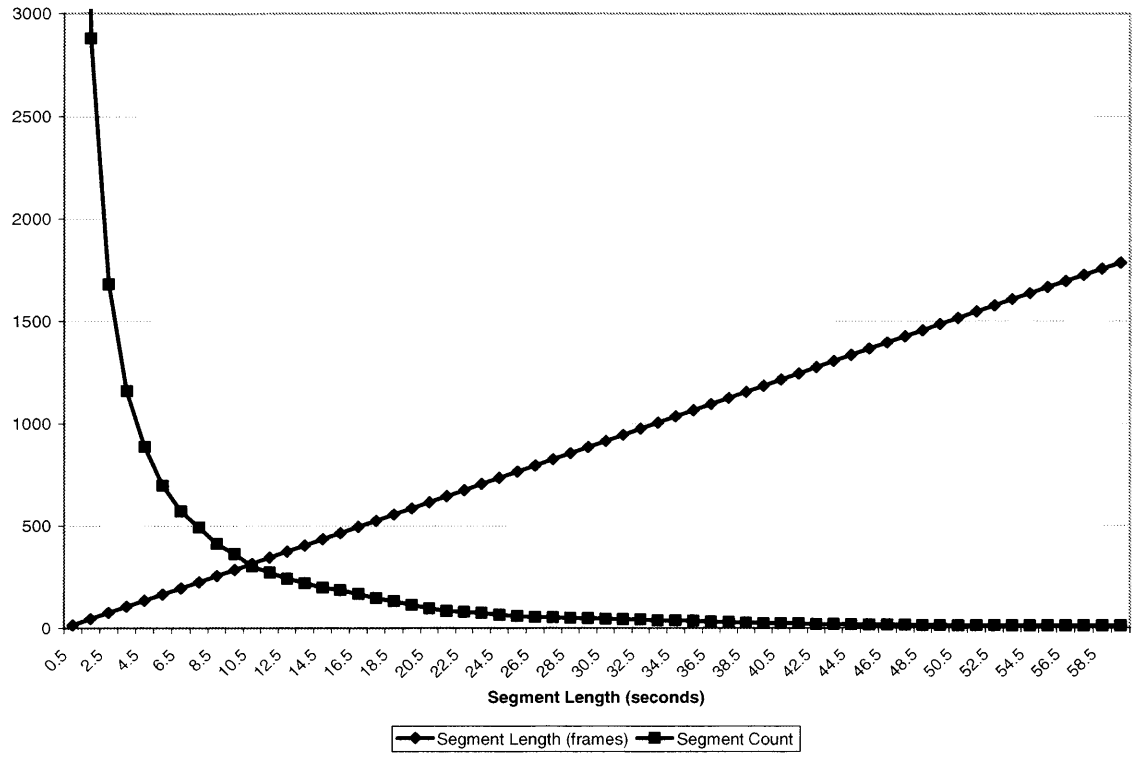

Figure 10: Segment length impact on segment quantity and dimensionality

At very short segment sizes, large quantities of activity segments are generated. As segment size increases, the quantity of generated segments quickly degrades. This is due in large part to the increase in the number of discarded points (see Figure 3). Dimensionality of the feature space is linearly determined by the segment length. Small segments in a comparably low-dimensional space are dense and compact. As segment size increases, the density of the feature space swiftly decreases.

Segment size impacts strongly the density of the input space, as well as the dimensionality of said space. Hypothesis selection in a dense, compact space may be equally difficult as selection in a sparse, expansive space. A trade-off here should also be considered. In order to evaluate hypothesis selection, we perform cross-validation of a decision stump modeling algorithm and a decision tree modeling algorithm.

5.4.4 Cross-Validation Error Rate

A major factor as outlined in the previous subsection is the ability of a modeling algorithm to generate a hypothesis in the given feature space. Chapter 3 introduced the concept of the curse of dimensionality, which states that increases in the dimensionality of the hypothesis space result in exponential growth of the possible decision boundaries. This impacts everything from training time to available model complexity. Models with high bias discard increasing numbers of possible hypotheses, while models with low bias require more time and computation to determine an optimal high-dimensional hypothesis.

In order to evaluate potential impact of the segment size on modeling performance, we use both a high bias/low variance modeling algorithm (decision stump) and a low bias/high variance modeling algorithm (decision tree). For each segment size, we perform ten-fold cross validation on both modeling algorithms to generate an estimate of overall model performance. Results of the decision stump cross validation are presented in Figure 11, the decision tree cross validation in Figure 12, and a comparison of the two in Figure 13.

We note in general that, for moderately small segment sizes, both models perform as expected. The decision tree models tend to perform at a higher degree of accuracy than the decision stump models. As segment size initially increases, performance of both modeling algorithms improves. We note that the decision tree model seems to destabilize, with poor performance and increased confidence intervals appearing at an

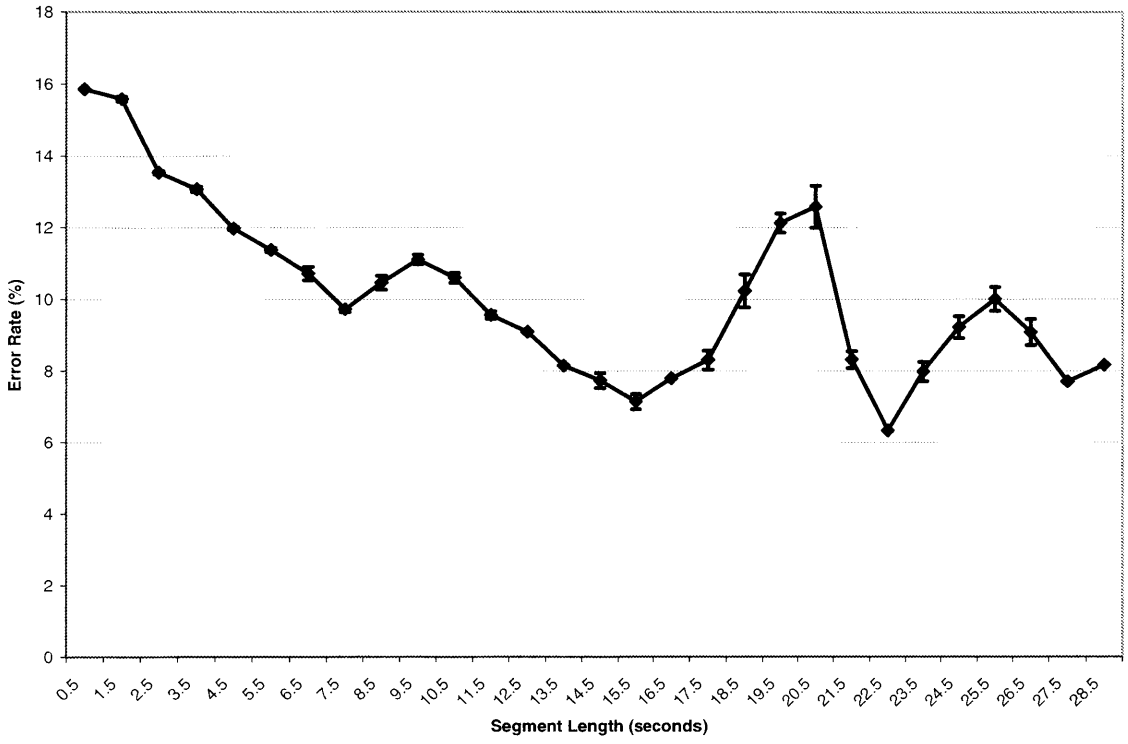

Figure 11: Segment length impact on decision stump performance

As segment length increases, the richness of the features available increases. The result is that overall performance tends to improve. As segment length passes approximately 17 seconds, the confidence intervals are broader, and model accuracy varies more with each change in segment length.

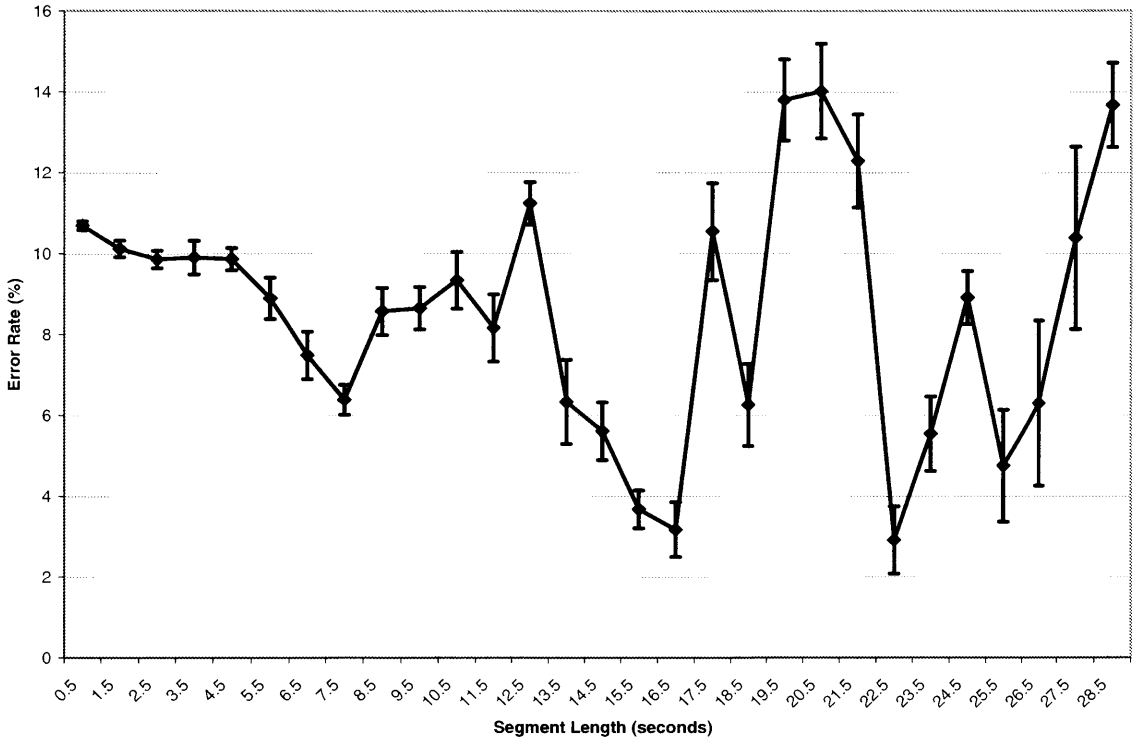

Figure 12: Segment length impact on decision tree performance

As with the decision stump models, performance improves reliably with increased segment length. Decision tree performance appears to be much more instable than the decision stumps as segment length increases. Performance varies widely from one segment size to the next, and the confidence of the true performance mean is lower than at shorter segment lengths.

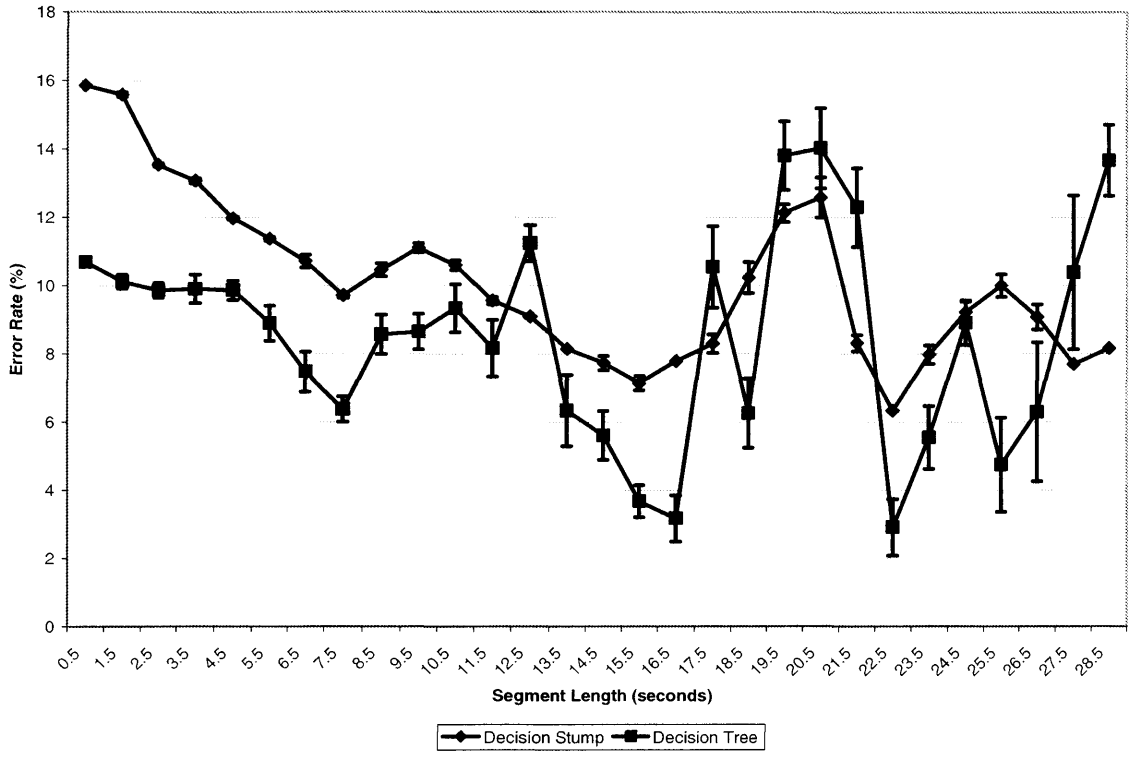

Figure 13: Performance comparison of decision stump with decision tree

In the overlaid graphs, comparison of the two modeling techniques becomes simpler. As the segment length increases to (a), both algorithms achieve a local minimum. As the segment length continues to grow (b), the performance of the decision tree algorithm degrades, as represented by poor performance and widened confidence intervals. The decision stump algorithm also shows signs of instability. Reasons for this are explained in large part by the data of Figure 5. As the dimensionality of the feature space increases, the quantity of possible hypotheses increases exponentially (see the curse of dimensionality, Chapter 3), while the quantity of available segments drastically decreases. The result is that modeling performance improves initially as the feature space expands, but degrades swiftly as the data becomes sparser.

earlier segment size than with the decision stumps. This is due in large part to the vastly increased dimensionality of the feature space, and the decreased density of available activity segments for modeling in said space. The result is that the decision tree algorithm has much more opportunity to overfit the training data, and performance degrades. Decision stump models are less affected due to their high bias toward simplistic models. All machine learning algorithms, both high and low bias, will exhibit similar performance improvements and destabilization.

In determining the segment size, a balance must be reached between the density of the activity segments in feature space (the quantity of segments) and the dimensionality of said feature space (the number of features generated). Examining model performance at different segment sizes seems to indicate a point of improved performance with strong confidence intervals. Identifying such a point would be a decent metric for selecting segment size.

5.4.5 Selecting Segment Length

We have examined several areas in which segment size impacts. These are the rate of data loss, the shift in the prior probabilities, quantity of available segments, dimensionality of the feature space, and cross-validation error rate of both high and low bias models. Of these, data loss rate and cross-validation error rate are most useful for optimizing. Ideally, a parameter could be generated to incorporate aspects from both data loss rate and cross-validation error rate. This is left as a problem for future work. Until such a parameter is identified, we provide a temporary metric for determining a decent segment length.

Of the two parameters identified, computational time for performing ten-fold cross validation is higher than evaluating the data loss function. We note that, for the flyer scene considered in this section, a locally optimal cross-validation error on both classifiers occurs with segment length set to seven and a half seconds. This corresponds

to the position on the data loss curve where the quantity of discarded points exceeds 20% of the total available data points. For experiments in this thesis, we perform segmentation of the activity set with the segment length which yields 20% data loss.

Future experimentation may justify use of a more extensive segment length selection procedure. For the purposes of performing the classification experiments of Chapter 6, this metric is adequate.

We examine the potential for a final adjustment to the segmentation algorithm. It is the addition of motion track information for a given window beyond the considered segment size of data.

5.5 Track Information Window

The overall result of track segmentation is an attempt to maximize the overall segment length with respect to other parameters. The major result of performing the segmentation step is a set of activity tracks which are of uniform length. Feature extraction can successfully be performed on the set of activity tracks, and the tracks become accessible to our machine learning tools. The balance which is reached allows shorter tracks to be considered alongside with long tracks. This is a major accomplishment. However, in some behavioral classification instances, this is not sufficient. The activity segments remain too short for behavioral classification to perform successfully, yet we determine in Section 5.4 that reduction of the available data points would not prove beneficial. An example of this is the training of models on the behavioral classification experiment using the sports tracks, discussed in depth in Chapter 6. We explore the effect of adding information about neighboring segments to the activity segment feature set. The information about the neighboring segments is a *window* of the track data.

5.5.1 Sports Behavior Data Set Revisited

We review the behavioral classification experiment on the sports scenes from Chapter 6. The class labels are the sport which generated the motion track, namely Frisbee, volleyball, and soccer. The motion tracks for the volleyball and soccer data span close to three minutes each, while the tracks for Frisbee are much shorter. The Frisbee players tend to move through the scene rather than remaining within it. From Section 5.4.5, in selecting a segment size, it is necessary to choose the segment size in consideration of all data sets which will be modeled. As a result, the segment size is limited by the Frisbee motion tracks.

When classifying the sports by behavior, the best classifier is only able to attain 34% cross-validation error (see Figure 14). In examining the misclassified activity segments, it is often the case that the misclassified instances closely resemble elements of another sport. An example is the goalie in a soccer game, who often stands in a limited region for a time, much as a volleyball player covering a portion of the space for the team. The classification error seems to be caused by the activity segments being insufficiently large; however, we determined in Section 5.4 that a larger segment length would discard an excessive quantity of the available data and would increase the risk of low bias model overfitting.

We examine the impact of allowing the features of an activity segment to contain summary information from a larger window of the source motion track.

5.5.2 Windowing Method

By adding features from a window on the original motion track, where possible we expand the amount of information available to the machine learning algorithms for modeling the motion track. The features we add are already extracted for each motion

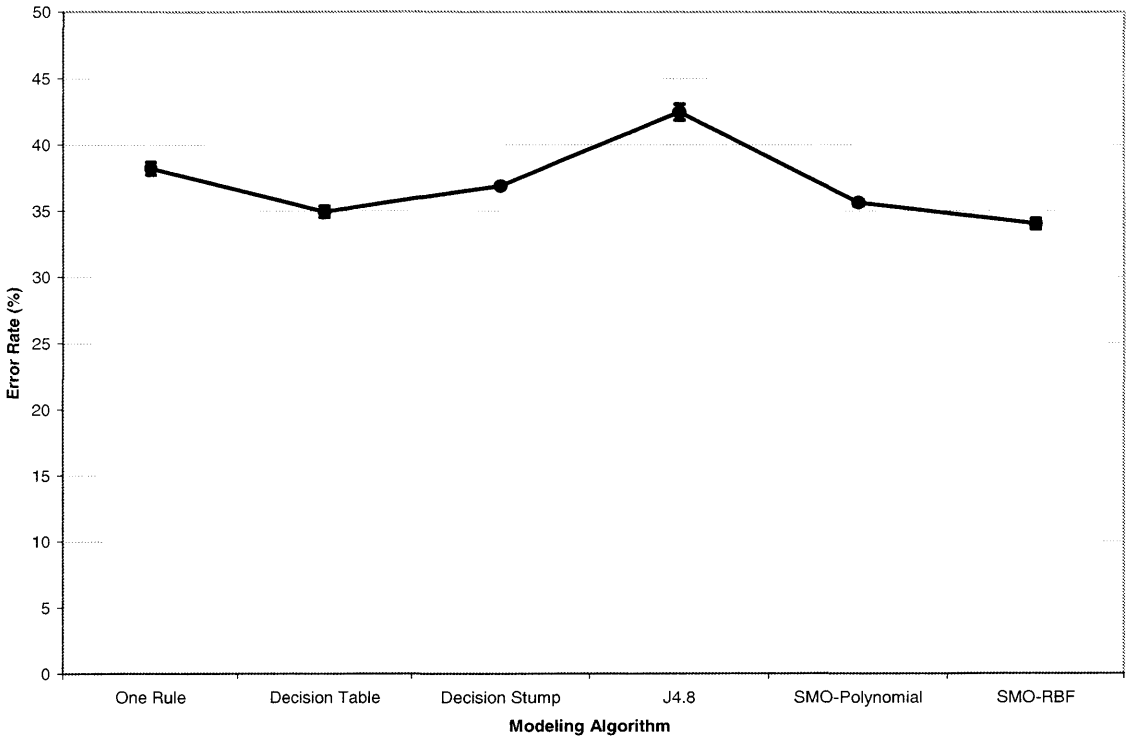

Figure 14: Sports behavior modeling, basic performance

This graph also appears in Chapter 6, under discussion of the classification experiments performed on the sports behaviors. The data set consists of motion tracks labeled by the sport which generated the motion track. Performance of all classifiers is poor. Our claim is that segment length is not sufficiently long to account for enough of the motion patterns for adequate comparison, but segment length cannot be extended without added loss of available data. Windowing is introduced to improve on performance by extending the features for a segment with generalized information of adjacent segments from the same motion track.

track. These are the minimum, maximum, and average values of the turning and speed over the length of a given segment. These parameters are introduced in Chapter 4. The values are added to the feature list for the segments both preceding and following the segment of consideration.

For segments which occur at the beginning or end of the motion track, the necessary neighbors for gathering the window features do not exist. In such cases, the values are left missing. Of the total number of activity segments in the sports scene, 12% have missing window features for the previous neighbor, and 12% have missing features for the following neighbor. Thus, a large number of the activity segments have been augmented with additional information of a larger window of their source motion track, and shorter tracks are not discarded.

Chapter 3 discusses approaches to solving the missing value problem in machine learning. For simplicity, we fill the missing values with the mean of the attribute. Alternate methods may have different outcomes. Segmentation was originally considered to solve the problem of comparing motion tracks of varying length. Through the combination of segmentation and windowing, we have reduced the unsolvable problem of comparing motion tracks to the task of replacing missing values.

Windowing provides information over a larger scale than the segment size. This is of benefit to longer motion tracks, yet does not penalize shorter motion tracks. We examine performance of classifiers on the sports behavior scene of classification experiment 5 to show that the addition of windowing features successfully improves model performance.

5.5.3 Windowing Results

Figure 14 contains the initial results from Chapter 6. Figure 15 shows the windowed segment performance, and Figure 16 displays both the original performance and the windowed performance.

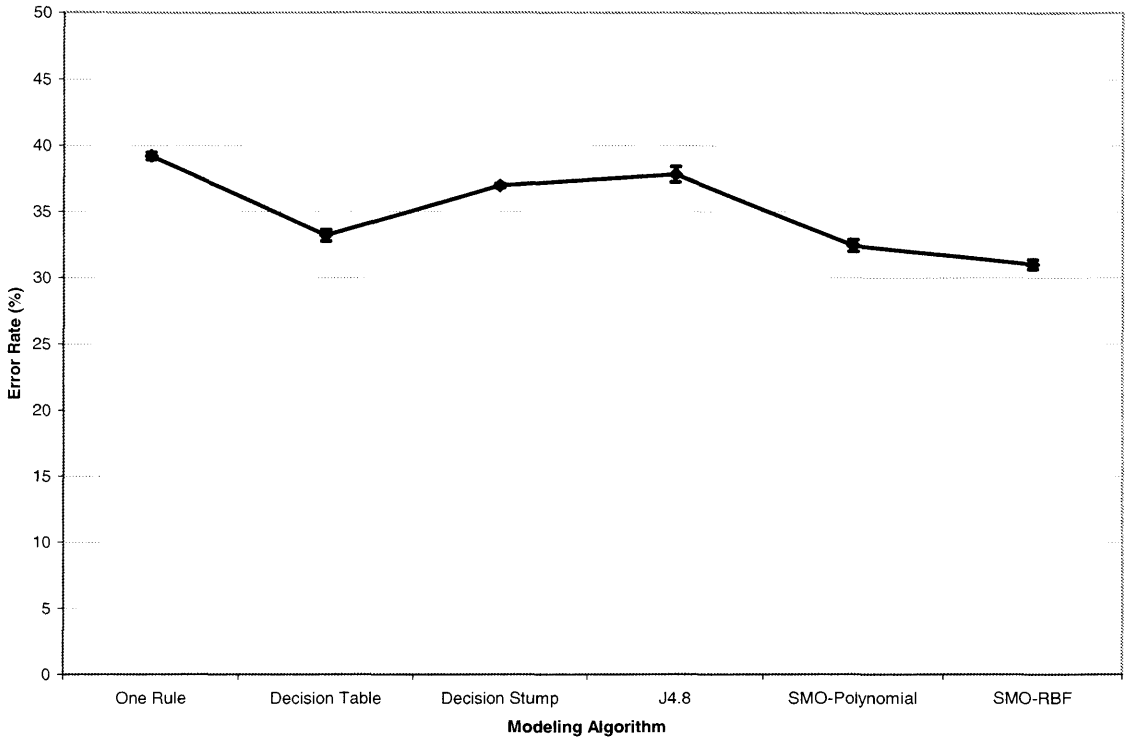

Figure 15: Sports behavior modeling, windowed performance

As with Figure 9, the various modeling algorithms are evaluated using ten-fold cross validation. The only change from the experiment which generated the data of Figure 9 to the experiment which generated the data of this figure is the inclusion of the windowing features discussed in Section 5.5. These features are gathered from a window of the segment size before and after the given segment, and consist of the minimum, maximum, and average values attained for the speed and turning features.

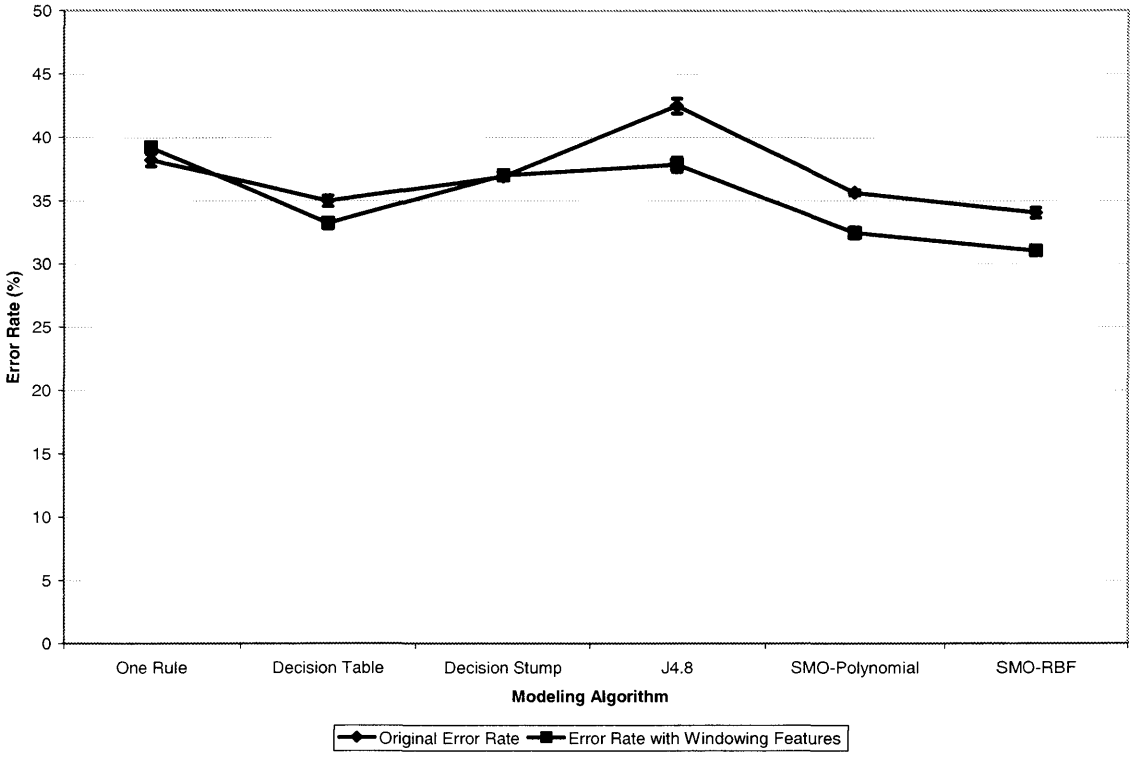

Figure 16: Sports behavior modeling comparison

On comparison of the performance before and after inclusion of the windowing features, definite improvement can be seen. Though the improvement is less than 5% error, it is still a significant change seen in all but the high bias modeling algorithms. For one rule and decision stumps, only the most dominant attribute is considered. Thus the windowing features are ignored, and no improvement is achieved. The lack of improvement is due to the learning algorithm, not the added metrics. Further experimentation may yield greater improvements over the base features.

We first note that for one rule and decision stump classifiers, performance does not improve. Both classifiers are highly biased, inasmuch as they select a single attribute on which to base their classifications. The selected attribute did not change with the addition of windowing features. For these classifiers, windowing will not show improvement unless a feature added through the window is more dominant than the feature previously utilized by the classifier.

For all other classifiers, performance on the windowed segments shows an improvement of 1.7% to 4.6%, an average of 3.1% improvement over the original activity segment performance.

While the improvement amounts are not extreme, they do indicate a significant performance change from the addition of a small amount of data. Further experiments are needed to optimize the windowing technique for added performance improvement. Possible changes might include determining additional features from each window which increase the performance, exploring how the treatment of the missing values affects performance, and potentially varying the window size. As with the segmentation process, these added methods must take into account the amount of data which must be estimated by the missing value replacement algorithms.

Segmentation overcomes the problem of comparing motion tracks of varying lengths. It balances lengthening segment size to increase the richness of data provided for each instance, with shortening the segment size to reduce the quantity of data points which are unusable. The windowing method shows promise for being a powerful counteragent to the downsides of short segment length. We now examine more fully the classification experiments performed on our data sets in Chapter 6.

CHAPTER 6: MOTION TRACK CLASSIFICATIONS

With the framework of machine learning theory from Chapter 3, and the segmentation techniques from Chapter 5, we now discuss the application phase of our project – modeling of single track activities. The first modeling approach tests a simpler metric, that of modeling motion-based classifications. We then turn to modeling a more complex classification, even behavior-based classification. Section 6.1 describes the classifications in detail. Section 6.2 evaluates performance of generated models on the classification tasks. Finally, Section 6.3 summarizes the experiment results.

6.1 Data Set Classifications

Our data sets are introduced in Chapter 4. The first set contains a flyer distribution activity, and the second contains various sports. We first consider motion classifications of these data sets in Section 6.1.1, followed by behavioral classifications in Section 6.1.2.

6.1.1 Motion-Based Classifications

The first set of classifications we consider are motion based, meaning the class labels are assigned with a simple measure of the motion in mind. The metric is primarily based on overall speed of the segment. As human labels are our primary target, we focus on human-assigned labels for each of the experiments performed. This adds complexity, as a person assigning speed-based labels uses an arbitrary and ever-changing threshold to determine which label to assign. Matching the classification method is compounded as a result, though performance is still high.

For the flyer scenes, an individual distributes flyers to other individuals traveling through the area. The flyer person meanders about the scene, occasionally intercepting others. The first motion classification experiment matches a given activity segment to the label of meandering or normal activity. A meander activity is loosely defined as any activity segment where the individual is stopped completely, or is moving slowly in an indirect path.

The second motion classification experiment is also based on the flyer scene. It moves to a higher number of activities being distinguished. These are stopped, slow, and normal paced, as well as all cars and bicycles grouped separately. A speed-based threshold approach performs equally well here, as all that is necessary is an additional threshold to differentiate slow activities from stopped activities. With this experiment, we add in more of a behavioral classification in the car label, as stopped and slow car activities are not extracted from the overall car motion track.

The third motion classification experiment switches to the sports scenes. Class labels distinguish the basic motion pattern of a player between standing, walking, or running. Added complexity arises due to the human-based class labels, as the dividing thresholds between class labels is arbitrarily enforced by the person generating the labels. There is likely a shift in class thresholds not only within sports, but across sports as well. For example, a person who is labeled as walking in a soccer game might be labeled as standing in a volleyball game, simply because of the context in which the motion takes place. The added complexity of human-based class labels provides a more realistic example of our target application.

Motion-based classification results are presented in Section 6.2.1. Having considered the motion-based classification experiments, we turn to the behavioral classification experiments.

6.1.2 Behavioral Classifications

The target area for consideration is behavior modeling. Behaviors are much more difficult to express explicitly than motion-based classes, as a single threshold cannot encapsulate all the variation which exists in a behavior. In generating a behavioral label, often an entire motion track is placed in a behavioral class. This is not to preclude the possibility of an individual exhibiting multiple behaviors over the duration of their motion track; rather, it is a result of the scenes which generate our motion data. We have no occurrences of a traveling person stopping to distribute flyers, nor of a soccer player becoming a Frisbee player.

The first behavioral classification experiment classifies the flyer person versus all other individuals in the scene. One might expect this problem to be very straightforward, as the behavior of the flyer person is distinct from other actors in the scene. On the other hand, there are times when the flyer person exhibits behavior which is similar to normal traffic, as well as times when the traveling individuals stop or move slowly.

The second behavioral classification experiment generates class labels of the sport being performed (e.g., Frisbee, volleyball, and soccer). The underlying structure of a track set is a result of the goals and rules which are specific to the sport being played.

Behavioral experiment results are presented in Section 6.2.2.

6.2 Model Performance Evaluation

To evaluate performance of machine learning classifiers on the experiments outlined in Section 6.1, we perform ten times ten-fold cross validation from Chapter 3 on a set of classifiers. Our classifier set consists of one rule (high bias, low variance), decision tables (low bias, high variance), decision stumps (high bias, low variance), decision trees (low bias, high variance), SVMs with Polynomial Kernel of power 2

(medium bias, medium variance), and SVMs with Gaussian RBF Kernel of $\sigma = 0.01$ and regularization constant $C = 1000$ (low bias, medium variance).

We examine the cross-validation error rates of the set of classifiers on the motion-based classification experiments in Section 6.2.1, and on the behavioral classification experiments in Section 6.2.2

6.2.1 Motion-Based Classification Results

The first set of classification experiments we examine is motion-based. Most of these classifications could be represented using a threshold on one or two of the input features. The experiments are outlined in Section 6.1.1.

The first motion-based classification experiment distinguishes meandering behavior versus all other behaviors (simply referred to as normal behaviors). Meandering is loosely defined as a slow paced, sometimes stopped, curvy motion. Model performance appears in Figure 17. The majority of the models perform at around 2% error. The SVM with RBF model has the best performance. All models perform well on the activity segments except for one rule. This preliminary data set shows that our activity segmenting and modeling process performs well.

For experiment two, we develop models of different paced activities, and the cars and bicycles in the scenes. The activities are broken into groups of stopped, slow, and walking. Model performance appears in Figure 18. Most models perform around 6% error. Decision stumps perform extremely poorly, due to its selection of a single attribute, and a single threshold; it cannot represent more than two classes. One rule out performs all others by far. As initial labeling of the activities was performed by a human user, the differences between what constitutes a slow activity versus a stopped or normal-paced activity is not a hard threshold, though the threshold approach of one rule performs well. As for the cars and bicycles, a concern was that their activity signature would be

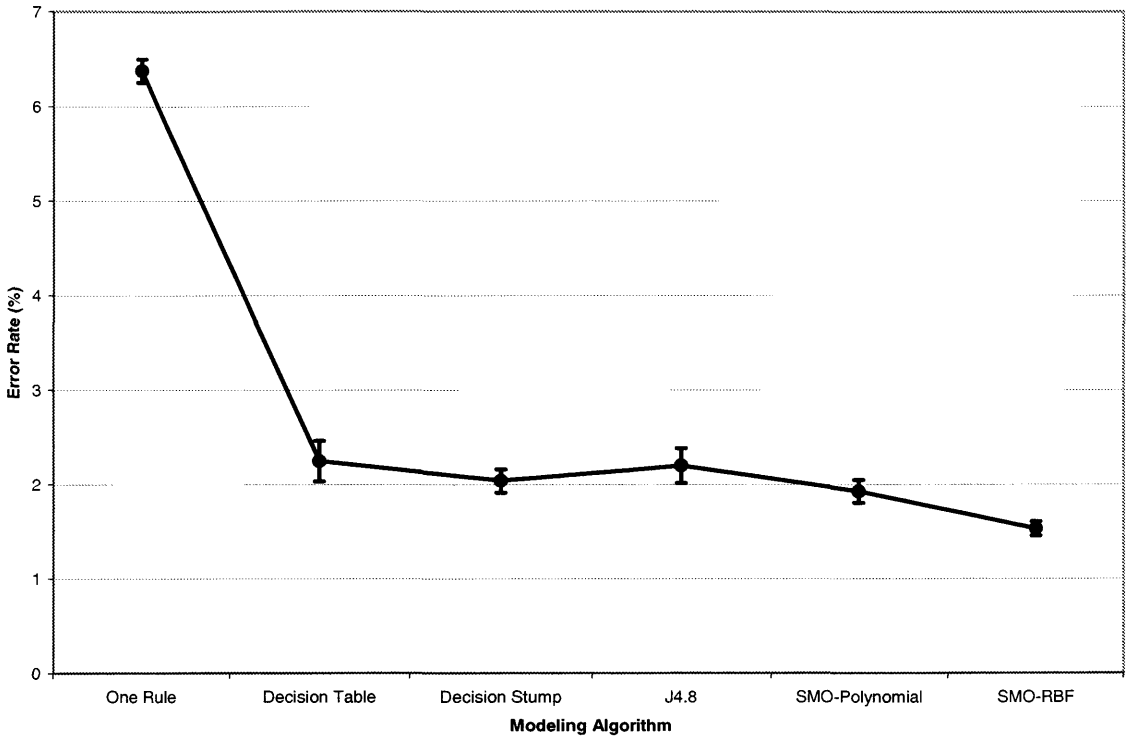

Figure 17: Performance on flyer scene, meander vs. normal

The majority of the models perform comparably. The class label is loosely defined in terms of the motion of the individual. With the exception of the one rule model, all of the modeling algorithms perform well on classifications based on motion.

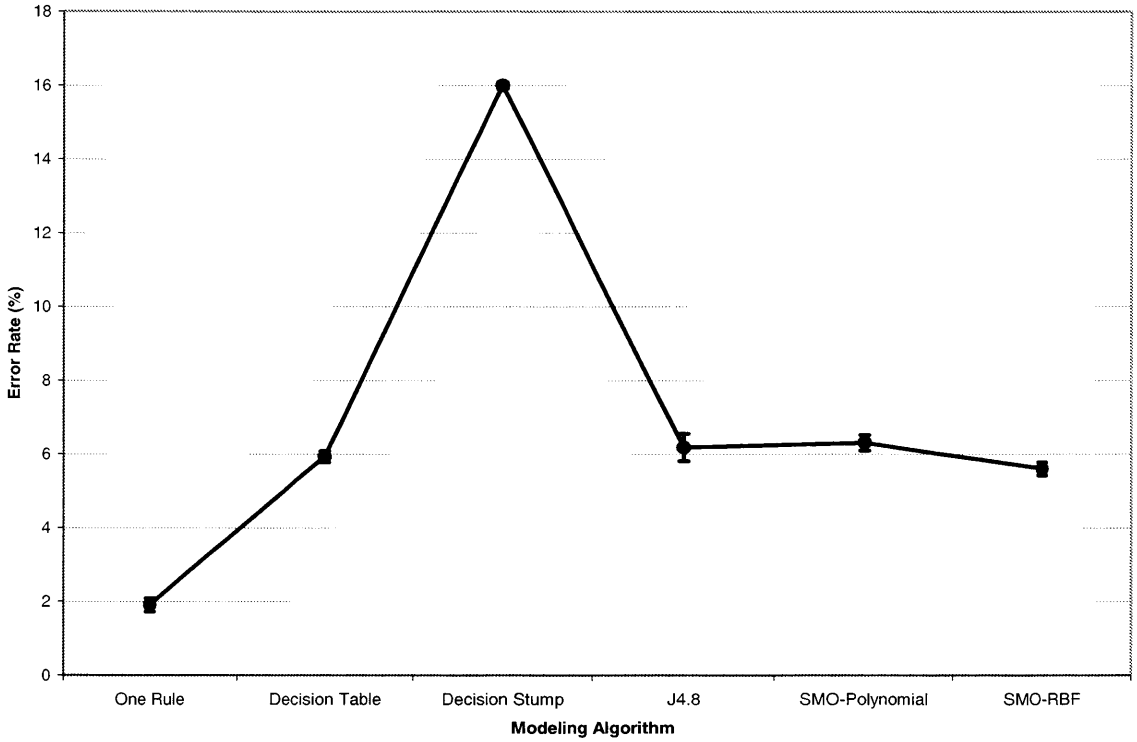

Figure 18: Performance on flyer scene, multiple speed labels and cars

Model performance is centered around 6% error rate. One rule outperforms all other classes with a set of thresholds based on average speed. Decision stumps have high bias in that they can only represent two-class decision surfaces, resulting in poor performance on this data set. The remaining models are comparable in their performance.

too easily confused with a person, as the vehicles stop and move slowly through the intersection. On the contrary, the majority of the cars and bicycles are correctly classified. We attribute this in large part to the segment length selection, where the greatest segment length possible is selected.

Our final motion-based classification experiment groups motions from the sports scenes as standing, walking, and running. Performance is reported in Figure 19. As with the second motion experiment on the flyer scene, decision stumps perform poorly due to the bias against decisions involving more than two classes. Other than the decision stump model, the remaining models perform between 6.6% and 9.3%. The best of these are the SVMs; the worst is the decision tree. Performance is slightly poorer than with the second motion experiment. The boundary between the activities was again allowed to be arbitrarily defined by the human user performing the activity labeling. This may be part of the impact on the performance.

For motion-based classifications, performance on all scenes is on average less than 10 % error rate. Our segmentation and feature selection process works well at modeling motion-based classifications. We now review the results of the behavioral classifications.

6.2.2 Behavioral Classification Results

Behavioral classifications represent a more difficult task for modeling, as the person generating the labels often is allowed to consider information beyond that which can be captured by features. The two experiments consider the behavior of the person distributing flyers from the flyer scenes, and people from different sports (e.g., Frisbee, soccer, and volleyball). No attempt is made to explicitly define the activities; this task is left to the modeling tools to discern.

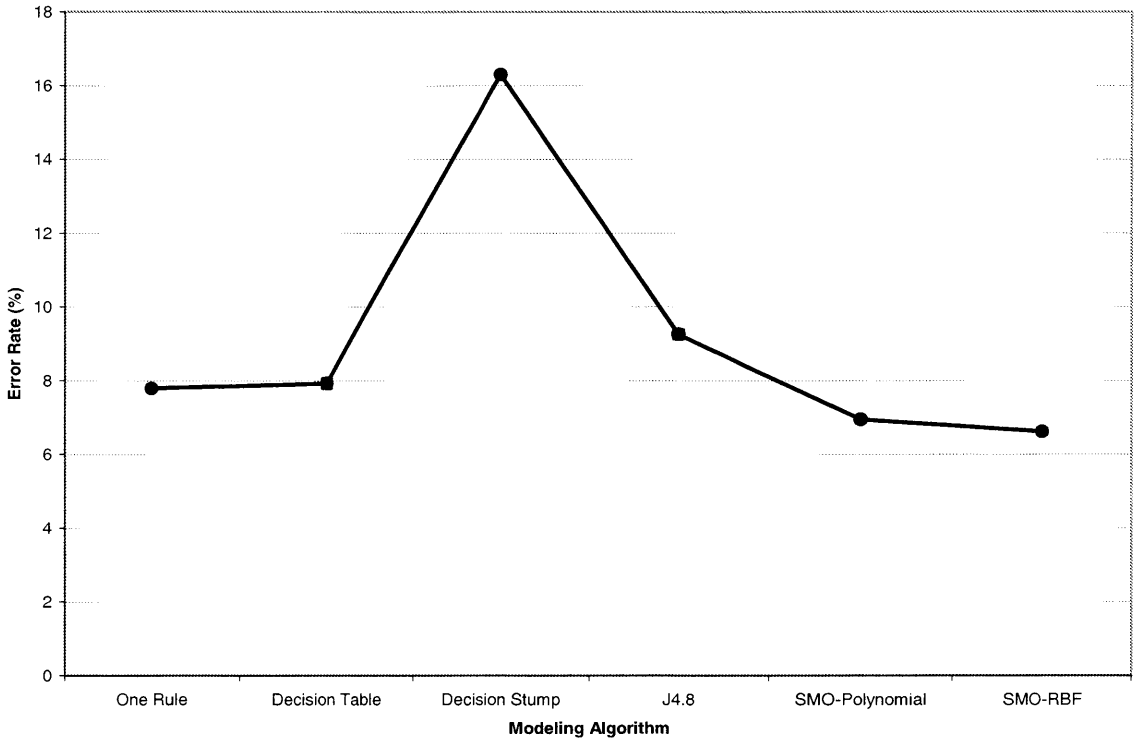

Figure 19: Performance on sports scene, motion classification

The motion classifications of this scene consist of standing, walking, and running activities. As noted earlier, decision stump models are highly biased toward two-class classifications. This explains the poor performance by the decision stump model on this data set. Of the other models constructed, performance is approximately 8% error rate.

For the flyer scene, we model the behavior of the flyer person against all other behaviors in the scene (e.g., other people, cars, etc.). The flyer person has a signature which tends to meander about the scene, occasionally moving to intercept others, and occasionally stopping to wait. Other individuals tend to travel at a constant pace through the scene, though at times they slow or stop. Each track is labeled as either a normal behavior or a flyer person behavior. Modeling results are displayed in Figure 20. The difference among modeling algorithm performances is slightly more spread out. The range is from about 7.5% to 12.5% error. The best models are produced by one rule and decision tree. The worst modeling algorithm at this task was the SVM constructed with the polynomial kernel. In general, overall modeling performance is promising. The modeling techniques were able to represent the ill-defined activities fairly well.

In the sports scene, the behavioral labels are generated with added knowledge of the source scene. Class labels reflect the sport which the individual was participating in when their motion was captured. The sports are Frisbee, soccer, and volleyball. Signatures of the different sports should be sufficiently different, as the motion patterns are strongly influenced by the underlying rules and goals of each sport. Model performance is recorded in Figure 21. It becomes immediately apparent that performance on this data set is far worse than those previously considered. Average model performance is 37% error. The best model is the SVM with the RBF kernel, and the worst is the decision tree. Chapter 5 discusses use of added information from a window into the original track data for improving performance on this data set. This metric allows information from neighboring segments to be considered by the model in evaluating a given segment. Preliminary experimentation shows approximately 5% decrease in error rate on all but the most simplistic of the modeling algorithms. Actual increase attainable has not yet been ascertained.

In general, behavioral classification performs well on differentiating the flyer behavior from other traffic behaviors. Sports differentiation proved more evasive, though

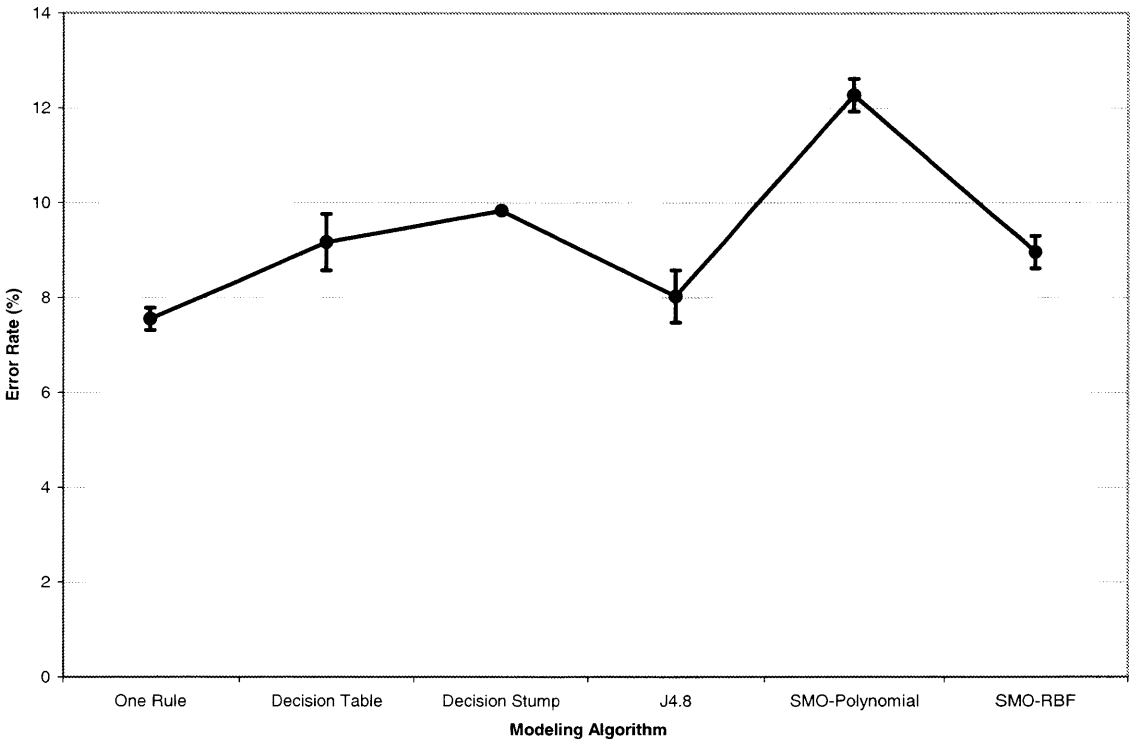

Figure 20: Performance on flyer scene, flyer behavior vs. other behaviors

Classification labels are assigned by the activity of the individuals in the scene. One of the individuals in the scene is distributing flyers to others who pass through the scene. As the nature of the flyer person's behavior is different, their motion track has different properties. This experiment indicates that behavioral modeling is a viable approach to scene analysis.

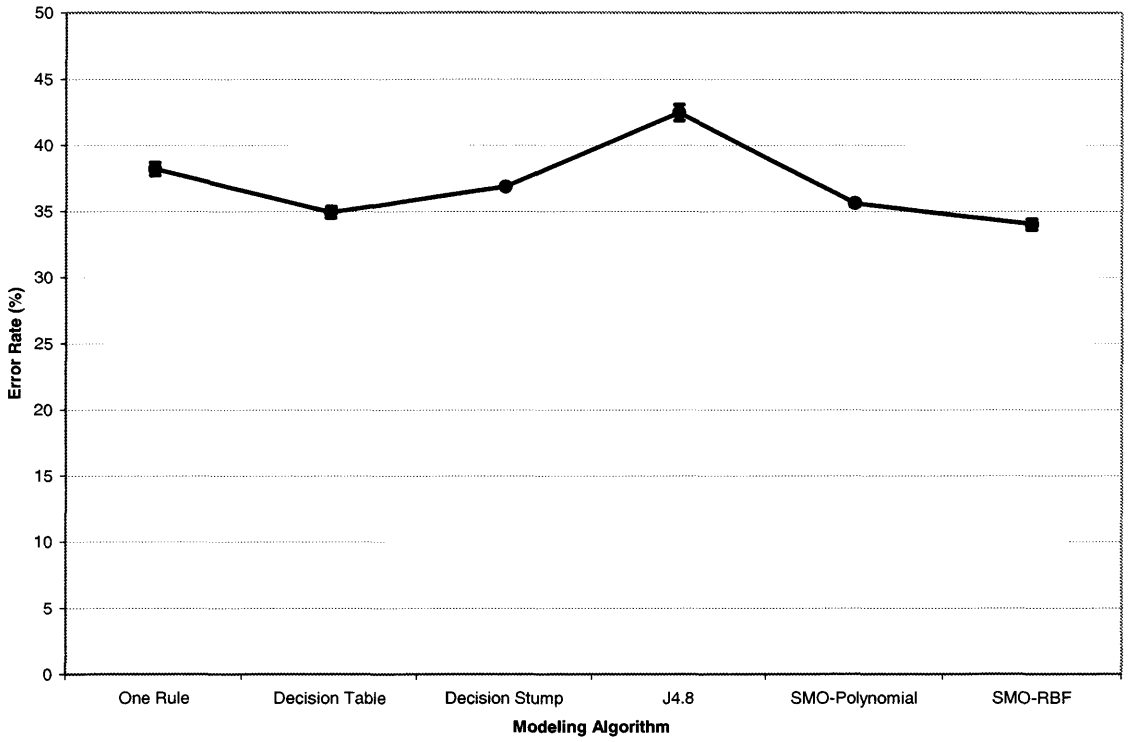

Figure 21: Performance on sports scene, behavior classification

Class labels are determined by the sport from which the motion track was captured, namely Frisbee, soccer, and volleyball. Average error rate of all models produced on these sports scenes is 37%, indicating poor overall performance.

methods, such as windowing, show promise for improving overall performance. We now examine results from both motion-based and behavioral classifications.

6.3 Discussion

The results of the various experiments provide insight into the potential for behavioral classification of motion tracks. In the motion-based classifications, a simple metric was selected based loosely on the average speed of the individuals being monitored. Specific thresholds separating the activities could have been used, but the arbitrary assignment of categories by a human user requires more flexibility from the modeling tools. These experiments stand as the preliminary proof of concept that such classifications on motion tracks are attainable, and validate the preprocessing system described in Chapters 4 and 5.

The behavioral classification experiments consist of an abstract label being assigned with no regard for which of the available features might best help distinguish between class labels. In the flyer scenes, the flyer person tended to move in a different pattern than other travelers. This was sufficient to recognize the activity in cross-validation experiments. For the sports scenes, the assignments proved much more difficult. The labels consisted of the sport which the individual was participating in. No other information was utilized in assigning the class label. The resulting models attempt to identify what sport is being performed based on the motion pattern of the individual.

Preliminary work from Chapter 5 indicates that there is potential for improvement in modeling behavioral data by the inclusion of windowing data. The improvement arises from an extension of the information available where track length exceeds segment length.

Despite the poor performance of the machine learning algorithms on the sports behavioral classifications, the overall performance of our approach to motion and behavior classification is a success. We consider areas for future improvement.

CHAPTER 7: FUTURE WORK

Having completed the discussion of the experiments (Chapter 6), we explore possible avenues for future development of this work. Three areas are identified. Section 7.1 describes analysis methods of more intricate information contained in a motion track set. Section 7.2 examines behavioral classification techniques. Section 7.3 moves to a higher level of modeling, where scene-level classifications become possible based on low-level motion and behavioral analysis of the scene components.

7.1 Expanding the Motion Track Data

In looking to expanding on the work contained in this thesis, the first extension would be in the features considered for classification of behaviors in a scene. These include reference object information, scene model incorporation, and identification of interactions among tracks.

In discussing our data sets in Chapter 4, we note the presence of reference objects in the scene. For our project, these were predominantly the Frisbee and balls for the sports games. As our efforts center on modeling human behaviors, we discarded the motion track of the reference objects. Added richness of the modeling of an activity should be plausible with incorporated information of the objects of reference. These could be an object which is being moved by the actors in the scene, such as the ball in the sports scenes, or a location, as with the opposing goals in the scenes. By understanding how an individual moves with respect to a reference object, added classification could be performed.

Similarly with the reference objects, incorporation of track data with a scene model would aid in understanding the behaviors of moving objects. To embed our

system into a scene model, positional information would necessarily be of greater importance. For the current level of analysis performed in this thesis, it is sufficient to discard the positions of the motion tracks, and center analysis on the motion of the track.

In a similar thread, modeling of track behaviors could be improved with integration of interaction-based information. Much work has been done in recognition of human interactions. Some are as simplistic as selection of a distance metric, below which all actors are interacting, or more complex metrics for determining both whether an interaction is occurring and, if so, what it is. Much effort has been invested in dual-track interactions. Multiple track interactions are of greater interest, moving toward large group behavior modeling and analysis.

The extensions discussed in this section center on extracting more informational content from the set of motion tracks. Next we consider future experiments in activity classification.

7.2 Activity Classification Extensions

In modeling motion behaviors, we have utilized activity labels which are user-provided. Possible extensions include use of role-based classifications, analysis of the impact of the number of independent, distinct activities in a scene, and exploration of a much more extensive hierarchy of activity information.

Our work has focused on behavioral classifications. As a direct result, we considered a limited subset of potential classifications, namely the motion primitive the actor is performing (e.g., standing, walking, running), and behavioral labels (e.g., flyer person, volleyball player). Expansion on the behavioral classifications is possible to identify role-based behaviors in a scene. An example of this would be identification of positions of players in a sport, such as the goalie or the forwards in soccer, or the quarterback in football. As the players take on different roles, their behavioral signature is distinguishable from other roles.

Similarly, our limited subset of activities will be expanded to a more structured hierarchical conceptualization of activity. For differing scenes, we generate a set of classifiers specific to the activities we wish to identify for the scene. We will consider methods through which activity models can be automatically generated and proposed to a human user, adjusted and generalized through experience over time, or even discarded over time. In such a manner, an activity recognition system will be generated.

Our current approach treats all activities in a scene as independent of all other activities. Earlier in this chapter, we discussed modeling interactions of motion tracks. In moving toward group behavior modeling, the assumption that an activity occurs independently of other activities is naïve. We will explore an improved method for considering multiple activities present in a scene simultaneously.

Having examined several extensions in activity classification techniques, we now move to scene level analysis techniques.

7.3 Scene Analysis

A higher goal of our work in this thesis is eventually developing modeling techniques which would be able to classify groups of actors. Our present work focuses on classifying an object based on how it moves. An obvious extension is to classify a group of objects based on how they behave individually. This constitutes a shift to a higher level of analysis than single tracks. Such methods are applicable at the scene level, examining all tracks in the scene.

A preliminary experiment of interest is whether sufficiently adequate behavioral classifiers from the work of this thesis will aid in higher-level scene classifications. As discussed earlier, scene-level classifications seek to identify the scene based on how its components behave. Our work develops classifiers to identify how components behave. In this manner, the lower-level work of our classifiers is used either as augmenting scene features, or as the sole input to a higher level classifier.

We have focused much of our discussion in this thesis on the selection of segment lengths for comparing motion tracks, and a similar question arises when we move to scene-level analysis. One modeling technique consists of generating scene-wide features at a series of time scales, and then performing scene-level comparisons at all time scales simultaneously. We have examined construction of classifiers for single track data at various time scales, though we always consider selecting a single time scale and using it for all comparisons. It would be possible to construct classifiers for each time scale, and to perform analysis of the scene across all time scales.

A long-term future project analyzes output of a set of classifiers monitoring a scene over an extended duration. All experiments in this project are performed over source scenes consisting of approximately three minutes duration, hence are not conducive for such analysis. With a much longer duration of scene information, time series analysis procedures become relevant. These include identification of time-varying patterns, anomalous shifts in activity levels, and outlier detections. Activity-based streams would be much more compact for storing than all images captured from a video surveillance system.

The work of this thesis expands several areas, and generates many questions in others. We have outlined but a few areas for future development. These are extensions to the information extracted from the motion track data, to the activity classification techniques, and a shift to scene-level analysis of motion track data.

CONCLUSIONS

The work contained in this thesis unites several key aspects of motion track and behavioral analysis. The most prominent of these are the segmentation work of Chapter 5, the windowing method for expanding track information from Chapter 5, Section 5.4, and the development of behavior-based classifiers from Chapter 6.

First, in preparing a set of motion tracks for modeling, it is impossible to compare two motion tracks with varying lengths. This led to segmenting the tracks uniformly at a time interval referred to as the segment length. We examined the impact that selection of the segment length plays in several areas, including the segment quantity for a given track set, data loss as segment size increases, shifts in the prior probabilities of the output class, and degradation of model performance as seen through ten-fold cross validation. We found that the impact of this parameter is high and our results are sensitive to the choice of segment length. Without understanding the importance of this single independent parameter, segment length selection could be selected with high bias, rather than with deeper understanding of all implications of the choice.

Second, after performing segmentation, performance was poor in modeling sports behaviors. As segmentation limits the richness of the amount of data sent to a classifier for a given segment of a given track, we developed the concept of windowing the input data for a track segment. Using windowing, track information from a larger scale can still be considered by a machine learning algorithm where such information is available. In cases where the window size exceeds the track length, the window information is left missing, to be filled in by a standard missing value approach. Performance on the sports behavior set showed improvement with use of a window three times the size of the segment itself. Further experimentation should be done to determine flexibility of the window of track-level information.

Third, we succeeded in our overall goal of developing motion-based classification models. Models of the flyer behavior performed reasonably well on activity data from another scene. Other user-imposed classifications, including speed-based classes, were fairly straightforward to match. Sports behavior modeling improved slightly with the inclusion of windowing information. The behavioral classifiers will improve higher-level scene analysis methods in future work.

The major contributions of this work are the analysis of segmentation length impact, the development of the windowing technique for improved segment comparison, and the generation of behavior-based motion models.

REFERENCES

Aggarwal, J. K., & Cai, Q. (1999). Human motion analysis: A review. *Computer Vision and Image Understanding, 73*(3), 428-440.

Ben-Arie, J., Wang, Z., Pandit, P., & Rajaram, S. (2002). Human activity recognition using multidimensional indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(8), 1091-1104.

Bennewitz, M., Burgard, W., & Thrun, S. (2002). Learning motion patterns of persons for mobile service robots. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (IRCA),* 3601-3606. Pittsburgh: Carnegie Mellon University.

Bobick, A. F., & Ivanov, Y. A. (1998). Action recognition using probabilistic parsing. *1998 Conference on Computer Vision and Pattern Recognition (CVPR),* 196-202.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining, 2*(2), 121-167.

Cannady, J. (1998). Artificial neural networks for misuse detection. *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98),* 443-456.

Cielniak, G., Bennewitz, M., & Burgard, W. (2003). Where is ...? Learning and utilizing motion patterns of persons with mobile robots. *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03),* 909-914.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods.* Cambridge, MA: Cambridge University Press.

Diffie, W., & Landau, S. (1998). *Privacy on the line: Politics of wiretapping and encryption.* Cambridge, MA: MIT Press.

Efros, A. A., Berg, A. C., Mori, G., & Malik, J. (2003). Recognizing action at a distance. *IEEE International Conference on Computer Vision,* 726-733.

Farin, G. (2002). *Curves and surfaces for CAGD: A practical guide* (5th ed.). San Diego, CA: Academic Press.

Haritaoglu, I., Harwood, D., & Davis, L. S. (1998). W4: who, when, where, what: A real time system for detecting and tracking people. *Proceedings of the third International Conference on Face and Gesture Recognition,* 222-227.

Hongeng, S., Bremond, F., & Nevatia, R. (2000). Representation and optimal recognition of human activities. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 1818-1825.

Ivanov, Y., Stauffer, C., Bobick, A., & Grimson, W. E. L. (1999). Video surveillance of interactions. *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR) Workshop on Visual Surveillance,* 82-90.

Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics, 14,* 201-211.

Martin, S., Nelson, B., & Sewani, A. (n.d.). APE: Adaptive prevention environments: A framework for adaptive virus detection and containment. Retrieved November 6, 2004, from http://www.cs.berkeley.edu/~steve0/papers/APE_martin_nelson_sewani_262A_03.pdf

Mitchell, T. M. (1997). *Machine learning*. Portland, OR: McGraw-Hill.

Mohan, A., Papageorgiou, C., & Poggio, T. (2001). Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Learning, 23*(4), 349-361.

Müller, K., Mika, S., Rätsch, G., Tsuda, K., & Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks, 12*(2), 181-202.

Nevatia, R., Zhao, T., & Hongeng, S. (2003). Hierarchical language-based representation of events in video streams. *Proceedings from 2003 IEEE Conference on Computer Vision and Pattern Recognition Workshop, 4,* 39-46.

Nieto, M. (1997). Public video surveillance: Is it an effective crime prevention tool? Retrieved November 6, 2004, from http://www.library.ca.gov/CRB/97/05/

Niyogi, S. A., & Adelson, E. H. (1994). Analyzing and recognizing walking figures in XYT. *Proceedings of IEEE Computer Vision and Pattern Recognition* 469-474.

Oliver, N., Rosario, B., & Pentland, A. (1998). Statistical modeling of human interactions. *Proceedings from IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Workshop on the Interpretation of Visual Motion,* 39-46.

Panangadan, A., Matarić, M., & Sukhatme, G. (2004a). Detecting anomalous human interactions using laser range-finders. *Proceedings from IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2136-2141.

Panangadan, A., Matarić, M., & Sukhatme, G. (2004b). Identifying human interactions in indoor environments. *Proceedings from Third International Joint Conference on Autonomous Agents and Multi-Agent Systems,* 1308-1309.

Patterson, D. J., Liao, L., Fox, D., & Kautz, H. (2003). Inferring high-level behavior from low-level sensors. *Proceedings of the Fifth International Conference on Ubiquitous Computing (UBICOMP),* 73-89.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185-208. Cambridge, MA: MIT Press.

Poggio, T., & Shelton, C. R. (1999). Machine learning, machine vision, and the brain. *AI Magazine, 20*(3), 37-55.

Polana, R., & Nelson, R. (1994). Low level recognition of human motion (or how to get your man without finding his body parts). *Proceedings of IEEE Computer Society Workshop on Motion of Non-rigid and Articulate Objects,* 77-82.

Sato, K., & Aggarwal, J. K. (2001a). Recognizing two-person interactions in outdoor image sequences. *Proceedings of IEEE Workshop on Multi-Object Tracking,* 87-94.

Sato, K., & Aggarwal, J. K. (2001b). Tracking persons and vehicles in outdoor image sequences using temporal spatio-velocity transform. Retrieved November 6, 2004, from http://www.ece.utexas.edu/projects/cvrc/PETS2001_final2.pdf

Tan, H. C. C., & De Silva, L. C. (2003). Human activity recognition by head movement using Elman network and Neuro-Markovian hybrids. *Proceedings of Image and Vision Computing New Zealand,* 320-326.

Tucakov, V., & Ng, R. T. (1998). Identifying unusual spatio-temporal trajectories from surveillance videos. *Proceedings of 1998 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery.*

Viola, P., Jones, M., & Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. *Proceedings of IEEE International Conference on Computer Vision (ICCV), 2,* 734-741.

watko@athena.mit.edu. (1998). World Web Math: Vector Calculus Summary. Retrieved November 12, 2004, from http://web.mit.edu/wwmath/vectorc/summary.html.

Witten, I. H., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with java implementation.* San Francisco, CA: Morgan Kaufmann Publishers.

Yan, H., & Matarić, M. J. (2002). General spatial features for analysis of multi-robot and human activities from raw position data. *Proceedings of the International Conference on Intelligent Robots and Systems, EPFL, Switzerland,* 2770-2775.

APPENDIX A: SUPPORT VECTOR MACHINES

Chapter 3 discusses machine learning theory in support of this thesis. Section 3.3 therein introduces a small subset of learning algorithms. Each is used in the course of the experiments performed in this thesis. This Appendix extends the brief discussion of Section 3.3 in regards to SVMs.

We explore the basic linear decision modeling technique through which the linear model is generated in feature space in Section A.1. We then discuss the decision boundary of an SVM – the hyperplane – in Section A.2. Section A.3 discusses margin slack variables and their use for evaluating appropriateness of a given hyperplane. Section A.4 discusses how support vectors for a given training set are selected. Section A.5 covers kernels, with specifics on how kernel selection contributes to the representational power of SVMs. Section A.6 discusses specific implementation details for our application.

A.1 Linear Decision Models: Single Layer Perceptron Algorithm

The basis of SVMs lies in linear decision models such as the Single Layer Perceptron. Its modeling method provides a substantial foundation for further discussion of SVMs.

As with all supervised learning algorithms, a perceptron is given a training set S consisting of a set of training examples of the form $(\mathbf{x}_i, y_i)$. The task is to map a given input vector $\mathbf{x}_i$ to positive or negative 1 based on the linear combination of the input values of $\mathbf{x}_i$ with a weight vector $\mathbf{w}$ and bias term b as follows (Mitchell, 1997):

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } w_1 x_1 + \mathrm{L} + w_n x_n + b > 0 \\ -1 & \text{otherwise} \end{cases} \tag{A.1}$$

or using vector notation (Cristianini & Shawe-Taylor, 2000):

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \tag{A.2}$$

where

$$\operatorname{sgn}(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise} \end{cases} \tag{A.3}$$

An instance $\mathbf{x}_i$ is correctly classified if $f(\mathbf{x}_i) = y_i$ (Cristianini & Shawe-Taylor, 2000).

A.1.1 Perceptron Weight Updates

We can view the weight vector **w** and the bias term b as defining a hyperplane decision surface in the n-dimensional space of instances[4] (Cristianini & Shawe-Taylor, 2000; Mitchell, 1997). The parameters **w** and b must be learned from a set of training instances. The training of the perceptron learning algorithm involves adjusting the weight vector **w** and the bias term b for misclassified training instances (correctly classified instances require no adjustment). Given a misclassified instance $\mathbf{x}_i$, **w** and b are updated as follows (Mitchell, 1997):

[4] Without the bias term b, the hyperplane would always pass through the origin of the n-dimensional space when the input vector is the zero vector.

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + \eta y_i \mathbf{x}_i \\ b_{k+1} &= b_k + \eta y_i \end{aligned} \tag{A.4}$$

The old weights $\mathbf{w}_k$ are combined with the misclassified input values from $\mathbf{x}_i$ scaled by a learning rate η. The correct output class y_i is used to determine whether the weights should be increased or decreased; a positive misclassification increases the weights, whereas a negative misclassification decreases the weights. The bias term *b* is updated in similar fashion, with the exception that none of the input terms contribute to *b*. Equation (A.4) is guaranteed to converge on the hyperplane which separates the two classes if the training set *S* is linearly separable (Mitchell, 1997).

A.1.2 Dual Representation

As this weight update occurs each time an instance is misclassified, the final values of **w** can be represented as a linear combination of the training instances **x** multiplied by their output **y** (Cristianini & Shawe-Taylor, 2000). Thus, the final (optimal) values of **w** would be:

$$\mathbf{w} = \mathbf{w}_0 + \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{A.5}$$

The parameter λ is determined by the number of iterations through the training set necessary for the hyperplane to converge. α_i indicates how often a given $\mathbf{x}_i$ is misclassified, and can be viewed as a model complexity metric (Müller et. al, 2001). The initial weights $\mathbf{w}_0$ are often randomly initialized (Mitchell, 1997), though without loss of generality, we can assume the initial weights are initialized to the zero vector (Cristianini & Shawe-Taylor, 2000), simplifying Equation (A.5) to

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{A.6}$$

The only constraint on each α_i is that it must be nonnegative (Müller et. al, 2001). In the case that the training data is not linearly separable, α_i increases without limit. Müller et. al introduce a user-specified regularization constant $C > 0$ which serves as an upper bound on all α_i (2001). Regarding the regularization constant, Poggio and Shelton state that "regularization solves the problem of choosing among the infinite number of functions that all pass through the finite number of data points by imposing a smoothness constraint on the final solution" (1999). Thus C provides a means for controlling the model complexity.

Substituting Equation (A.6) into the classification formula of Equation (A.2) yields

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \right) \tag{A.7}$$

Equation (A.7) is the dual representation form of Equation (A.2) (Cristianini & Shawe-Taylor, 2000; Müller et. al, 2001). Most of the further discussion of SVMS deals with the dual representation of Equation (A.7) rather than the original formula as in Equation (A.2). The optimal hyperplane can now be represented by α_i and b, and the problem of solving for them is an optimization problem known as *constrained quadratic optimization* (Witten & Frank, 2000).

Our discussion thus far has centered on linear models, beginning with the basic perceptron model and extending it to a dual representation form. The basic perceptron algorithm is extended with a gradient descent algorithm to handle cases when the training

set S is not linearly separable. The premise is that the changes in the position of the hyperplane defined by (α_i, b) can be viewed as the slope of a surface over which the hyperplane is moving. Adjustments should be made until a (local) minimum in the surface is reached. In this case, some of the $\mathbf{x}_i$ are misclassified; however, the misclassifications are optimized in such a way as to minimize the error over the entire training set S.

SVMs have a similar metric by which a hyperplane specified by α_i and b can be optimized even in cases of a training set which is not linearly separable, namely the margin of a hyperplane. Solving for the optimal hyperplane is equivalent to solving for the (α_i, b) which define the maximal margin hyperplane (see Figure 22) (Cristianini & Shawe-Taylor, 2000). We now discuss the hyperplane in more detail.

A.2 Maximal Margin Hyperplane

The linear perceptron model introduced in Section A.1 utilizes a hyperplane as a decision boundary between classes. In this section, we discuss the maximal margin hyperplane, which utilizes a margin on the hyperplane as the decision boundary between classes.

The (functional) margin of a given training instance $(\mathbf{x}_i, y_i)$ is defined by Cristianini and Shawe-Taylor (2000) as

$$\gamma_i = y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \tag{A.8}$$

or, using the dual representation described above,

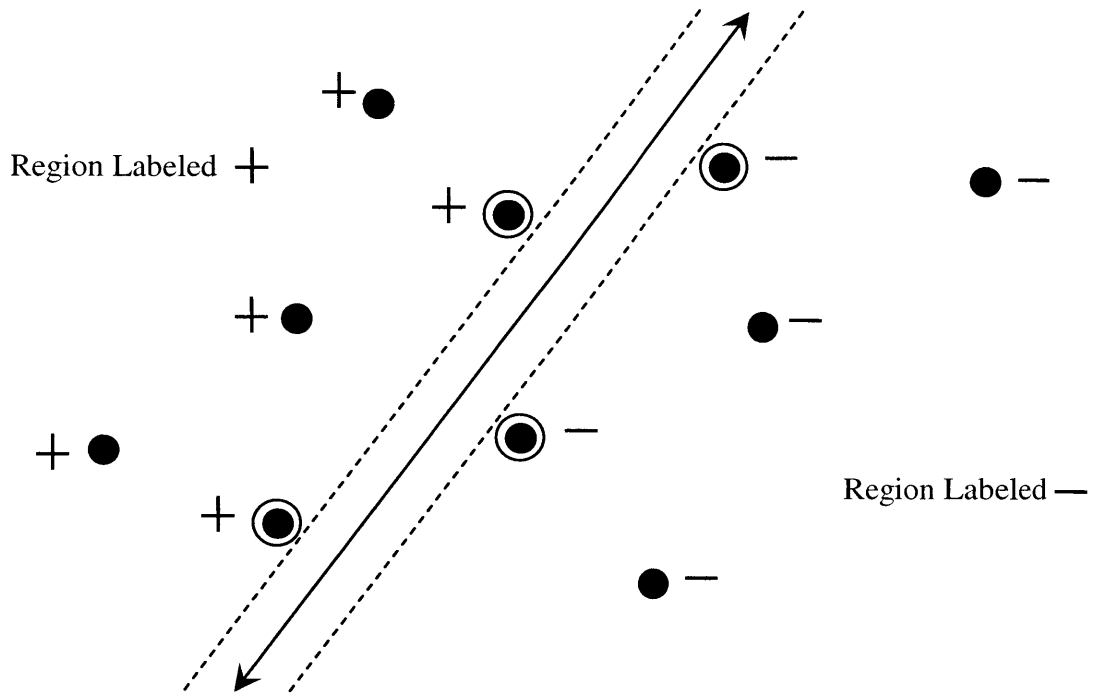

Figure 22: Marginal hyperplane with support vectors

The SVM algorithm solves for the values (α_i, b) which represent a hyperplane (a). The margin of the hyperplane, γ (dashed lines) determines the decision boundary of the hyperplane. Support vectors (circled) are the instances of the training data set which lie on the decision boundary. All other instances can be removed without affecting the position of the hyperplane.

$$\gamma_i = y_i\left(\sum_{j=1}^{1}\alpha_j y_j \mathbf{x}_j \cdot \mathbf{x}_i + b\right) \tag{A.9}$$

Note that if $\gamma_i > 0$ then the instance $\mathbf{x}_i$ was correctly classified. For a given (α_i, b), every $\mathbf{x}_i$ has a margin (Cristianini & Shawe-Taylor, 2000).

Selection of the best hypothesis to represent a given training set *S* can now be restated as the task of identifying the (α_i, b) which yield the maximal margin over all possible hyperplanes, defined as the maximal margin hyperplane (Cristianini & Shawe-Taylor, 2000). As noted previously, a linear model converges to an optimal solution where all instances of the given training set *S* are correctly classified. In terms of the maximal margin hyperplane, $\mathbf{x}_i$ is correctly classified and the corresponding γ_i are positive for all *i*. For the case where the training set *S* is not linearly separable, an added metric is necessary.

A.3 Margin Slack Variables

Cristianini and Shawe-Taylor introduce margin slack variables (ξ_i) to allow misclassified instances to be taken into account when measuring the suitability of a proposed hyperplane (2000). A pictorial example of this can be seen in Figure 23. For a given training instance $(\mathbf{x}_i, y_i)$, a hyperplane (α_i, b), and a desired margin γ,

$$\xi_i = \max\left(0, \gamma - y_i\left(\sum_{j=1}^{1}\alpha_j y_j \mathbf{x}_j \cdot \mathbf{x}_i + b\right)\right) \tag{A.10}$$

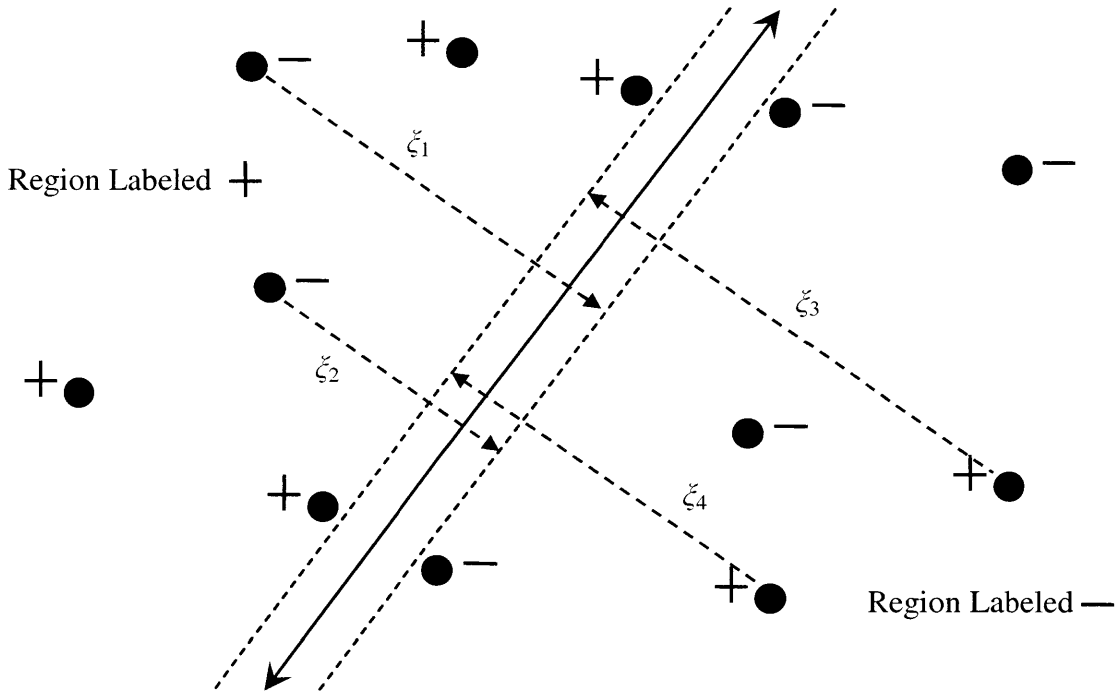

Figure 23: Marginal hyperplane with slack variables

In fitting a hyperplane to a training set, in cases where the training data is not linearly separable, some training instances will always remain incorrectly classified. Rather than discard them, these training instances are assigned slack variables (ξ_i) as an error metric for how accurately the hyperplane margin fits the training data.

Correctly classified instances have $\xi_i = 0$, whereas instances which are misclassified or lie within the margin defined by γ have nonzero slack variables. The norm $\|\xi\|_2$ measures the overall performance of a given hyperplane (α_i, b), and a desired margin γ (Cristianini & Shawe-Taylor, 2000).

A.4 Support Vectors

Recall from Section A.1.2 that all α_i are constrained to be between zero and the regularization constant C, inclusive. α_i roughly indicates the complexity of a given instance $\mathbf{x}_i$, whereas ξ_i indicates instances which cannot be fit by a given hyperplane. The overall relationship between α_i and ξ_i can be represented using the Karush-Kuhn-Tucker (KKT) conditions as follows (Müller et. al, 2001):

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(\mathbf{x}_i) \geq 1 \quad \text{and} \quad \xi_i = 0 \\ 0 < \alpha_i < C &\Rightarrow y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad \xi_i = 0 \\ \alpha_i = C &\Rightarrow y_i f(\mathbf{x}_i) \leq 1 \quad \text{and} \quad \xi_i > 0 \end{aligned} \tag{A.11}$$

For a given i, if α_i is zero, this implies that the classification of $\mathbf{x}_i$ is correct, hence the associated margin γ_i is positive, and the margin slack variable ξ_i is zero. Alternatively, if α_i is between zero and the regularization constant C, then the classification of $\mathbf{x}_i$ is exactly correct, with the associated margin γ_i equal to the maximal margin of γ, and the margin slack variable ξ_i is zero, since no adjustment is necessary. Finally, in the case of α_i equal to C, $\mathbf{x}_i$ is misclassified, the associated margin γ_i is negative, and the margin slack variable ξ_i is nonzero.

In general, the margin γ_i defined in Equation (A.9) represents the distance an instance $\mathbf{x}_i$ lies from the hyperplane. The instances which lie closest to the maximal margin hyperplane (Witten & Frank, 2000) and instances with nonzero margin slack variables (Müller et. al, 2001) are support vectors. The maximal margin hyperplane can be constructed from the support vectors for two classes, whereas the remaining training instances can be discarded as redundant data (Witten & Frank, 2000). From Equation (A.11), only the α_i associated with a training instance $\mathbf{x}_i$ which lies either on or within the margin are nonzero (Müller et. al, 2001). These $\mathbf{x}_i$ are the support vectors for a given training set *S*.

A.5 Kernels: Mapping Attribute Space to Feature Space

Our discussion thus far has dealt with linear models fit to a training set *S*. Cristianini and Shawe-Taylor note the general trend in machine learning applications of altering the representation of the given data to a version more suitable for modeling (2000). More formally,

$$\mathbf{x} = (x_1, \mathrm{K}\ , x_n)\ \mathrm{a}\quad \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \mathrm{K}\ , \phi_N(\mathbf{x})) \tag{A.12}$$

In general, the step of changing the representation of the data consists of mapping the input space *R* into a new space, $F = \{\phi(\mathbf{x}) \mid \mathbf{x} \in R\}$ (Cristianini & Shawe-Taylor, 2000), as follows:

$$\begin{aligned} \phi &: \Re^{\mathrm{N}} \rightarrow F \\ \mathbf{x}\ &\mathrm{a}\quad \phi(\mathbf{x}) \end{aligned} \tag{A.13}$$

The original input space R is referred to as attribute space, while the resulting space F is referred to as feature space (Cristianini & Shawe-Taylor, 2000). The dimensionality of feature space F is in no way constrained by the dimensionality of attribute space R (Witten & Frank, 2000).

SVMs embed the step of mapping from attribute space to feature space into the dual representation of Equation (A.7) as follows:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) + b \right) \tag{A.14}$$

The hyperplane (α_i, b) lies in feature space F rather than attribute space R. The feature map ϕ distorts the hyperplane, resulting in a much more complex decision surface in R.

A.5.1 Mapping with Kernels

Equation (A.14) maps the training set S from attribute space to feature space, and is dependent in part on our knowing the specific dimensions of the chosen feature space. Cristianini and Shawe-Taylor identify an optimization in which the inner product $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})$ is computed directly in attribute space as a function of $\mathbf{x}_i$ and $\mathbf{x}$ (2000). This direct computation method is a *kernel* function, represented as

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \tag{A.15}$$

Using this notation, the decision rule can be evaluated after λ iterations as (Cristianini & Shawe-Taylor, 2000):

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{A.16}$$

A.5.2 Kernel Selection

By analyzing properties of Equation (A.15) and the kernel *K*, Cristianini and Shawe-Taylor note that any function which maintains certain properties can act as a kernel function (2000). Specific requirements include that the function be symmetric and positive semi-definite (Cristianini & Shawe-Taylor, 2000). We are not attempting to generate nor propose new kernel functions; rather, we utilize kernel functions which have already been identified, and attempt to find the best fit to our specific application. Possible kernel functions for consideration are displayed in Figure 24 (Cristianini & Shawe-Taylor, 2000; Müller et. al, 2001).

Poggio and Shelton note that "changes to the kernel used in the SVM had little effect on the final performance" of the model (1999).

A.6 Implementation Details

The task of solving Equation (A.16) with respect to Equation (A.11) is a dual quadratic optimization problem (Müller et. al, 2001). Several approaches exist in mathematical analysis for solving the dual quadratic problem. The approach which we consider is Sequential Minimal Optimization (SMO), as it is optimized for application to SVMs. Our discussion is specific to the implementation of SMO which accompanies the WEKA toolkit of (Witten & Frank, 2000).

Platt outlines the SMO algorithm in depth, including source code and analysis of performance in comparison with another approach to generating an SVM model (1999). Other quadratic programming optimizers attempt to solve multiple quadratic equations within an inner loop, resulting in a large big-Oh bound on the runtime. SMO solves

Gaussian Radial Basis Function	$K(\mathbf{x},\mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{c}\right)$
Polynomial Function	$K(\mathbf{x},\mathbf{y}) = \left((\mathbf{x}\cdot\mathbf{y})+c\right)^d$
Sigmoidal Function	$K(\mathbf{x},\mathbf{y}) = \tanh\left(\kappa(\mathbf{x}\cdot\mathbf{y})+c\right)$
Inverse Multiquadratic	$K(\mathbf{x},\mathbf{y}) = \frac{1}{\sqrt{\|\mathbf{x}-\mathbf{y}\|^2 + c^2}}$

Figure 24: Kernel functions for SVMs

Cristianini and Shaw-Taylor discuss methods through which new kernel functions can be developed (2000). We do not develop new kernel functions, but rather utilize several of the prominent kernel functions available. The kernel functions we use are the polynomial function and the radial basis function, as these kernels accompany the WEKA implementation of Platt's SMO algorithm (1999).

small (usually two) quadratic optimizations at a time analytically, avoiding large matrix computations and improving overall performance (Platt, 1999).

This appendix covers many of the important features of SVMs in the hope that such information may prove useful for those unfamiliar with their inner workings. Section A.1 introduces the perceptron algorithm, from which SVMs are generalized. The decision surface of an SVM is the maximal margin hyperplane, as discussed in Section A.2. Section A.3 discusses the importance of margin slack variables in evaluating the fitness of a given hyperplane in fitting a training set. Section A.4 examines support vectors and their uses in SVMs. Kernel selection is discussed in Section A.5, and Section A.6 discusses the implementation which we utilize for this thesis, namely Platt's SMO algorithm. With this understanding of SVMs, their usage in this thesis may be more easily comprehended and integrated.