

University of Nevada, Reno

# **HMM-Based Techniques for Intent Recognition in a Simulated Realistic Naval Environment**

A thesis submitted in partial fulfillment of the  
requirements for the Degree of Master of  
Science in Computer Science and Engineering

by

Alexander James Gamino

Dr. Mircea Nicolescu, Thesis Advisor

May 2016



THE GRADUATE SCHOOL

We recommend that the thesis  
prepared under our  
supervision by

**Alexander Gamino**

entitled

**HMM-Based Techniques for Intent  
Recognition in a Simulated Realistic Naval  
Environment**

be accepted in partial fulfillment of  
the requirements for the degree of

**MASTER OF SCIENCE**

Dr. Mircea Nicolescu, Advisor

Dr. Monica Nicolescu, Co-advisor

Dr. Aleksey Telyakovskiy, Graduate School  
Representative

Dr. David Zeh, Graduate School Dean

May 2016

## **Abstract**

Activity recognition aims to recognize the actions of one or more agents from a series of observations. Intent recognition is an area of research dedicated to automatically detecting and predicting the intentions of agents in an environment before they are realized. Intent recognition occurs in conjunction with activity recognition, and both may use the same set of low-level features. When applying intent recognition to simulated environments, realism is a key factor for producing systems which are adaptable to real-life settings. This thesis explores techniques for creating intent recognition systems which are readily adaptable to real world scenarios.

We seek to develop methods of intent recognition for multiple ships in a realistic naval environment, where the task is complicated by realistic physics and ship movements. Our intent recognition system is primarily based on Hidden Markov Models that have been trained on large datasets of simulated scenarios, where ships are enacting the patterns which we wish to later recognize. These patterns involve ships performing various maneuvers, such as blocking or ramming. Based on the detection of a set of highly discriminatory features extracted from the data, the HMM is able to recognize patterns in order to reliably separate ship actions into various intent classes. Using a temporal window of activities, our system predicts ship intentions before they are realized, potentially allowing for responsive actions.

## **Dedication**

For everyone who helped  
me in my research

## **Acknowledgements**

I would like to acknowledge the contribution from my advisor Dr. Mircea Nicolescu, and give special thanks to both Dr. Monica Nicolescu and Dr. Aleksey Telyakovskiy for being on my committee. I would also like to give thanks to everyone who researched previous iterations of this project, with special regard to Mohammad Saffar, whose guidance was pivotal in the formative stages of my research.

# Contents

Abstract .....	i
Dedication .....	ii
Acknowledgments .....	iii
List of Figures.....	vi
1 Introduction.....	1
2 Background and Related Word.....	3
2.1 Activity and Intent Recognition.....	3
2.1.1 Overview.....	3
2.1.2 Activity Recognition .....	4
2.1.3 Intent Recognition.....	6
2.2 Plan Recognition.....	8
3 Hidden Markov Models .....	13
3.1 Overview .....	13
3.2 Parameter Estimation.....	16
3.3 Viterbi Decoding .....	18
3.4 Evaluation .....	19
4 Design.....	21
4.1 System Overview .....	21
4.2 Simulated Environments.....	23
4.3 Preprocessing.....	24
4.4 Intents .....	26

4.4.1 Ramming .....	27
4.4.2 Blocking.....	27
4.4.3 Herding.....	27
4.4.4 Benign.....	28
4.4 Feature Extraction .....	28
4.5 Training.....	31
4.5 Testing.....	32
5 Results .....	33
5.1 Overview .....	33
5.2 Experiment One: Ramming Behavior .....	34
5.3 Experiment Two: Blocking Behavior.....	39
5.4 Experiment Three: Benign Behavior.....	44
5.5 Experiment Four: Herding Behavior.....	49
5.6 Review of Results.....	54
6 Conclusion.....	55
6.1 Closing Remarks .....	55
6.2 Improvements and Future Work.....	56
6.2.1 Fix Herding Data .....	56
6.2.2 Multi-agent Domains .....	56
6.2.3 Parallelization.....	57
6.2.4 High Level Intents.....	57
6.2.5 Additional Features.....	57
Bibliography .....	59

# List of Figures

2.1	The activity, and intent recognition process .....	4
3.1	A Simple HMM .....	14
3.2	Video frames demonstrating American Sign Language .....	15
4.2	Graphical representation of the naval simulation program .....	24
5.2.1	Prediction results for Ram Test #1.....	35
5.2.2	Confusion Matrix for Ram Test #1.....	35
5.2.3	Scatterplot for Ram Test #1.....	36
5.2.4	Prediction results for Ram Test #2 .....	37
5.2.5	Confusion Matrix for Ram Test #2 .....	37
5.2.6	Scatterplot for Ram Test #2 .....	38
5.3.1	Prediction results for Block Test #1 .....	40
5.3.2	Confusion Matrix for Block Test #1 .....	40
5.3.3	Scatterplot for Block Test #1.....	41
5.3.4	Prediction results for Block Test #2.....	42
5.3.5	Confusion Matrix for Block Test #2 .....	42
5.3.6	Scatterplot for Block Test #2 .....	43
5.4.1	Prediction results for Benign Test #1.....	45
5.4.2	Confusion Matrix for Benign Test #1 .....	45
5.4.3	Scatterplot for Benign Test #1 .....	46
5.4.4	Prediction results for Benign Test #2 .....	47
5.4.5	Confusion Matrix for Benign Test #2 .....	47
5.4.6	Scatterplot for Benign Test #2.....	48
5.5.1	Prediction results for Herd Test #1.....	50
5.5.2	Confusion Matrix for Herd Test #1 .....	50
5.5.3	Scatterplot for Herd Test #1 .....	51
5.5.4	Prediction results for Herd Test #2.....	52
5.5.5	Confusion Matrix for Herd Test #2.....	52
5.5.6	Scatterplot for Herd Test #2.....	53

# Chapter 1

## Introduction

The ability for a system, or artificial intelligence agent to observe the actions of other agents (human or otherwise) in an environment, recognize these actions, and use this information to create inferences about future events is known as intent recognition. Intent recognition plays a pivotal role in many aspects of computer science, specifically artificial intelligence and robotics, because it allows an agent to gain a sense of *understanding* about its environment: the way it works and the reasoning behind the actions of its inhabitants. This fundamental cognitive ability plays a crucial role in understanding the behaviors and motivations of other entities within the environment. The ability for an intelligent agent to examine the actions of other agents in an environment and understand their goals allows for the development of sophisticated learning techniques. Using these techniques, an agent can learn to understand and replicate observed behaviors and factor this knowledge into its own plan making.

Intent recognition plays a critical role in many distinct fields, from speech-to-text, human-computer interaction and assembly line automation to weather prediction, autonomous drones and bionic limbs. Intent recognition is an active area of research, enabling unique solutions to a variety of problem domains.

The research in this paper centers around the concept of using intent recognition to train simulated naval vessels to recognize the actions, plans and

goals of other vessels within the simulation. Using a realistic simulator, it is possible to generate large data sets which contain information describing the ship intentions we will later attempt to recognize. The simulator allows us to easily generate different configurations of ship intentions, for both training and testing.

It is then possible, using these data sets, to extract certain highly descriptive *features* (information separating one intention from others) that can be used to determine which intent the ship is likely executing. These features can describe aspects such as speed, acceleration, rotational velocity and relative heading. Once features have been computed, the system then examines the current state of a ship's features. Taking into account its features at previous time steps, the system makes a prediction about what a particular ship is doing, and what it is likely to do next.

Chapter 2 of the thesis will describe previous work in this research domain, which has provided essential insight into techniques used in our approach. The methods discussed in Chapter 2 will also help to provide a brief history of how the field of intent recognition has evolved over time, and put in context the contemporary approaches used here. Chapter 3 will examine in detail some of the technical concepts necessary to fully understand the later sections of this paper. It includes a detailed overview of Hidden Markov Models, starting with the general concept, then moving on to specific mathematical constituents. Chapter 4 is a discussion of the design and implementation of the intent recognition system. In addition, it also includes a discussion of simulated environments, and how they relate to data collection. Chapter 5 discusses the experimental results achieved by the system on a case by case basis, covering the areas in which this technique does well, as well as its limitations. Chapter 6 provides the conclusion, briefly summarizing the research performed in this paper, and providing suggested avenues for additional research.

# Chapter 2

## Background and Related Work

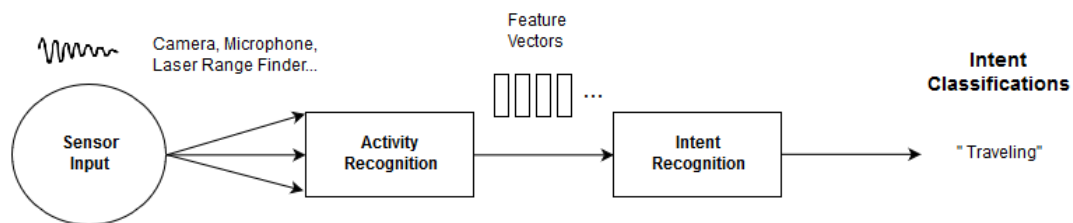
### 2.1 Activity and Intent Recognition

#### 2.1.1 Overview

In the fields of intent and activity recognition, agents can exist in real world scenarios (humans or robots), or exist within a simulated environment (software agents). Activity recognition and intent recognition are both aspects of machine learning which aim to make inferences about agents based on observations of their behavior, their interactions with other agents, as well as their interactions with the environment. These studies are important for many areas of research, including computer vision, pattern recognition and classification, artificial intelligence and human-computer interaction. The ability to recognize activity and intent allows us to reason about what other agents are doing, why they are doing it, what they may do next, as well as allow us to plan a response. These are pivotal, cognitive actions which are essential for higher-level reasoning.

Research into this field is divided into several subareas, including activity and intent recognition. Activity recognition specifically refers to the task of extracting meaningful information, features and patterns from low-level data, such as sensor input, in order to detect activities. The input is usually unfiltered, noisy data such as that from a camera feed, microphone, sonar, radar, GPS or laser range finder. Meaningful information in this case can refer to anything which will enable a system to perform classification. For instance, when performing activity recognition within a video stream, the task may rely on movements of regions of connected pixels. Another example could be determining syllables within a voice recording. Intent recognition is a higher-level processing step which utilizes the low-level information produced by activity recognition, and attempts to recognize

relationships, goals, and intents of agents within the observable environment. Depending on the application, the definition of an intention can be very different. For instance, if the application is path finding, an intention could be trying to determine where an agent intends to go, or how it intends to get there. In speech recognition, intent recognition could mean determining what word is being said, or if the sentence poses a question. Figure 2.1 shows a graphical representation activity and intent recognition.



**Figure 2.1:** The activity, and intent recognition process

There are many varied approaches to the fields of intent and activity recognition. Some utilize biologically inspired artificial neural networks, or genetic algorithms, or a combination of both (neuroevolution) [38]. Other may use simpler statistical methods such as Bayesian networks [22]. However much research being conducted in these areas has focused on representing and learning based on a temporal representation of characteristic activity sequences. This type of approach has led to the widespread adoption of models such as the Hidden Markov Model, which is a simple and efficient method of classification for temporal, or sequential, activity sequences. This paper focuses on such an approach, utilizing temporal data and Hidden Markov Models.

### 2.1.2 Activity Recognition

There has been much research in the area of activity recognition; in general, activity recognition has been used as a starting point in many robotics and artificial intelligence methods, as in [1, 2]. The research outlined in [2] demonstrates activity recognition by teaching a robotic arm, with a high degree of freedom, how

to interact with and manipulate primitives by utilizing examples from a tele-operator. The  $k$ -nearest quantized pattern vectors are used for the training of potential classes; specifically, they are used to determine low-level actions and their corresponding controls. These classified actions are then used as input to a Hidden Markov Model, for added context and smoothing, which helps to reduce the rate of misclassifications. In [1] a robot is taught how to recognize and perform various assembly tasks by observing human demonstration of the act. In this work, the sensory data gathered during the observation phase is used to create the observation symbols for training the parameters of a Hidden Markov Model. The states of the HMM correspond to the states in the assembly task; in this way the learned HMM is employed to model the data generated from the human demonstration of the task. This HMM is then used to control the robot performing the task, and the observation symbols now correspond to the reference commands for the robot controller, which determine which actions need to be performed in order to complete the assembly task.

Research in the field of activity recognition is not always focused on solving one particular task. Some research formulates activity recognition as a general theory, with many potential applications. The research conducted in [3, 4] are examples of this. In [3] the general concept of coupled Hidden Markov Model is introduced. A coupled HMM is essentially a joint HMM (built from two or more individual Hidden Markov Models) which is created by taking the Cartesian product of their states and transition parameters. One of the prominent drawbacks in using Hidden Markov Models for temporal pattern inference is the requirement of satisfying the Markov assumption, a topic discussed in the Chapter 3 of this paper. This is the assumption that the probability of the state at time step  $t+1$  depends solely on the preceding time step  $t$ . The coupled Hidden Markov Model does much to relax this constraint and allows for the encoding of more complicated, interacting patterns than the traditional Hidden Markov Model. In [4] a 3D SIFT-based descriptor for video streams is introduced. Unlike much of the previous SIFT-based research in video stream action recognition, which typically analyzed video streams in a frame-by-frame fashion, the approach in this paper operates on the entire video segment. This approach treats the video data as a three dimensional space, looking for patterns not only spatially (between regions of

pixels), but also temporally between adjacent video frames. Classification is performed by training Support Vector Machines (SVM), using the features produced from SIFT for each action class. Discrimination between classes is then performed by determining the model whose data is farthest from the SVM hyper-plane. This approach proved to be much better at video classification than previous SIFT-based methods, and had the advantage of capturing temporal patterns, which frame-by-frame analysis tended to miss.

### **2.1.3 Intent Recognition**

While the previously mentioned research examined techniques to recognize activity, though many can be adapted to recognize intent, there is also a large amount of work which concentrates specifically on the problem of intent recognition and classification. Some of the earliest formal research on the subject can be found in [5]. This paper introduces the concept of a hierarchical library of event classifications, which are composed of various individual intentions. It then defines the concept of *minimum covering entailment*, which selects plan models in which the library contains all of the recognized events, while minimizing the number of unrelated observations. Unrelated observations refer to observations which are not included in the plan model. The approach highlighted in this paper is very general, and therefore applicable to many different types of intentions. However, the hierarchical structure of the library imposes a significant penalty on the performance of the classification algorithm. In addition, the approach outlined in this paper is specific to a single agent scenario.

A probability-based approach can be found in [6], which used Bayesian networks to represent intentions. In this paper the highest level nodes in the Bayesian network correspond to the overall intentions, while the lower-level sub-nodes represent contributing sub-actions. Given this representation, it is sufficient to simply traverse the Bayesian network to determine if a high-level intent is likely, and therefore gain a sense of confidence that the corresponding action is indeed occurring. Although this approach is quite effective on the problems discussed in

the paper, there are significant problems when attempting to represent more complex actions as Bayesian networks. Scalability also proves to be a problem when given multiple interacting agents, or large numbers of separate intentions.

The techniques in [7] illustrate a more biologically inspired approach to the task of intent recognition. This paper introduces the concept of *forward models*, which are functions which take as input the current state of the system and a command to be applied on it, and outputs a prediction for the next state of the controlled system. Similarly, this paper also introduces the concept of an *inverse model*, which is a function which takes as input the current state of the system and the target state or goal, and produces the sequence of events which are needed to reach or maintain this goal. A *forward model* paired with an *inverse model* form the background for the techniques used in this paper. The forward model produces an estimate about the future states, which are in turn given to the inverse model, allowing it to adjust any parameters if needed. This concept is based on the biological mechanism of mirror neurons [8]. The approach works well when the number of possible intentions is small, however it scales poorly to large numbers of intentions.

Demiris' work in [9] highlights many of the difficulties in the field of intent recognition. This publication formalizes two concepts: *perspective taking* and *hierarchical representations*. Perspective taking refers to the concept of examining behaviors of an outside agent, and determining what is the agent's goal. This concept is critical for intent and activity recognition, as well as for many other supervised learning techniques. For the concept of hierarchical representations of intent, basic low-level actions form a constituent basis for higher level intent, in a bottom-up approach. A clear example of such an approach can be found in [10] which uses a three-layer hierarchy composed of simple actions at the lowest level, sequences of actions in the middle layer, and symbolic representations of goals at the top-most layer.

A more general example of intent recognition is seen in [11]. In this paper, a probabilistic approach is suggested for intent recognition in the context of elder care. The paper covers the many aspects involved when creating an intent recognition system, outlining concerns such as plan abandonment, partially fulfilled plans, and reacting to hostile agents in the environment. This work

outlines how such a probabilistic model could achieve the desired results, however it is entirely theoretical and the proposed solution was not implemented.

Similar to the case of activity recognition, Hidden Markov Models are a popular and effective technique for performing intent recognition, especially in the case of temporally-structured intent data. In [12] Hidden Markov Models are used for the representation and recognition of intents for individual robots in a multi-robot soccer domain. The method outlined in this paper begins by calculating low-level state features from the robot's sensor input, and uses vectors of such features to train a set of Hidden Markov Models which correspond to various actions which the robots can take. Once trained on known data, these Hidden Markov Models can then be used to infer how likely a given set of input features was produced by each model. This can be extrapolated to the general case where a Hidden Markov Model is trained on some arbitrary intent, which can be recognized by some set of observable, discrete features. However, the complexity of the environment in this paper is quite low and controlled, with a very small number of states and possible actions and intents.

The research in [13, 14] utilize the Hidden Markov Model based technique outlined in [12], but perform intent recognition in a much less structured environment. This paper demonstrates that it is possible to recognize the intents of a human agent in an environment, either involving interactions with objects in the scene or with another human, based on features synthesized from data gathered by simple sensors (a laser range finder and a visible light camera). While the research in [13, 14] shows the applicability of a Hidden Markov Model based approach to many different domains, it still suffers from the same set of problems as [12], namely scalability to large numbers of agents due to its high computational complexity.

## **2.2 Plan Recognition**

The examples we have looked at insofar have been exclusively intent and action recognition methods. Now we are going to examine a higher level concept, which

builds upon the previously discussed techniques: plan recognition. The ability to determine another agent's goals and plans is essential in allowing humans to determine what that other agent is doing, why they are doing it and what they may do next. This fundamental cognitive ability is crucial to many aspects of interpersonal interactions and relationships, as many of these presuppose an inferred understanding of the motivations of the participants. As the complexity of human-computer interactions increases and techniques become more sophisticated, research in this field attempts to provide computers with comparable plan recognition capabilities.

Plan recognition techniques play an essential role in a wide variety of real world applications. This includes smart homes, video surveillance, human computer interaction as well as intelligent user interfaces. Research in the field of plan recognition dates back at least thirty-five years, when it was formally defined in a paper by Schmidt, Sridharan, and Goodson [15]. Since its inception, there have been significant developments in the field of plan recognition; these advances have been driven primarily due to three main causes. First, the pressing need for fast and sophisticated plan recognition techniques in a wide variety of potential applications. Second, breakthroughs in the related fields of machine learning, probabilistic modeling and optimization, in addition to the creation of more powerful computers on which to perform these methods. Third, the advancement of information gathering tools, including simulations.

The work described in [15, 16, 17] represents some of the earliest work in plan recognition research. In following with the dominant artificial intelligence strategies of the time, the approaches in these papers are all predominantly rule-based. These papers attempted to establish rules which would mimic the nature of plan recognition. However, over time researchers discovered that without a comprehensive, underlying theory to give these rules structure, they quickly became difficult to maintain and failed to scale well.

In [18] Kautz and Allen put forth a proposed "General Plan Recognition" framework, a conceptual framework which would later be used by many researchers in the field. In [18] the authors formally defined the problem of plan recognition as finding the minimal set of *top-level actions* capable of explaining the set of observed actions. Plans were represented as graphs in which root nodes

were the *top-level actions*, and the expansion of these actions gave unordered sets of child actions along which a plan could be decomposed into its most basic observable actions. Therefore, the researchers in [18] proposed that the problem of plan recognition was essentially the problem of graph covering. In a follow-up paper [19] Kautz presented an approximate implementation of this approach to the problem laid out in his previous paper, by recasting the problem into computing the vertex covers of the plan graph. One shortcoming of these early implementations of plan recognition was their inability to take into account the *a priori* likelihood of different plans and goals. For example, if the observed action were getting onto a train, then the algorithm would view “travel” and “terrorist attack” as having equal likelihood, as they both explain, or cover, the observed actions equally well.

As a reaction to the previously mentioned problems with the graph covering approach to plan recognition, Charniak [20] instead postulated that the general problem of plan recognition was best formulated as a problem of *abduction*. The word abduction in this context is a concept credited to philosopher C.S Peirce [39]; its meaning is essentially: “If we know that event A caused event B and we have observed B, then we can assume with a degree of confidence that A occurred”. In the context of plan recognition, this can be extrapolated into: “If an agent has observed in training that a goal G has been achieved by a series of actions A, and A has been observed, then we can postulate that the agent is trying to accomplish G.” Being able to classify the problem of plan recognition as a form of abductive reasoning is important as it allows for simple, standardized computational formulations. In addition, it has provided a link to related problem domains such as probabilistic inference and diagnosis.

In [21] we see one of the earliest purely abduction-based approaches to plan recognition; in this paper Hobbs *et al* created a method of abductive inference plan recognition based on the idea of cost-limited theorem-proving. This paper aims to find *proofs* for various observations, called *narratives* (a theorem to be proved). The approach in this paper would examine various observations and attempt to fit them into potential narratives based on *a priori* knowledge, using a cost-limiting approach. This can be seen as analogous to a doctor giving a prognosis (*proving*) on a medical condition (*narrative*) based on known symptoms of that particular

condition (*observations*). Later research in the field showed that this approach is a form of probabilistic reasoning.

Building upon the research founded in [20, 21] Charniak and Goldman [22] argued that if plan recognition were indeed a problem of abduction, then the best approach would be based on probabilistic (Bayesian) inference. Similarly to the cost-limiting approach in [21], Bayesian inference favors the idea of minimal explanations in the case of equally likely possibilities. This approach also handles cases in which the likelihood of two explanations is different, but the complexities of the explanations are the same. To illustrate these points, consider that the system has observed three situations which explain a narrative equally well for a given set of observations when a person is going into a barber shop. They are: *going to get a haircut and a shave* being one, *going in to get just a haircut* being the second, and *going in to get just a shave* being the last. Basic probability theory (with some assumptions) will favor the simpler explanations. Again consider the example of a person getting onto a train, where the two hypothesis were “travel” and “terrorist attack” and each of these situations explained the observed symbols equally well; in this case the *a priori* probabilities would differentiate, and travel would be seen as the most likely explanation. Building upon the work in [21], Charniak and Goldman proved that Hobbs and Stickel’s cost-based abduction technique could be formulated as a probabilistic search for the most likely *a posteriori* explanation for a set of observations. At the time of its inception, the Bayesian approach to plan recognition was met with some controversy and skepticism, however since then probabilistic inference has become the *de facto* method of performing plan recognition in one form or another.

Another broad area of research in plan recognition revolves around the idea of reformulating the basic concept into a parsing problem. The work in [23] puts forth the idea that reasoning a plan from a set of observations is hierarchical in nature. The approach in this paper was to create and use a plan hierarchy to determine plans from sets of observations, similar to techniques used to parse sentences into parse trees from known grammars. Early forays into parsing-based techniques for plan recognition promised better efficiency than the more traditional methods being employed at the time, however this was at the cost of having to make strong assumptions about the order of plan steps. One of the significant drawbacks to

early work in this area was the inability to recognize partial plans or multiple interleaved plans. Later approaches, which combine this technique with some of the Bayesian and probabilistic inference-based approaches mentioned earlier, have been making significant headway into solving the aforementioned shortcoming [24, 25, 26].

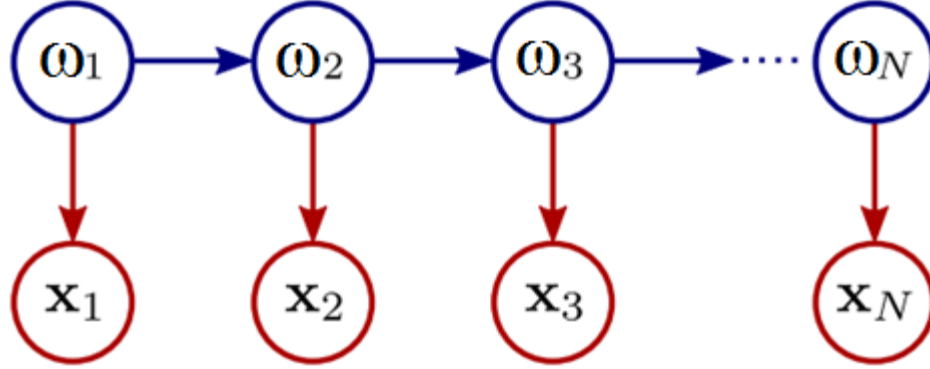
Lastly there has been a great amount of work in Hidden Markov Model (HMM) based approaches to plan recognition. These approaches borrow ideas which achieved prominence in the field of signal processing, notably including voice recognition applications, a field where HMM based approaches proved very useful and popular. HMM approaches to plan recognition offer the improved efficiency of the parsing based approaches, when compared against abduction techniques, in addition to other advantages such as incorporating likelihood as a quintessential aspect of plan recognition and supporting machine learning for automatically creating the plan models needed for determination. An HMM-based approach is the technique used for the research in this paper, and HMMs will be covered in greater detail in the following chapter.

# Chapter 3

## Hidden Markov Models

### 3.1 Overview

Hidden Markov Models are a popular and successful statistical model for recognizing temporal patterns. In this section we provide a formal definition for the Hidden Markov Model, as well as methods for training and classification. Before continuing it is important to define the underlying foundation of the Hidden Markov Model: the Markov process. In many patterns it is possible to predict a future state given the structure of the current state and a finite history of previous states, independent of the previous states beyond that. In other words, this is a *memoryless* prediction; its predictive capacity is based solely on the current state, and a finite number of previous states, not taking into account anything that happened outside that window. A process is called a  $j^{\text{th}}$ -order-Markov process if its prediction is based on only the  $j$  most recent events. A Hidden Markov Model is based upon the problem being an assumed Markov process. The structure of an HMM is represented by an underlying Markov chain, with a set of unobserved (hidden) states. The definition of a Markov chain is simply a stochastic state space, where all transitions have the aforementioned *memoryless* attribute. The directed graph in Figure 3.1 gives a good visualization of the basic structure of an HMM, demonstrating that the observations are serially independent.



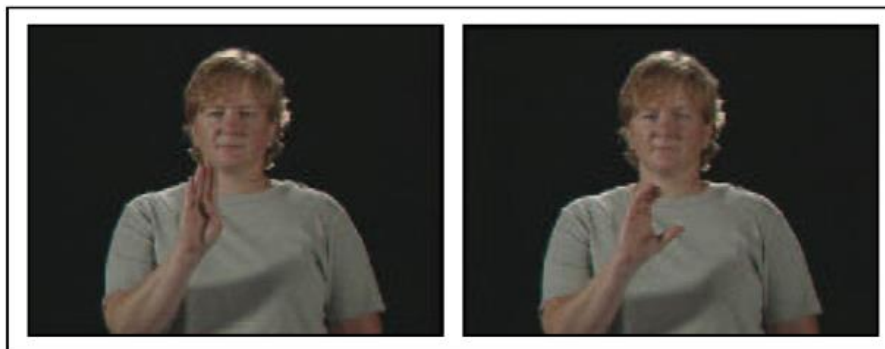
**Figure 3.1:** A simple HMM with observed states  $X_1...X_N$  and hidden states  $\omega_1... \omega_N$

Utilizing an HMM to model the dependent structure of time series patterns is a good way to simplify complex temporal patterns, due to the aforementioned Markov property being mathematically convenient. The unique aspect of this approach is, as the name suggests, the ‘hidden’ part. Unlike a traditional Markov chain, in a Hidden Markov Model we cannot observe the states directly, instead we can see only the effects, or *observations*, of a system. Using these observations, it is possible to calculate the probability that the HMM is in one state or another, based on the current observations. To understand this we will introduce some notations. As seen in Figure 3.1,  $\{X_t\}=\{X_t, t=1,2,3,..N\}$  refers to the set of observations, while  $\{\omega_t\}=\{\omega_t, t=1,2,3,..N\}$  denotes the Markov chain with  $N$  states. By the definition of a Markov chain, we know that the  $\{\omega_t\}$  follow the Markov property, that is  $P(\omega_{t+1} | \omega_t = \omega_t, \omega_{t-1} = \omega_{t-1}, \dots, \omega_0 = \omega_0) = P(\omega_{t+1} | \omega_t = \omega_t)$ . It follows that  $P(X_{t+1} = x_{t+1} | \omega_t = \omega_t)$ , given the independent property of the underlying Markov chain. We call a pair of processes  $\{\omega_t, x_t\}$  a  $J$ -state Hidden Markov Model. This relationship shows that a Hidden Markov Model is the combination of two processes; a Markov chain  $\{\omega_t\}$  as well as some random state-dependent noise  $\{X_t\}$ , which exists as an extension of this Markov chain [36, 37].

As shown in the graphical representation of an HMM in Figure 3.1, we have a series of observed measurements  $\{X_t\}=\{X_t, t=1,2,3,..N\}$  each providing information about the world in the corresponding discrete world state  $\{\omega_t\}=\{\omega_t, t=1,2,3,..N\}$ . The world states in an HMM are connected together in such

a way that the preceding states influence the current state, and may potentially help to resolve situations in which measurements are ambiguous.

One of the traditional applications of HMMs is the recognition of sign language hand gestures [27] (Figure 3.2).



**Figure 3.2:** Video frames demonstrating American Sign Language gestures

At each frame of video there may be information about what gesture is currently being presented, however this information may be ambiguous. However, we can impose certain *a priori* knowledge about what gestures are likely to precede other gestures using an HMM, and thus improve the accuracy of recognition.

One final note is the existence of three basic forms of HMMs. These various forms are differentiated by their method for modeling output probabilities. The *discrete* HMM will output discrete symbols for classification, which typically refer to quantization level (classifications). This model is good for the classification of finite, discrete elements, for example the sides of a coin. The *continuous* HMM will output a combination of *probability density functions*, or *pdfs*, which estimate the model as a continuous function. This model is good for observations which are continuous in nature, for instance weight and height. The final form of an HMM is a *semicontinuous* HMM, a hybrid approach combining elements of both the discrete and continuous Hidden Markov Models. Similar to the discrete HMM, this model is trained on discrete input from a finite set of classifications; however, similar the continuous form the actual model is a continuous Gaussian *pdf*. This approach essentially attempts to model the variance of an observation class. The discrete HMM is the form used in the proposed research here. In order to utilize a

Hidden Markov Model for the purposes of intent recognition, according to [34, 35, 36, 37] one must be able to:

- Estimate the HMM parameters to maximize the objective function
- Determine (decode) the maximal state sequence given the observations
- Evaluate the likelihood of a model given the observation sequence

These three topics will be discussed in the following sections.

## 3.2 Parameter Estimation

Parameter estimation is sometimes known as the learning or training step. During this step, the model for an HMM is created based on some observations of the process we wish to model. Given a set of examples from a process, we would like to estimate the HMM model parameters which best describe that process. A Hidden Markov Models parameter are denoted as  $\lambda = (A, B, \pi)$ . In this context  $A$  is a transition matrix, which stores the probability of state  $i$  following state  $j$ . It should be noted that the transition probabilities are independent of time  $t$ , and that an arbitrary state is denoted as  $\omega$ :

$$A = [a_{ij}], a_{ij} = P(\omega_t = \omega_i \mid \omega_{t-1} = \omega_j)$$

$B$  represents an array of observations, which store the probability of observation  $k$  being emitted from the state  $i$ , where an arbitrary observation is denoted as  $x$ , independent of  $t$ .

$$B = [b_i(k)], b_i(k) = P(X_t = x_k \mid \omega_t = \omega_i)$$

$\pi$  refers to an initial state probability array, representing the overall probability of being in some arbitrary state in the HMM:

$$\pi = [\pi_i], \pi_i = P(\omega_t = \omega_i)$$

In addition to these parameters, an HMM has two more that are inherent to the model. The first is the number of states in the HMM, typically denoted by  $N$ ; although these states are hidden there is often some physical significance assigned to them, for examples *heads* or *tails* in a coin flipping model. The second parameter is the discrete observation symbol alphabet, typically denoted by  $S$ , where  $S = (S_1, S_2, \dots S_M)$ . Unlike  $A$ ,  $B$  and  $\pi$ , these parameters are not estimated as part of the training step. They are defined instead as initialization constants, which must be present before training or testing (covered in sections 4.6 and 4.7) of an HMM can begin.

Assuming that the Markov assumption and the independence assumption hold, it is possible to model our problem by estimating the parameters  $A$ ,  $B$ , and  $\pi$  (collectively  $\lambda$ ). There are two standard approaches, which depend on the form of training examples available. These two approaches are referred to as supervised and unsupervised learning. If the training data contains both the inputs and outputs of a model, we can perform supervised learning. In this approach we use the inputs to the model as the observations, while the outputs are used as labels of the classifications. However, if we do not already have the corresponding classifications, we must instead perform unsupervised learning. In unsupervised learning, a model is estimated mathematically based on recognized patterns in the input data. Since the system has no way of measuring the correctness of this model, the model produced must be evaluated on data of known classifications once training has been completed. In this section, and in this thesis as a whole, we will be focusing on unsupervised learning techniques for Hidden Markov Models, specifically on the popular Baum-Welch algorithm.

The Baum-Welch algorithm was first proposed in statistical papers by Leonard Baum, in the 1960s [28]. Proven to always converge, the Baum-Welch algorithm will find a set of locally optimum HMM parameters for a given set of observations. Put another way, the Baum-Welch algorithm will find a local maximum for the probability of observing some observation sequence,  $O$ , given an HMM,  $\lambda$ :

$$P(O | \lambda)$$

Because of this, it is important to provide data which encapsulates the given problem in all aspects, doing otherwise may lead to a model which is biased toward certain partial cases. In other words, the HMM will only learn to recognize patterns which it has been shown previously, so it is important to include all potential cases in training. In addition, while the algorithm is only guaranteed to find a locally optimal solution, starting the algorithm with a strong initial guess will help the procedure to find a globally optimal solution [37].

The Baum-Welch algorithm is a special form of the Expectation Maximization algorithm. In essence, the Baum-Welch algorithm is an attempt to calculate the value of two auxiliary variables  $(\alpha, \beta)$  through calls to expectation and maximization functions. This will produce an estimate of the HMM parameters  $(\lambda')$ . Given this estimate, the process is repeated until convergence has been achieved.

### 3.3 Viterbi Decoding

For any problem which contains hidden variables (such as HMMs) the process of determining which variables are the source of some output is known as decoding. In the case of HMMs the decoding step aims to calculate the sequence of hidden states which is most likely to have produced a given set of observations. The most common approach to solving this problem is known as Viterbi Decoding. The Viterbi algorithm is an efficient, general purpose, dynamic programming algorithm for finding the most probable state path through a model, for a given observation [35, 37].

The Viterbi algorithm, defines a parameter  $\alpha_t$ , which is the hidden state sequence  $(\omega_{1:t})$  most likely to have produced the observation sequence  $(X_{1:t})$  :

$$\alpha_t = \max \mathbf{P}(X_{1:t}, \omega_{1:t})$$

This is known as the Viterbi parameter. This probability naturally depends on the probabilities of the of the preceding states: the series of states ending in state  $\omega_{t-1}$  ( $\omega_{1:t-1}$ ) which have outputted the observations  $X_{1:t-1}$ . This relies on the preceding states and observations at  $t-2$ , and so on. By multiplying each of these probabilities with the probability that the model transitioned from  $\alpha_{t-1}$  to  $\alpha_t$  and emitted observation  $X_t$ , we get several possible calculations for potential hidden state sequence, each of which depend on even earlier states.

Using a recursive dynamic programming technique we can find the maximal value of  $\alpha_t$ . The initial values of  $\alpha_1$  are equal to the joint probability of starting in state  $\omega_1$  and emitting the observation  $X_1$ . Then we can recursively fill out an  $N \times T$  table of possible state transition and emission paths, calculating the entries column wise, as the value of  $t$  increases. Once the table has been calculated, the most likely state sequence  $\alpha^*$  can be found by back-tracing from the maximum value for  $\alpha_t$ . [35].

### 3.4 Evaluation

Given an HMM model and a series of observations, we would now like to compute the probability that the observed sequence ( $O$ ) was emitted from the HMM ( $\lambda$ ). We are computing  $P(O | \lambda)$  for the given observation sequence - this is called the evaluation step, and is also known as *testing*. This problem could be expressed at a higher level as trying to determine how well this model explains the observations, so that the best model could be chosen from a set of models. The probability of emitting the observations  $O$ , from an arbitrary sequence of HMM states  $Q$ , according to our given model  $\lambda$ , is as follows:

$$P(O|Q, \lambda) = \prod P(o_t|q_t, \lambda)$$

Where  $o$  and  $q$  represent individual observations and HMM states belonging to  $O$  and  $Q$ , respectively. The probability for the state sequence  $Q$  given the HMM model  $\lambda$  is:

$$\begin{aligned} P(O|\lambda) &= \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{t-1} q_t} P(O|\lambda) \\ &= \sum_Q^P (O|Q, \lambda) P(Q|\lambda) \end{aligned}$$

Using these two equations, we can simplify  $P(O | \lambda)$  as:

$$P(O|\lambda) = \sum_Q^P (O|Q, \lambda) P(Q|\lambda)$$

Thus, we now have an evaluation function for determining how well a given model can explain a given observation [36]. This forms the backbone of the HMM techniques used in this paper, as it enables us to classify by selecting the HMM model that best explains the observations. With multiple models in the classification step, we can evaluate an observation sequence and recognize its corresponding classification. In addition we can use this to determine when an observation sequence is ambiguous between multiple classes (multiple HMMs report high probability of emitting the same sequence).

# Chapter 4

## Design

### 4.1 System Overview

The goal of the research outlined in this paper is to produce an intent recognition system, which can take the output of a naval simulator, covered later in section 4.2, and produce as output a *guess* as to what intention a ship is likely expressing. Currently, the simulation runs are quite simple. They describe the movements of two ships, which we refer to as the *own ship*, and the *other ship*. The own ship represents our point-of-view; it is the ship for which we are performing intent recognition. The behavior of this ship is non-malicious and benign. In the event of hostile activity, this ship will behave realistically, performing evasive or defensive maneuvers which aim to keep it out of harm's way. In our experiments this ship is always the victim, or target of the other ship.

The other ship is the ship which we wish to observe. From the own ship's perspective, the other ship represents an agent whose intentions are completely unclear. Depending on the experiment, the behavior of the other ship can change drastically. Sometimes it will behave aggressively, sometimes defensively, and sometimes passively. It is also possible for the other ship to begin the simulation in a neutral manner and become aggressive later, or express some combination of intents. In a real world situation, being able to determine what kind of behavior this ship is expressing would prove highly valuable. Being able to predict the other ship's behavior, or at least being aware of it while it is occurring, allows the own ship to formulate counter strategies to factor into its own plans.

The simulated environment in which these ships exist is an infinite plane of water in all directions; the ships move two dimensionally across this space. At present, the only obstacles that exist in the simulation are the two ships; there is nothing else that could impede a ship's movement. Additionally, in this

environment there is no wind, waves, or weather. These variables are unrelated to the problem domain and are not deemed useful features, which also helps eliminate a possible source of noise in the training and testing data.

The own ship is aware of the other ship by way of simulated sensors. These sensors aim to replicate real world instruments, such as radar and sonar. These sensors have no noise, and their readings are guaranteed to be accurate. However, these sensors do have *range*, and thus the own ship is only aware of the other ship when it is within a distance threshold. These sensors capture basic information about the other ship, such as its position, velocity, heading, angle and rotational velocity. The own ship is also presumed to be aware of this information with regards to itself. When the own ship is not aware of the other ships presence, the corresponding portions of the output data are omitted.

At every time step (one second), the simulation produces a record in a file that contains all state information about the own ship, the sensor readings about the other ship, the current simulation time, estimations of CPA (closest-point of approach) time and distance, and a Boolean flag indicative of whether the two ships are colliding. For the purposes of training and testing the current simulation time is ignored, as is the collision flag, since by that time it is too late for the own ship to react, and data collected after that point is not useful.

Although it is theoretically possible for the simulator to work concurrently with the intent recognition system, for the purposes of testing this is currently not the case. All testing (as well as training) is done completely offline. The simulator will record its output data to a file, storing state information for each time step of the simulation. Once all of the needed data has been produced, the file is then passed to the intent recognition model (a completely separate process) for the purpose of training or testing. In the case of training, the intent recognition module will produce a HMM, trained on the input data using the Baum-Welch algorithm. In the case of testing, the intent recognition module will load all user-specified HMMs, will read the input file line by line, and for each line will produce classification likelihoods for each HMM.

## 4.2 Simulated Environments

The use of simulated environments for intent recognition provides many advantages. For instance, the costs associated with producing large amounts of real-world ship movement data would be too high for testing purposes. A simulation mitigates this problem; once it has been created it can be used repeatedly with negligible cost. In addition, the perfectly repeatable nature of a simulator allows examining the results of an experiment with the exact same states and configurations, without introducing any new variables. This proves to be invaluable when attempting to isolate corner cases, or when measuring the effects of changes to the system.

However, in order to produce practical results it is important that the simulation used for intent recognition be as true-to-life as possible. The naval simulator used in this paper performs highly realistic ship maneuvers. This is due to the use of a realistic physics engine, which can correctly simulate a ship's limited maneuverability in the water. The behavior of the ship controlling agents in our simulator is likewise highly realistic, and the behaviors emulate authentic ship operation. The fact that this simulation produces routes that a boat can realistically follow allows for the research described in this thesis to have potential real world applications. A visualization of the naval simulator can be seen in Figure 4.2.



Figure 4.2: Graphical representation of the naval simulation program

### 4.3 Preprocessing

The first task the intent recognition module must perform, once the simulator has produced the output ship intention data, is the cleaning and parsing of the resulting data in order to remove any information which is redundant, incomplete or not useful for the purposes of classification. This process will eliminate data which meets any of the following criteria (to be explained in the following paragraphs):

- Missing data on the other ship (when outside sensor range)
- Not enough history between valid frames
- Violations of assumptions about the structure of the data (mis-labeled data)
- Metadata, or data which implies omnipotent knowledge

Note that the aforementioned criteria do not imply that the data is somehow incorrect, or even incomplete. This is data simply not useful for the purposes of our intent recognition system. The presence of this data, when attempting to perform training, could potentially lead to bias, ambiguity or unnecessary complexity within the resulting model. It is likewise necessary to strip this data out during the testing step, as the form of the observations used to train the HMM, must match the form of the data used to test the HMM.

The removal of incomplete data does not have any effect on the classifier, as there is no situation in which we could potentially recognize intent without the presence of another ship in the simulation. Occasionally, the other ship will enter the own ship's sensor range, only to move beyond it again. In this situation, if there is not enough frame data to provide the needed history, all data will be discarded, as it is considered partial. This is true no matter how long the other ship's absence lasts, even if it is just for one frame. If there is enough history, the data will be kept. Special logic is in place to make sure that history data only contains adjacent frames, so there is no possibility of "gaps" in the data, as a result of the ship leaving sensor range.

The previously mentioned situation in which the other ship leaves the effective sensor range is just one of the possible ways to cause inadequate history between valid frames. Another way the history might be considered insufficient is if the data is too close to the beginning of the simulation, and therefore doesn't have enough history. In this situation the frame being examined is placed into the history, the preprocessor moves to the next frame, and continues to do so until enough history has been created.

Scenarios are generated using Monte Carlo simulation. Because of this, it is sometimes possible to produce data which runs counter to the desired training data. For instance, if the scenario were supposed to be describing a hostile action, there might still be a portion of data in which the ship acts benign prior to engaging in hostile behavior. This section of benign behavior is indeed part of the simulation run, however it is easy to recognize that it isn't representative of hostile behavior. It is the preprocessor's job to scan an input data for periods of time in which the activity of the boat seems to run counter to the expected classification. This is not possible in all cases, as that would require an already functioning intent

recognition model. However, there are a few situations in which it is easy to parameterize intentions; for example, a boat remaining stationary is a good indication of a benign intent. Likewise, a ship moving backwards would obviously not be indicative of a ramming intent. Flagging these inconsistencies keeps the resulting models uncontaminated by data which could hurt the overall performance.

The presence of metadata within the file, such as timestamps and ship IDs, is useful when performing analysis on results or synchronizing intent recognition with video output from the simulator. However, it is not useful for the purposes of intent recognition. All frames of data are separated by the same temporal distance: the time step of the simulation. Therefore, knowing the absolute time is inconsequential for the intent recognition training, as all relative temporal calculations can be made without it. Similarly, the IDs of the ships are convenient for the same reasons, but are ultimately irrelevant to the problem of intent recognition, and are therefore removed by the preprocessor.

The preprocessor also performs basic error checking and validation. With regards to validation, the preprocessor will make sure that the data produced by the simulator obeys the basic laws of physics. If the change in one of the ship's parameters is too large between subsequent frames, a flag is set, warning the operator that the data likely contains errors. In addition, this process also performs basic error checking on all data, including type verification, checking for underflow or overflow, null values, and the presence of Nan and Inf values (not a number, and infinity, respectively). This process ensures that all data entering the intent recognition module is uncorrupted, useful, and will not negatively impact the performance of the resulting training or testing.

## 4.4 Intents

The intent recognition system used in this research was designed, trained and tested using four different intent behavior classifications: *ramming*, *blocking*, *herding* and *benign*. Ramming, blocking and herding are considered hostile

actions, while the benign intent is considered non-threatening. The training and testing files produced by the simulator were designed such that each file encapsulated one intention class.

#### **4.4.1 Ramming**

The ramming intention describes a situation in which the other ship will attempt to make contact with the own ship. This is often characterized by the other ship's heading changing to face the own ship, followed by an increase in the other ship's velocity. In most situations, the own ship will perform evasive action in response to a ramming attempt. This often causes what resembles a *following* pattern (when the other ship appears to be following the own ship) between the own ship and the other ship.

#### **4.4.2 Blocking**

The blocking intention describes a situation in which the other ship attempts to intercept the own ship's path, to prevent it from moving further in that direction. This behavior is usually characterized by the other ship changing its heading toward a point where the own ship will likely travel to in the near future. Once this has been achieved, the other ship will alter its velocity, in order to reach that point at a time slightly before the own ship. The other ship will then attempt to maintain its goal of blocking, should it succeed.

#### **4.4.3 Herding**

Herding refers to an intention in which the other ship attempts to control the movements of the own ship, by strategically maneuvering itself around the own ship. When herding, the other ship will often place itself in a position which would

cause a collision should the own ship maintain its current course, leaving the own ship no choice but to go in the direction the other ship allows. This behavior can be used similarly to block, as a method of area denial. In addition, it can be used to force the own ship to move into areas where it normally would not travel.

#### **4.4.4 Benign**

The benign intention refers to actions by the other ship that are not inherently hostile in nature. Unlike ramming, blocking, and herding, this behavior does not imply that the other ship has any interest in interacting with the own ship in any way. Ships which are acting benign will typically maintain a course of action, unperturbed by the presence of the own ship. In some cases this will mean the other ship will simply be traveling between two locations, and the course it charts will remain constant in the presence of the own ship. This could also refer to situations in which the boat is sitting idle, making no attempts to move anywhere. The nature of this intent makes it slightly more difficult to model, as it encompasses a wide range of potential ship configurations.

The intent recognition model produced in this research could theoretically be applied to any form of intent between the own ship and the other ship, assuming that the behavior has a well-defined, recognizable pattern and the simulator were able to produce representative data. However, the features currently being used for training and testing may not encapsulate the information needed to recognize an arbitrary new intent. As such, additional feature extraction would likely be necessary if applying these methods to new behaviors. The specific features used for the research in this thesis will be covered in more detail in the next section.

## **4.5 Feature Extraction**

Once the data has been through the preprocessor, the next step is to pass it through the feature extractor. Feature extraction is a process which takes as input raw data

and computes a set of derived values, which aim to capture useful information while being non-redundant. In the context of our research, this means computing features which will help differentiate one intent from another during the testing stage. The features derived from feature extraction are typically used as input to subsequent layers of machine learning algorithms. In our approach, features are used as input to the Baum-Welch algorithm, in the case of training an HMM, and are used as the input to the evaluation step, in the case of testing an HMM.

During the feature extraction phase, our intent recognition module computes seven features based on the input data. The computed features are the following:

- An indication whether the other ship is facing toward the own ship. This is a Boolean value, set by using a constant angle threshold, and can assume the values *facing toward* or *not facing toward*.
- A measure of the change in absolute position of the other ship relative to the own ship; this feature can take three values: *closer*, *farther*, or *relatively stationary*.
- A measure of the change in velocity of the other ship, which can take on the values of *accelerating*, *decelerating*, or *constant*.
- The change in angle between the own ship and the other ship. This feature can assume the values of *turning toward*, *turning away*, or *constant*.
- The change in relative heading between the two ships (that is, the angle between the own ship's heading and the other ship's heading), with the possible values of *increasing*, *decreasing*, or *constant*.
- A measure of the change in the closest point of approach time or CPA time. CPA time refers to the time when the two boats are closest to each other. This could be either in the past, in which case the CPA time will be

negative, or in the future, in which case the CPA time have a positive value. This feature can take on the values of *positive* or *negative*, indicating whether or not the boats have already moved beyond their closest proximity.

- A measure of the change in the closest point of approach distance, or CPA distance, between the own ship and the other ship. CPA distance is a measure of the closest two points which connect the own ship to the other ship. This value is compared against a threshold, and can take on the value of *below threshold*, *above threshold* or *equal to threshold*.

Notice that a majority of these features are measurements of change (called delta features) and not of absolute values. Delta features can be considered estimates of the first order derivatives of the features [30]. The advantage of using delta features is that they transform a contextual measurement, which would otherwise be specific to an individual situation, into a relatively stable point in a new feature space. This allows for easier comparison across unique scenarios. Notice also that these delta features are further discretized, or binned, into predefined, finite categories. This essentially reduces the dimensionality of the program into a vector of size seven, where each element of the vector can take on either two or three values. There are a total of 324 ( $2 \times 3 \times 3 \times 3 \times 3 \times 2 \times 2$ ) possible states that the feature vector can take at any given time. This is beneficial, as it reduces a large continuous problem space into a simple, compact discrete problem space, while retaining important discriminatory information. This in effect reduces the complexity of the resulting model.

There is one final step in the feature extraction process: the encoding of temporal information. In order to facilitate this, before the feature data is fed into the training or testing step, it is grouped together with its nineteen preceding frames of information, combining all 20 into a single feature. This allows the intent recognition system to treat these 20 frames of data as one atomic section, allowing the HMM to make its determinations based on the current state, as well as a small amount of history leading up to the current state. This expands our ability to

classify intent behavior, as it gives additional context to the current state of the simulation. Knowing what specific actions are likely to precede other actions is useful for recognizing patterns while they occur. This in turn greatly bolsters the system's ability to not only recognize behaviors, but predict them as well. Thus, the input features for our intent recognition system take the form of a vector of size 20 representing history, where each element is a discrete value between 1 and 324, representing the configuration of the ships at that time.

## 4.6 Training

Once the data has been preprocessed, and the result run through the feature extraction subsystem, the feature data is ready to be used for HMM parameter estimation (training), a process which was discussed in detail in Section 3.2. The training step begins by specifying the location of the input classification data; this is a directory which contains the files to be trained on (the files generated from the simulator). The user will also pass the desired output HMM file name to the system, as an argument. As an optional parameter, the user can specify the desired number of hidden states present in the HMM to be trained, defaulting to 5. The optimal number of states for a given problem must be found through experimentation.

The system will then load the input files, one at a time, and begin to perform feature extraction on each. Simultaneously, the program will take the computed features as they are made available, and use them to update the HMM. This is done by passing the features into the Baum-Welch, which will perform expectation maximization in order to update the HMM's parameters. This updating process is repeated for every datum of every input file, until all files have been processed. The result is then saved to the disk.

It should be noted that the system does not have to start from scratch when computing the model parameters of an HMM. It is possible to provide the program with an existing HMM as an argument, which can be loaded into the system upon startup. This enables the Baum-Welch algorithm to use the existing HMMs

parameters as an initial guess. This allows the user to improve upon existing HMMs by introducing new data which may not have been available during the initial training run.

## 4.7 Testing

Assuming that an HMM has been trained for each of the desired intents, it is then possible to perform the testing step of the intent recognition system. The testing process is initialized similarly to the training processes. The user will specify the location of a set of testing files (which do not necessarily correspond to only one class), and the system will begin to load them into memory, one at a time. The feature extraction subsystem will then load individual records from this file, line by line, and produce the corresponding feature data. As it is generated, the feature data will be run through a suite of HMM evaluations, one for each of the trained intent classifications. This will produce a vector of probability information, indicative of how likely it is that each model produced the observation sequence (feature). This process of evaluation was discussed in detail in Section 3.4. The resulting probabilities are then normalized against the sum of the probability vector, producing probabilities for each intent class which sum to a 100 percent.

The end result of the testing process is a collection of probability reports, one for each of the input data files. The output report will contain prediction information for every applicable time step of the simulation data, giving probability estimates for each of the classes available. These files can be used for validation testing, or used as heuristic method of measuring performance against revisions. In addition to performing batch analysis of simulation data, it should be noted that the testing system is also potentially capable of real-time predictions. This could be for either a real world ship or a simulator, provided that some form of communication logic is present between the intent recognition system and the source of the ship sensor data. If provided, it would be trivial to run the intent recognition system on single fields of data as they became available in real time.

# Chapter 5

## Results

### 5.1 Overview

In this section we analyze the results of the intent recognition module against labeled data produced by the simulator. The data used for these tests is novel to the intent recognition system, meaning the HMM model is classifying new data based on what it learned during the training stage. The results in this section are split into four experiments, one for each of the intent behaviors. In each section a brief overview of the system's overall performance is provided, on an intent by intent basis, followed by a closer examination of two representative test cases from each intent classification. The results will be provided graphically in two formats: a composite line graph, showing the probability of each intent at every time frame of the simulation run, and a confusion matrix displaying the rates of classification and misclassification for each intent. In addition, a simple scatter plot will be provided to give an idea of the ship's paths over time.

Overall the results presented in this section are quite strong. It is very common for the intent recognition system to produce results which are accurate more than 90% of the time. Due to the nature of the intent space (four possible intent classifications), guessing would result in an accuracy of approximately 25%. The behavior classifications for ramming, blocking and benign behavior were correct most of the time, failing to provide correct predictions only when ship maneuvers became similar to other intents. The herding behavior's results were quite poor, with results often performing only slightly better than the guessing case. The reason for the herding behavior's poor results was found upon examining the data, and will be discussed in Section 5.4.

When viewing the prediction result graphs in this section, it is worth noting that there may be an interval at the very beginning of the time series data, when no

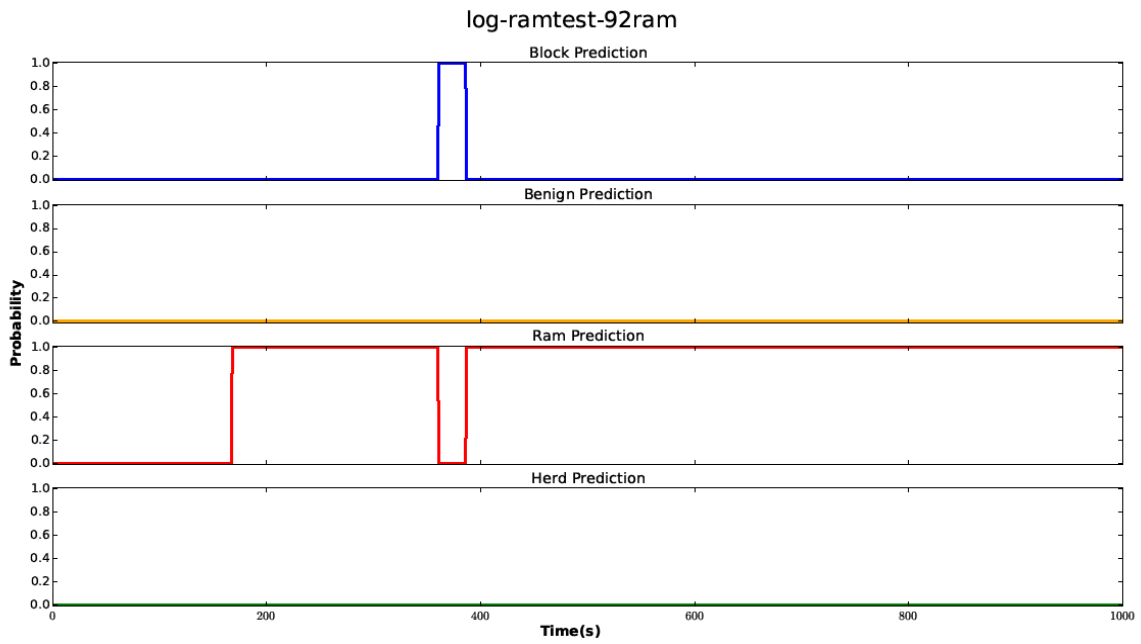
prediction is being made. This is expected behavior, as this section of data represents the time it takes for the own ship and the other ship to get close enough to each other so that they are within instrumentation range. However, this interval is non-constant, and in fact may not exist in certain situations when the boats begin the simulation at a range small enough. In addition, once a boat is within range the system will wait 20 frames before producing intent classification data, as this period of time accounts for initializing the history buffer with valid previous frames, necessary for the intent recognition system to function.

## **5.2 Experiment One: Ramming Behavior**

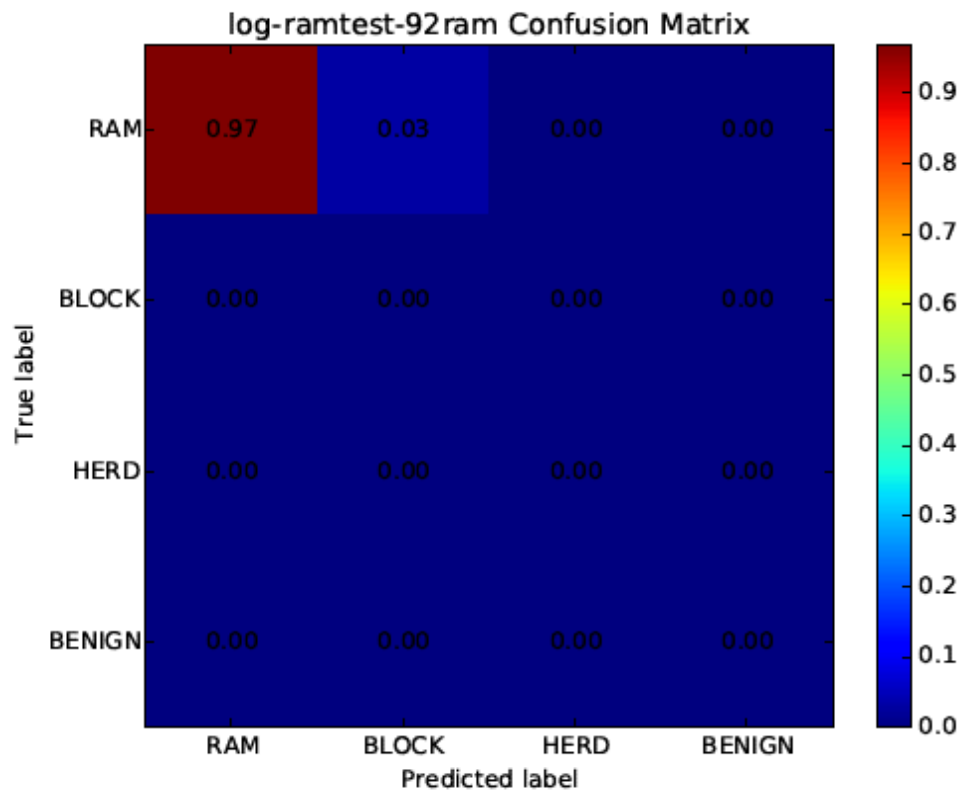
Recognizing ramming behavior proved to be the intent recognition system's strongest use case, averaging an accuracy of 91% over 20 unique runs. This performance is likely due to the well-defined, unambiguous nature of the ramming maneuver. When the other ship is attempting to ram, the actions which it performs are distinct from the other three intent classifications, thus the possibility for confusing a ram with another intent is low.

The largest source of false classifications comes from confusion with the blocking behavior. This confusion stems from the other ship's change in movements as it gets close to the own ship. When the distance between the two ships is low enough, the other ship often has to perform severe correction to its path, in order to continue approaching the own ship. This particular maneuver is similar to behaviors made by ships attempting to block. Once this correction has been made, ramming will usually become the dominant intent again. Ramming may occasionally be misclassified as herding and benign, however these cases are very rare and never last for any significant period of time.

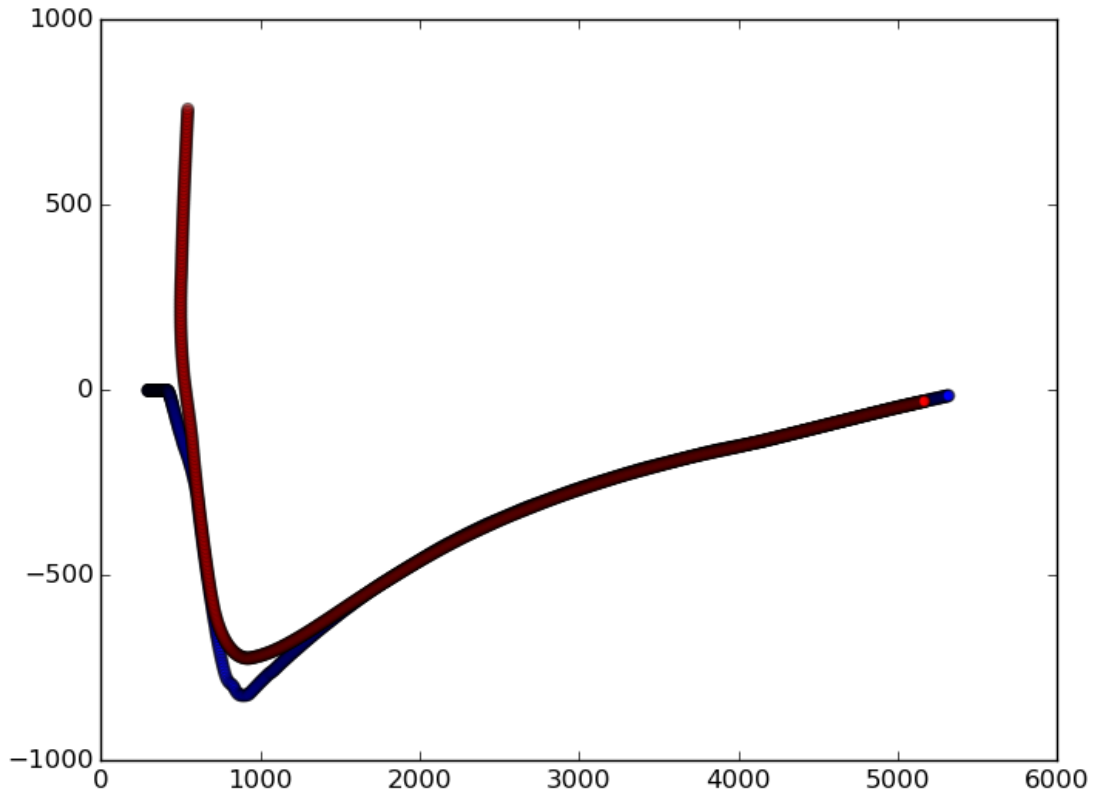
### 5.2.1 Ram Test #1



**Figure 5.2.1:** Prediction results for Ram Test #1



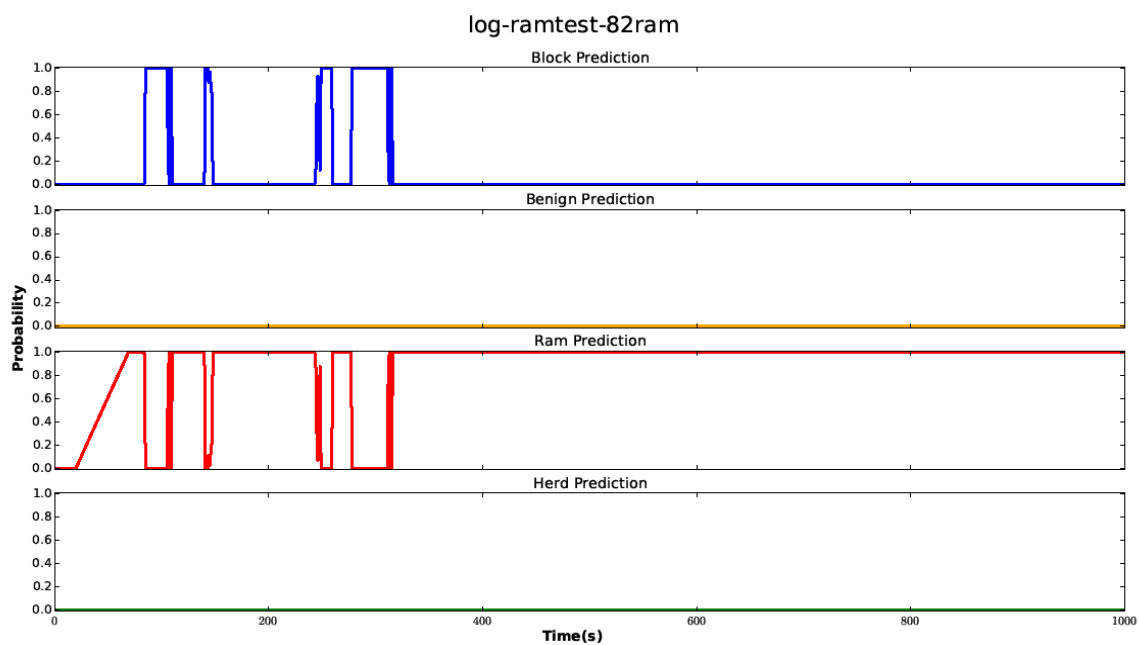
**Figure 5.2.2:** Confusion Matrix for Ram Test #1



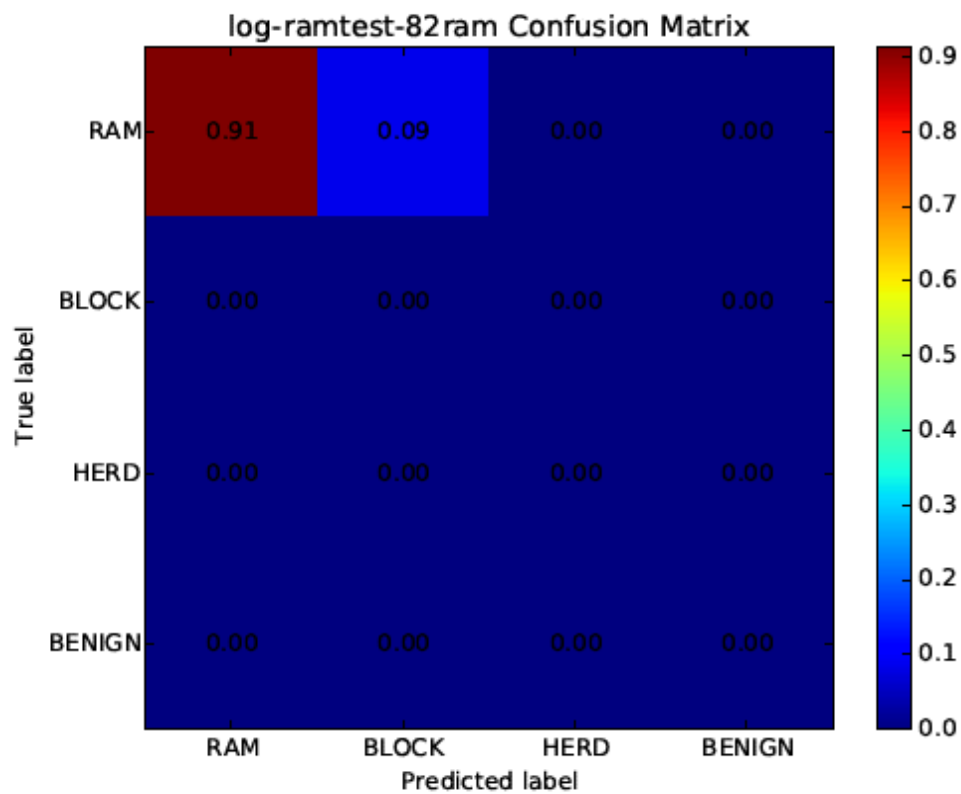
**Figure 5.2.3:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Ram Test #1

Test case #1 is an exemplary case of the ram behavior being recognized correctly and consistently. This is a straightforward case of the other ship approaching from a distance, with the intent to ram into the own ship. In this test case the ram behavior is being predicted correctly 97% of the time, while the remaining 3% is due to a misclassification with block when the two boats begin to get close to one another. Once this period has passed and the other ship has corrected its trajectory, ram is correctly recognized for the remainder of the test. There is no misclassification with herd or benign in this test case. Also, note the appearance of the following pattern, mentioned in section 4.4.1.

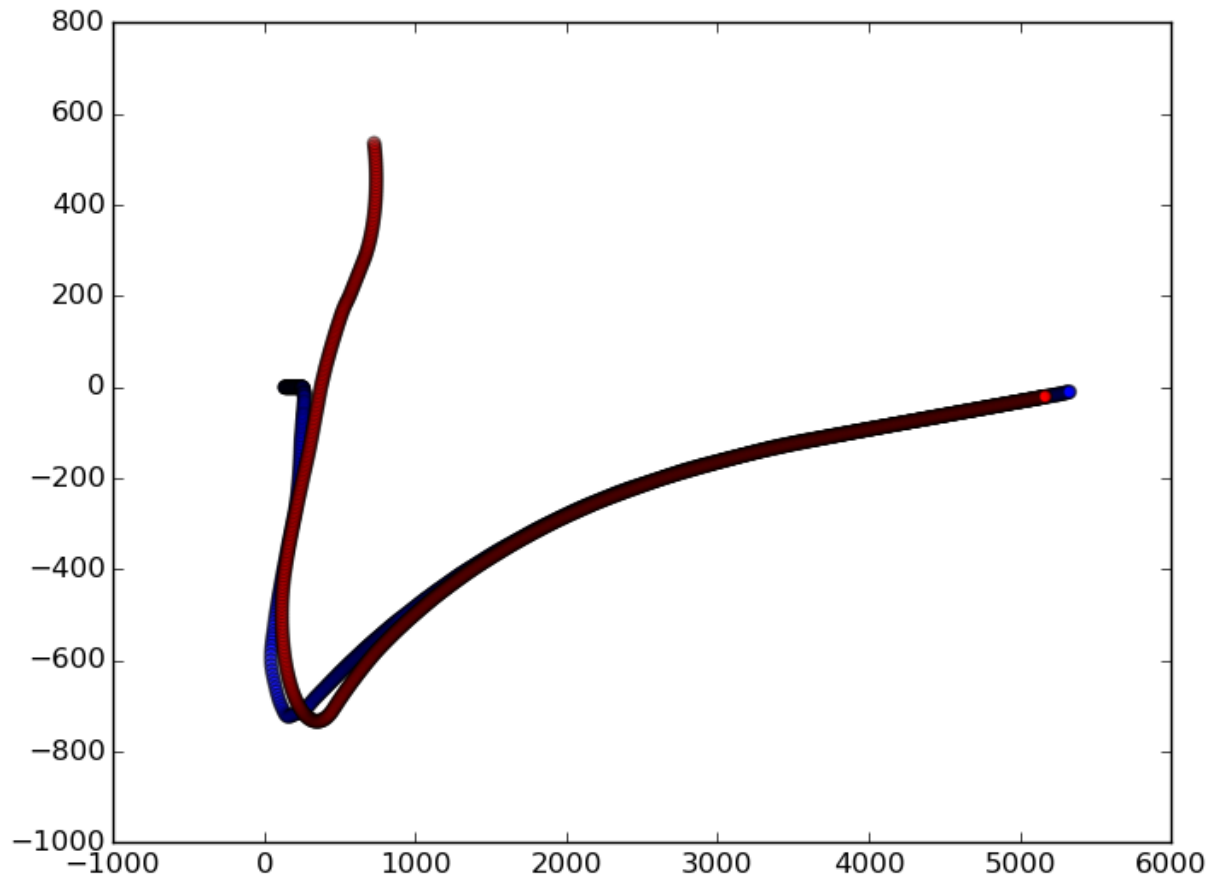
### 5.2.2 Ram Test #2



**Figure 5.2.4:** Prediction results for Ram Test #2



**Figure 5.2.5:** Confusion Matrix for Ram Test #2



**Figure 5.2.6:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Ram Test #2

Test case #2 demonstrates a slightly more complex ramming situation; in this test there is a large period near the beginning of the run, in which the intent recognition system changes its predictions from ram to benign, and vice versa. In this test cases, the two ships began the simulation in closer proximity than in test case #1. Because of this, the confusion with block occurs earlier in the run. Also worth noting is the oscillatory behavior of the confusion at the beginning of the run, this is due to evasive actions taken by the own ship when the other ship came too close. This in turn caused the other ship to make a correction, before engaging in normal ramming behavior. Overall, test #2 did not perform as well as test #1, having only 91% accuracy, where accuracy is the ratio of correctly predicted frames to total frames. This result is still quite good. The drop in accuracy in this test is likely due to the more complex nature of the two ship's interactions, when compared with test #1.

## 5.3 Experiment Two: Blocking Behavior

Recognizing blocking behavior consistently proved to be a slightly more challenging process than recognizing ramming behavior. This is due to the fact that blocking is not as conceptually simple as ramming, thus making it more difficult to recognize in all cases. Whereas ramming is simply characterized by one ship attempting to occupy the same location as another ship, blocking behavior implies a slightly more advanced level of tactic, characterized by one ship attempting to prevent another ship from entering or leaving a certain area. Due to this increased complexity, there are situations in which it is possible to confuse a blocking behavior with any of the other three intent classes: ram, benign or herd. Overall the accuracy of our model is still high, averaging roughly 74% across 27 test cases.

Inaccuracy in these test cases most often comes from misclassification as benign behavior. This confusion is due to the fact that in many cases a blocking behavior can be similar to a benign behavior in the early stages before the other ship comes within close proximity of the own ship. Initially, in both the benign and block behaviors, the other ship may appear to be harmlessly traveling to a location, without necessarily appearing to engage the own ship. It is only when the gap between the two ships closes, that the blocking behavior manifests more clearly. A similar situation occurs with the ramming behavior, once the two boats are in very close proximity, as attempts to block the own ship from moving can be seen as a ramming attempt when there is little distance between the two. Blocking behavior is almost never misclassified as herding behavior.

### 5.3.1 Block Test #1

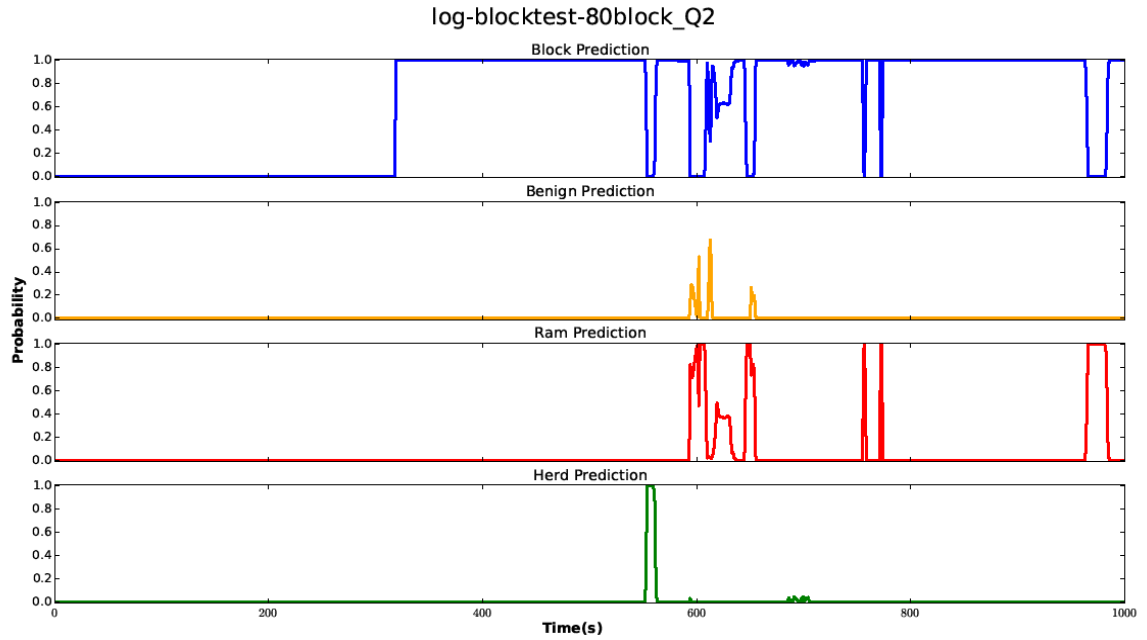


Figure 5.3.1: Prediction results for Block Test #1

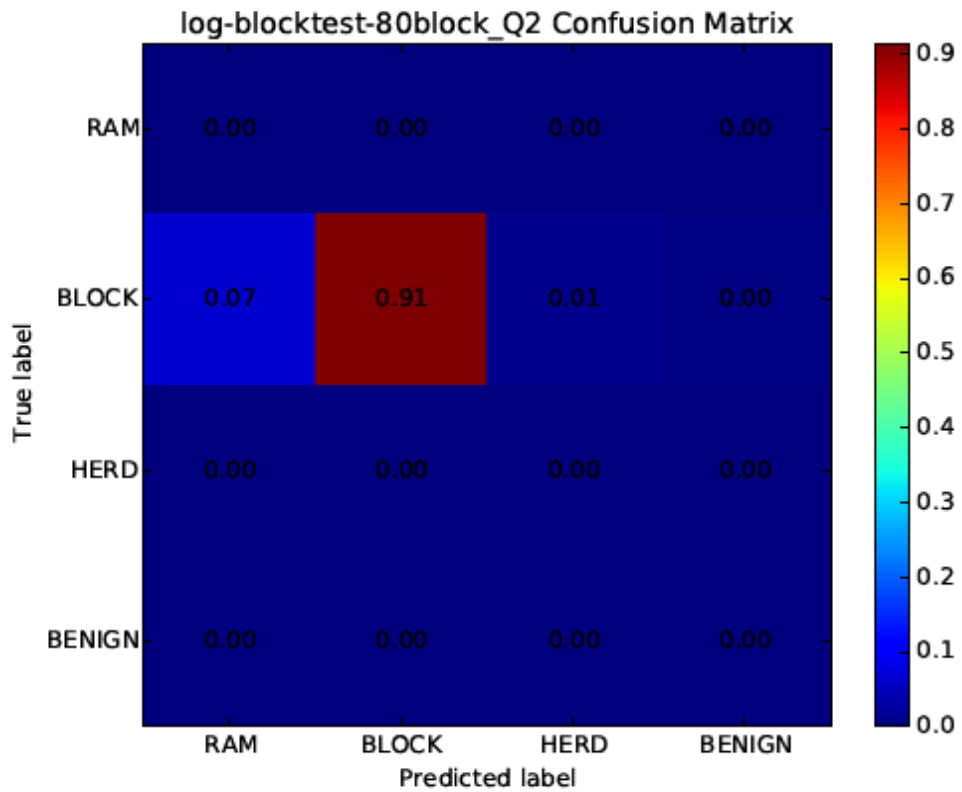
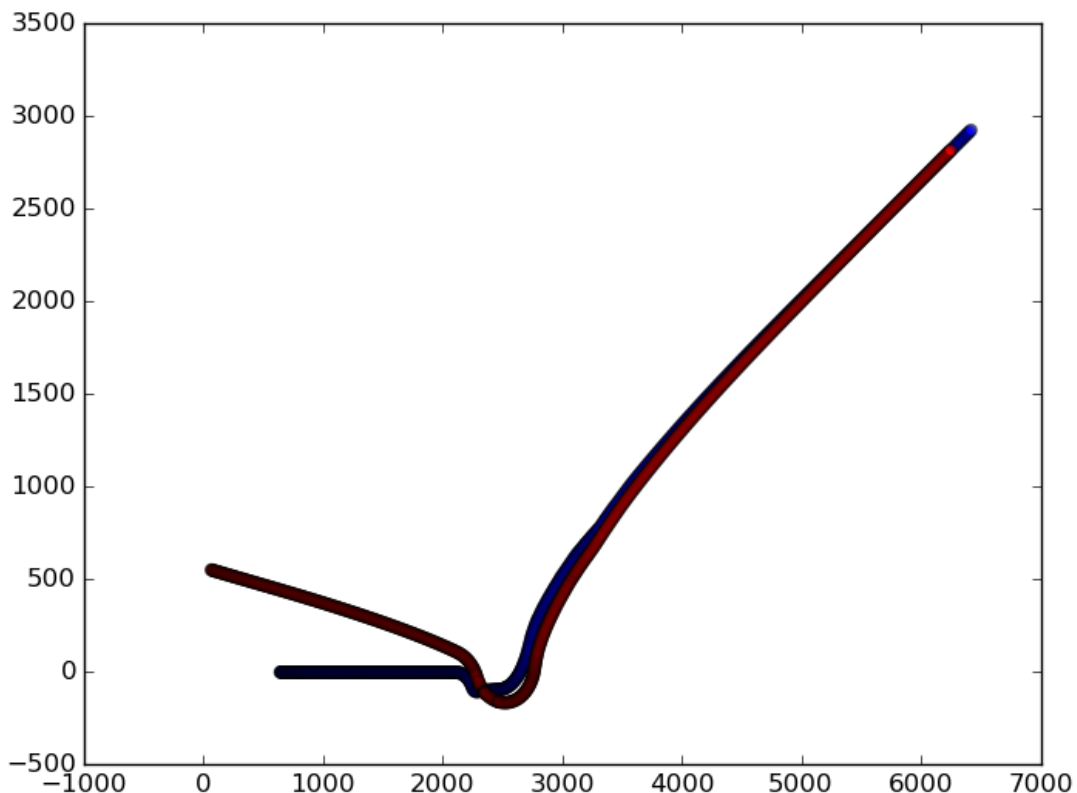


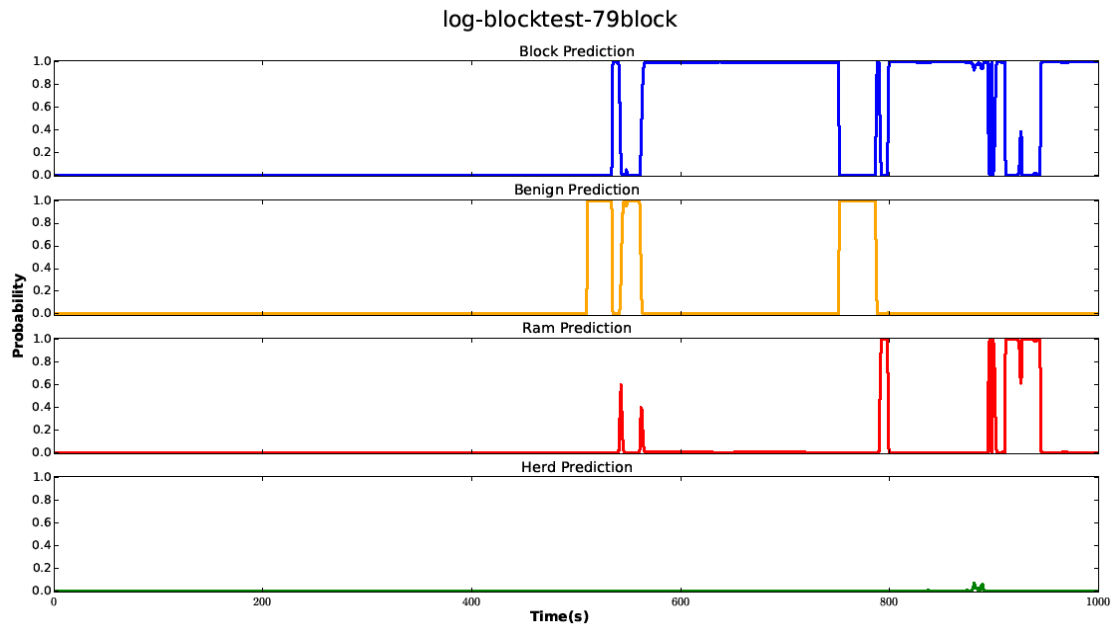
Figure 5.3.2: Confusion Matrix for Block Test #1



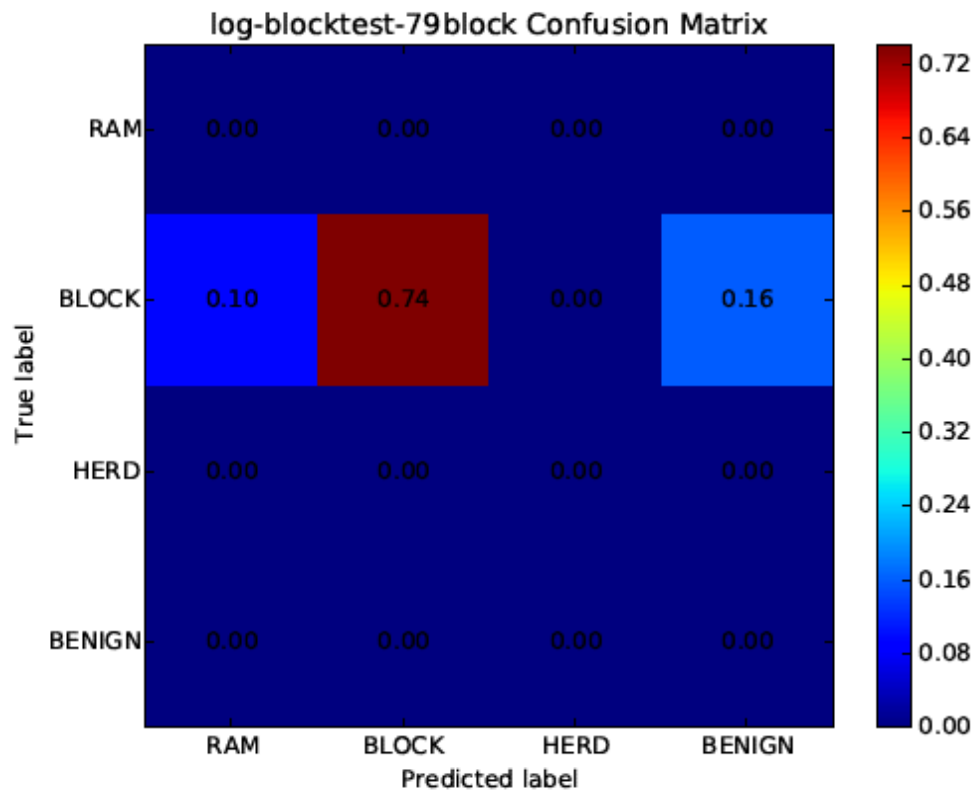
**Figure 5.3.3:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Block Test #1

Test case #1 demonstrates a blocking scenario with high accuracy and low ambiguity, with an overall accuracy of 91%. This run begins with the intent recognition immediately, and correctly predicting block as the intent of the other ship. Despite the high accuracy of this run, this test case includes situations in which the blocking behavior is misclassified as each of the other intent classes, at one point or another. As would be expected, these misclassifications begin to manifest more often as the boats get closer in proximity. During this period, the intent recognition system predicts a brief period of herding behavior, followed by a period of high confusion during which ram, benign and block are all given non-zero probabilities. It should be noted that during this period, block is still predominately favored by the intent recognition system, over the other behaviors. The remainder of the run is mostly correctly classified as block, with a few spikes of ram intent. All the aforementioned periods of misclassification occurred during periods of complex interaction between the two ships, in which one ship would be changing its behavior in reaction to the offensive, or defensive actions of the other.

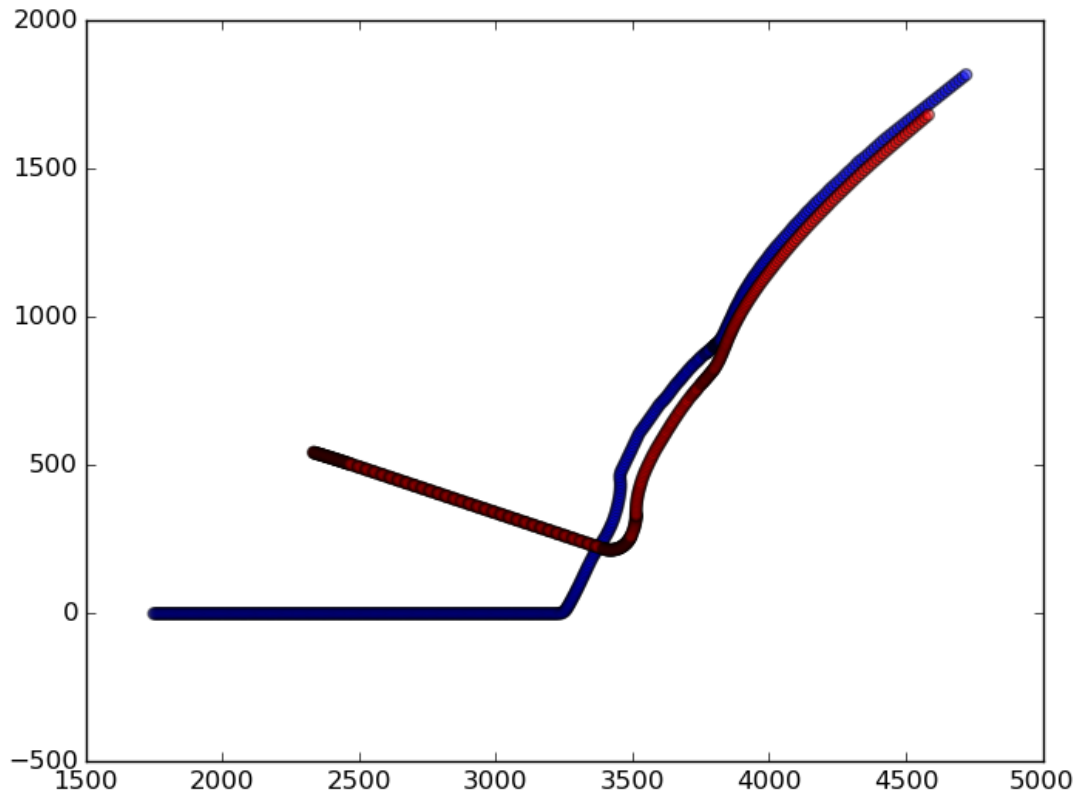
### 5.3.1 Block Test #2



**Figure 5.3.4:** Prediction results for Block Test #2



**Figure 5.3.5:** Confusion Matrix for Block Test #2



**Figure 5.3.6:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Block Test #2

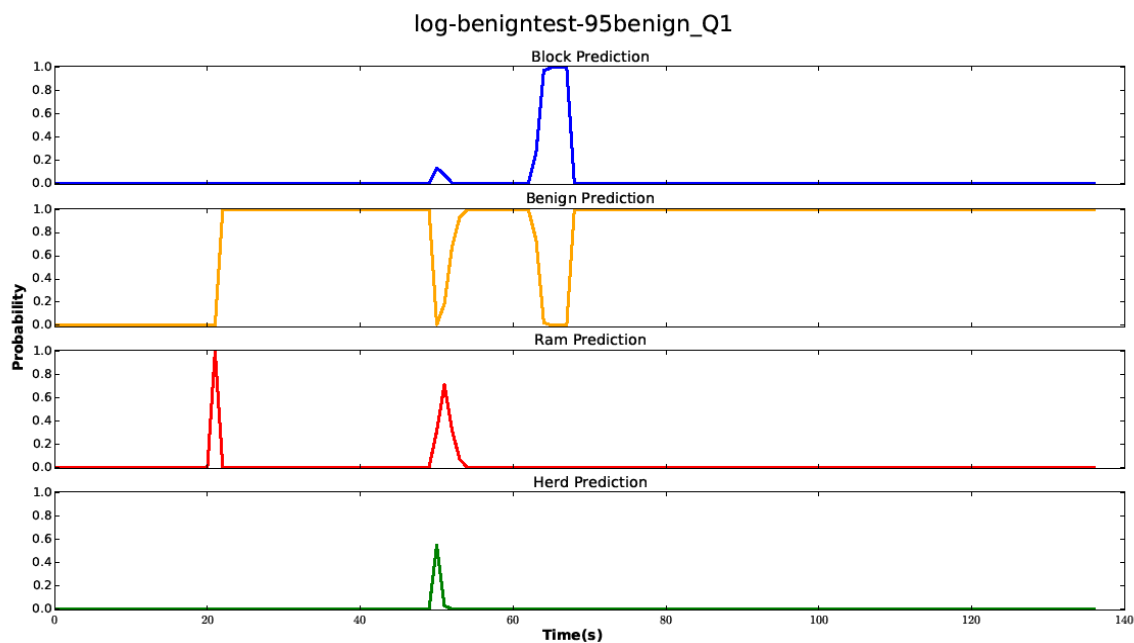
Test case #2 demonstrates a more contested run, during which there are larger periods of misclassifications by the intent recognition system. Unlike the saw-toothed nature of the misclassifications in test #1, here the system appears to be more certain of the misclassifications it makes, with longer periods of intent stability. These misclassifications fall into the structure mentioned at the beginning of Section 5.3, with benign misclassifications occurring more frequently toward the beginning of the simulation (when the other ship is making its approach), and ram misclassifications occurring toward the end of the simulation, when the other ship's behaviors are more likely to appear as a ram, due to a following behavior.

## 5.4 Experiment Three: Benign Behavior

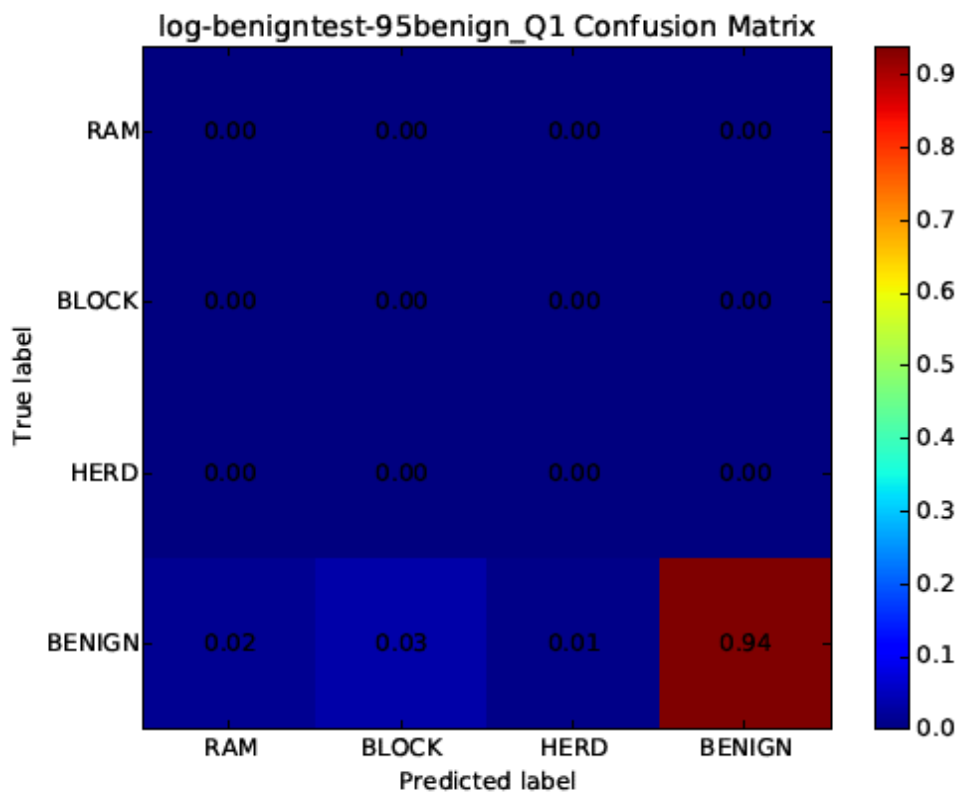
Recognizing benign behavior proved also to be a case in which the intent recognition system provided high accuracy a majority of the time. On average, the intent recognition system has an accuracy of 85% across 20 simulation runs. Benign behavior is somewhat difficult to formalize, as it is simply the scenario in which the other ship is not performing any form of hostile action toward the own ship. This could mean a wide variety of actions, such as passing by, moving away, or sitting idle in the water. Because of this, benign behavior is analogous to an “everything else” category, describing situations that do not meet the specifications of ram, block or herd.

The benign behavior is most often confused with a block, likely due to situations when the other ship happens to be moving toward a location in the own ship’s path. In situations where the two boats are already in close proximity, this could easily be seen as a blocking behavior. Less often this behavior can be misclassified as a herd, or even less often as a ram.

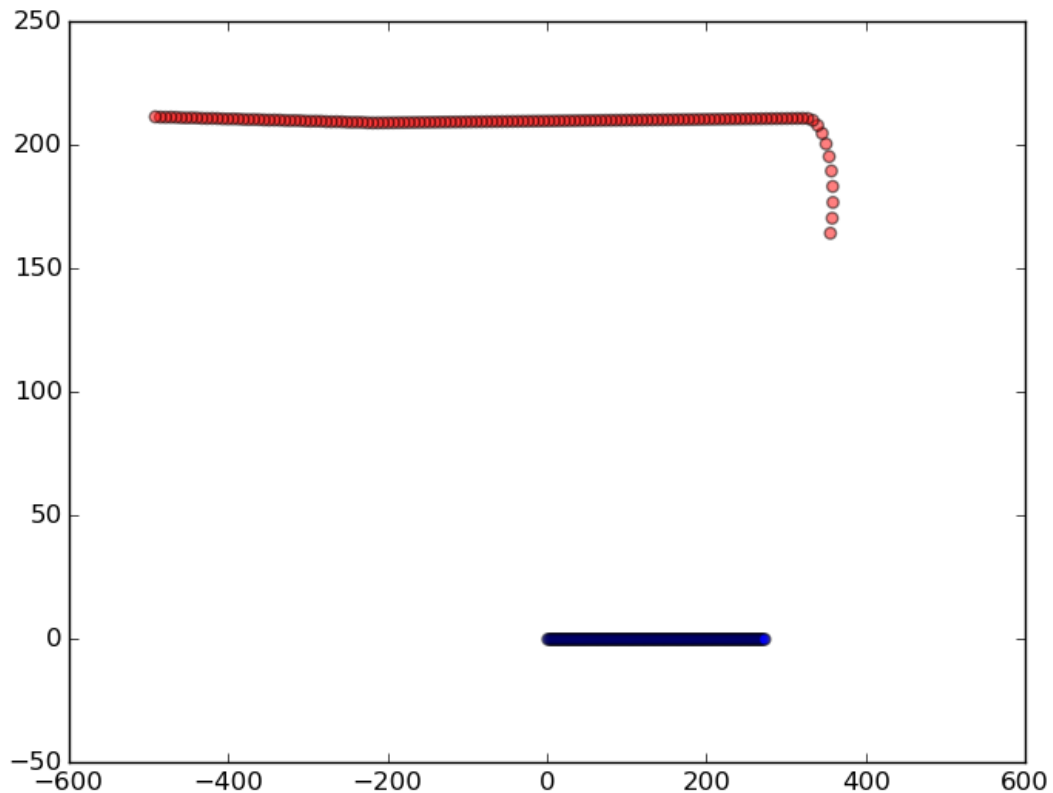
### 5.4.1 Benign Test #1



**Figure 5.4.1:** Prediction results for Benign Test #1



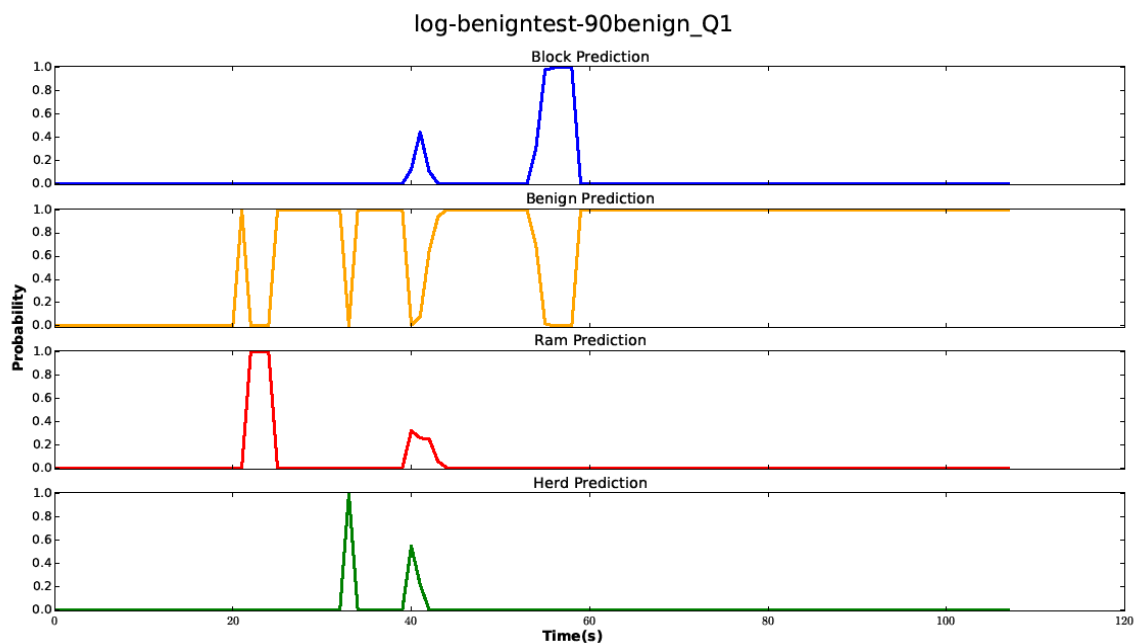
**Figure 5.4.2:** Confusion Matrix for Benign Test #1



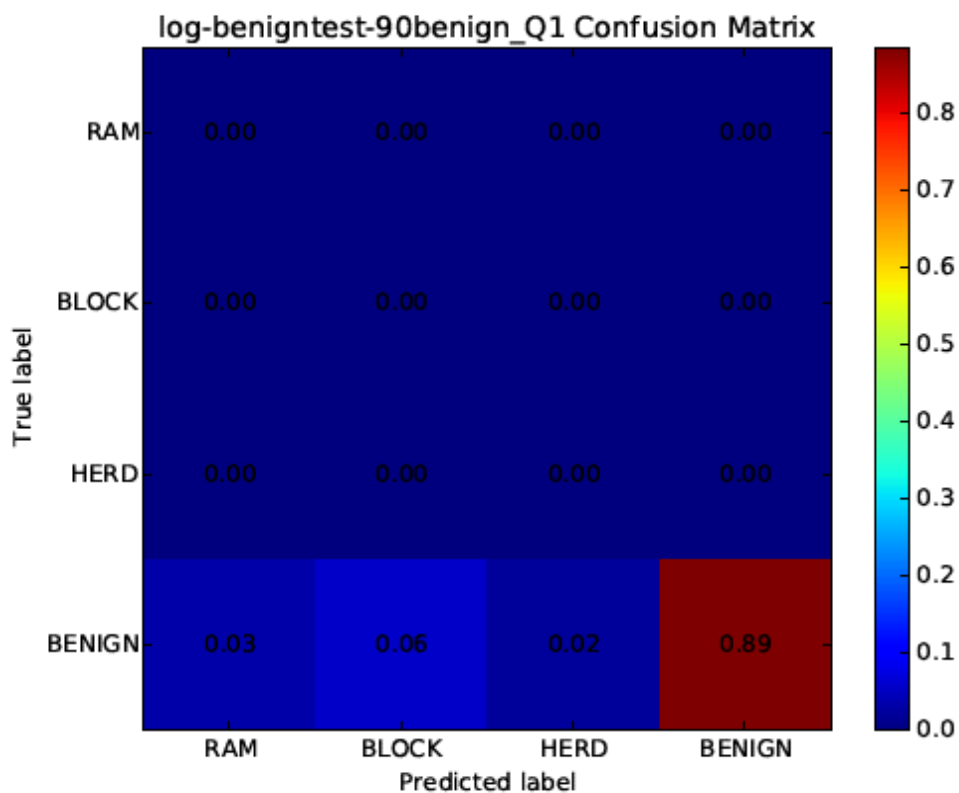
**Figure 5.4.3:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Benign Test #1

Benign Test #1 demonstrates a benign run with 94% accuracy. Despite the fact that there is misclassification with all three other intent classes, none have very much persistence in the prediction results. One interesting fact about this run is the existence of an interval when the system believes the current behavior is a combination of ram, herd, and block, without any probability of the correct behavior of benign. This period of confusion only lasts for a couple of simulation frames, but investigating the cause of this behavior might be worth additional research. Once this period is over, there is another small interval where blocking is the dominant behavior, followed by the remaining half of the simulation being correctly and consistently classified as a benign run. This simulation run can be described as a scenario in which two boats pass by each other, while going opposite directions. It is during the short period of time when the two boats are passing next to each other that the misclassifications in the run begins to manifest. Of course, once this period passes the system returns to correctly identifying the benign behavior.

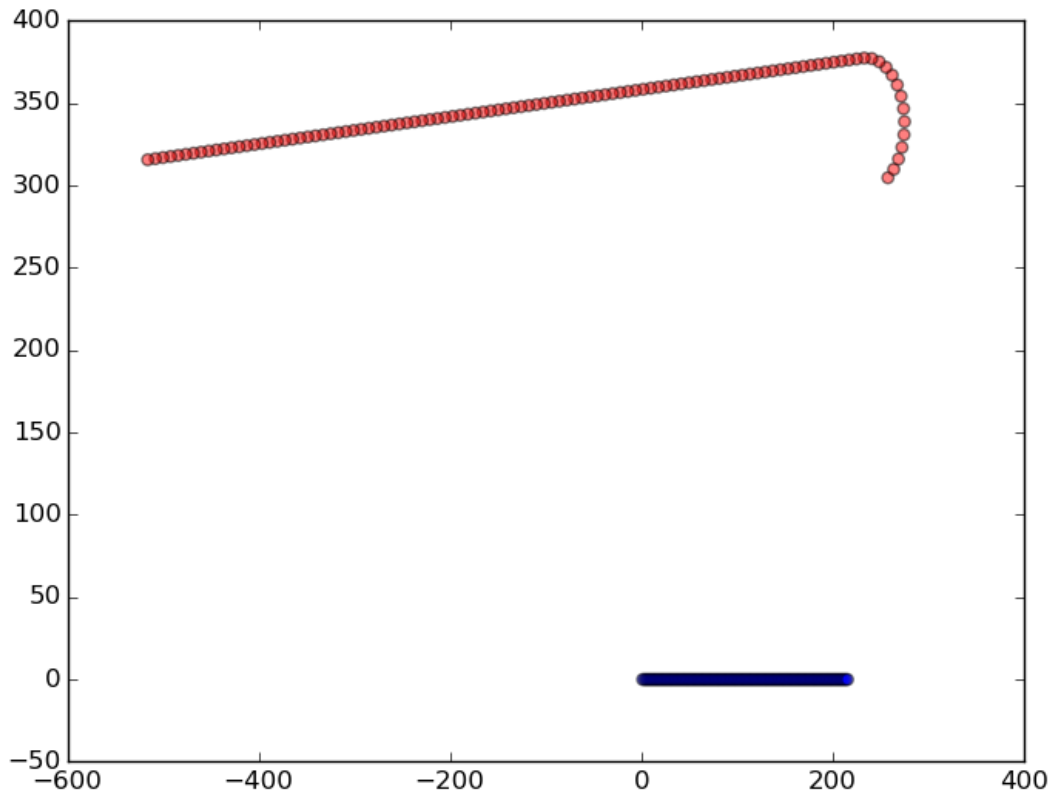
### 5.4.1 Benign Test #2



**Figure 5.4.4:** Prediction results for Benign Test #2



**Figure 5.4.5:** Confusion Matrix for Benign Test #2



**Figure 5.4.6:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Benign Test #2

The second test case performs slightly worse than the first, with an overall accuracy of 89%. This case demonstrates the system going through a period when once again misclassifications of every kind are present. Despite roughly 11% of the simulation being misclassified, the intent recognition system is never wrong for a prolonged period of time. Most misclassifications occur briefly before the system returns to making the correct prediction of benign. Similarly to the first test cases, misclassifications in test #2 tend to occur in the first half of the simulation, before the two ships move farther away from one another. Notice the very similar movement patterns between the two scenarios in Figures 5.4.3 and 5.4.6.

## 5.5 Experiment Four: Herding Behavior

Herding behavior proved, by far, to be the most difficult behavior class to correctly recognize. Overall the accuracy of this intent class was 30% over 26 individual test cases. The reason for the poor accuracy of this intent turned out to be a problem with the generated data. Upon analyzing the files generated for the training of the herd classifier, it was noticed that all of the files produced for both training and testing contained large sections of data which were indistinguishable from the benign behavior. This was due to the nature of the initial states of the ships, which caused a benign run to look identical to a herd run for a bulk of the simulation. Only when the two ships came within close proximity did the two behaviors begin to disambiguate themselves. As a result, the Hidden Markov Model used in these tests was trained on data which was only partially useful for the herd intent.

In some cases, more than 80% of the data produced for these files was found to be ambiguous with benign. This caused the benign HMM to produce stronger classification for these situations, due to the fact that it was trained solely on the benign behavior and not on a mixture of herd and benign. The nature of the data made it difficult to isolate portions which solely demonstrated herding behavior, as this would leave only a small amount of data to work with. This revelation became known very late in development, and as such it was not possible to request new herding data within a reasonable timeframe.

However, there are certain patterns in the prediction results, which upon analysis help separate herd runs from benign runs. The most prominent of these patterns appears at the very end of a simulation run, during the period when the two ships are closest. For most of our tests cases, the final stretch of intent data would show herd as the predominant behavior. This behavior is highly consistent across our testing runs, and makes intuitive sense since only when the two ships are close a herding action begins to differentiate itself from a benign ship which is simply passing by the own ship.

### 5.4.1 Herd Test #1

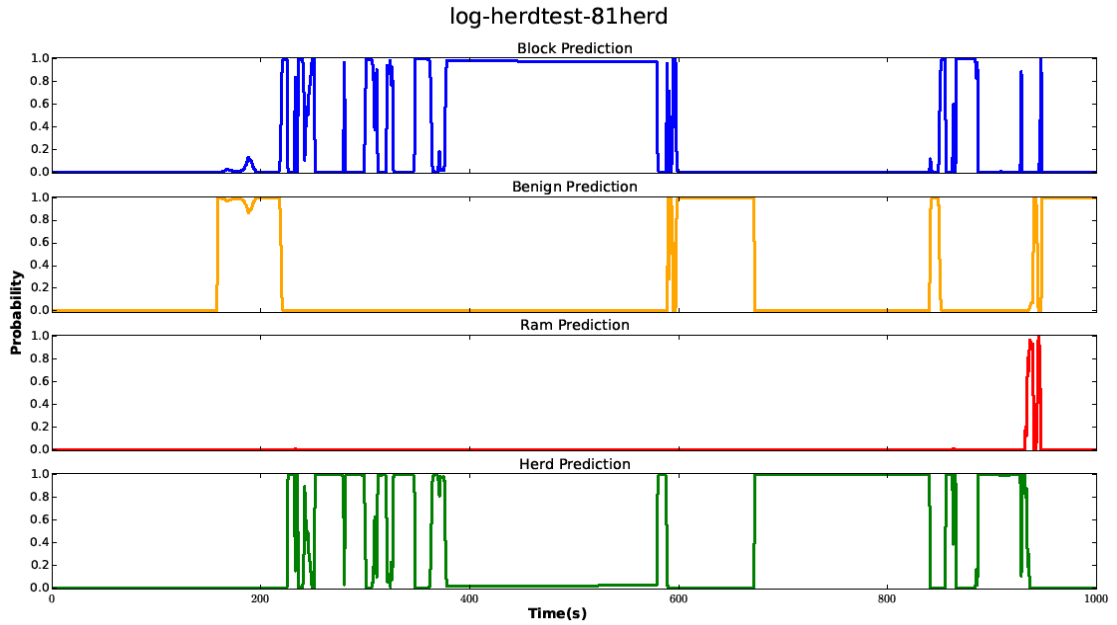


Figure 5.5.1: Prediction results for Herd Test #1

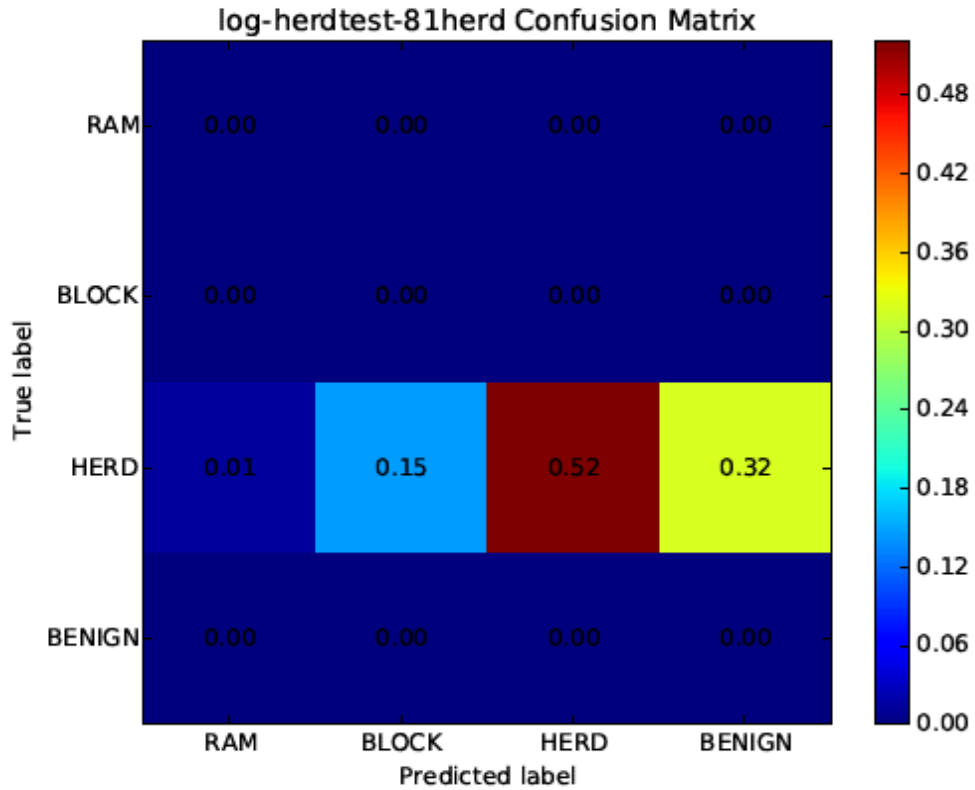
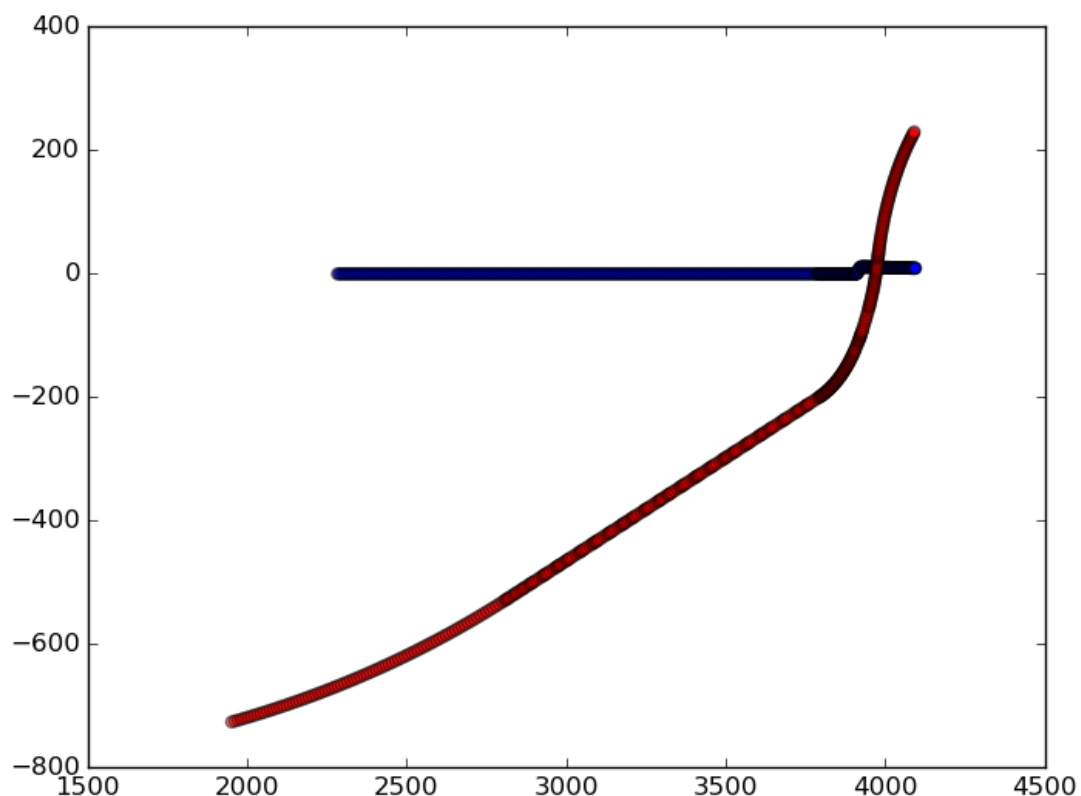


Figure 5.5.2: Confusion Matrix for Herd Test #1



**Figure 5.5.3:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Herd Test #1.

The first herd test cases shows a situation in which the herd behavior is performing above average. In this run, the simulator predicted the herd behavior more than half the time (52%). However notice that benign classifications are also quite high (32%), demonstrating clearly the ambiguity between the two classes. Also notice how the largest section of continuous intent class is benign, and not herd. By observing the probability results it is clear that the intent recognition system had no consistent recognition for this run, instead constantly switching between herd, block and benign. This is due to the fact that the training data did not isolate any one unambiguous pattern. However, as stated previously, herd behavior does become the dominant probability at the very end of the simulation run. This is the only discernable, consistent pattern among the training data.

### 5.4.1 Herd Test #2

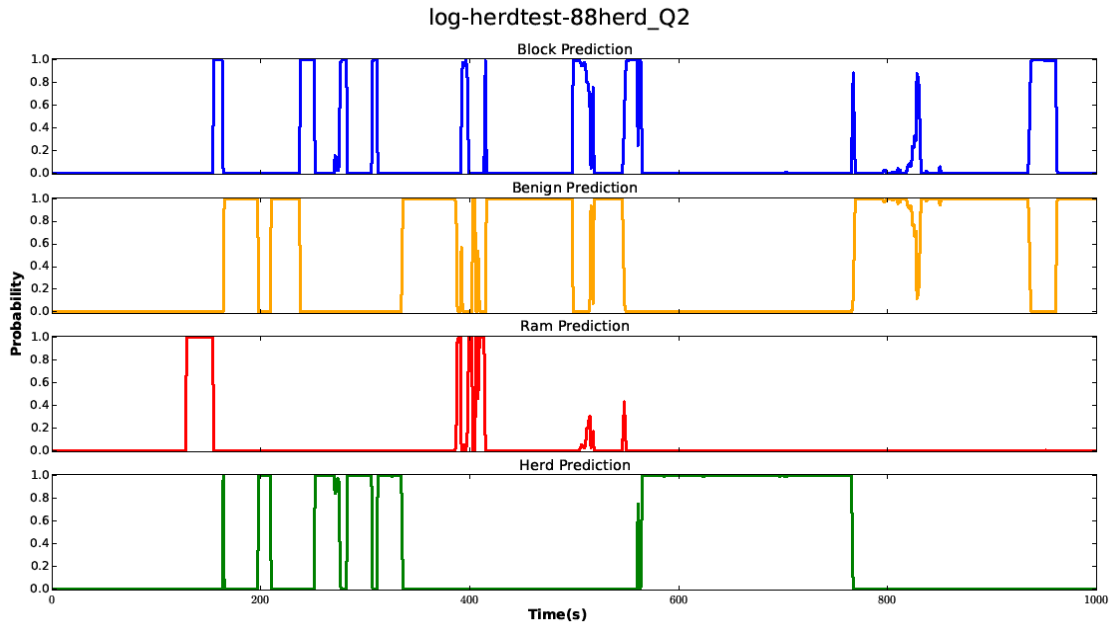


Figure 5.5.4: Prediction results for Herd Test #2

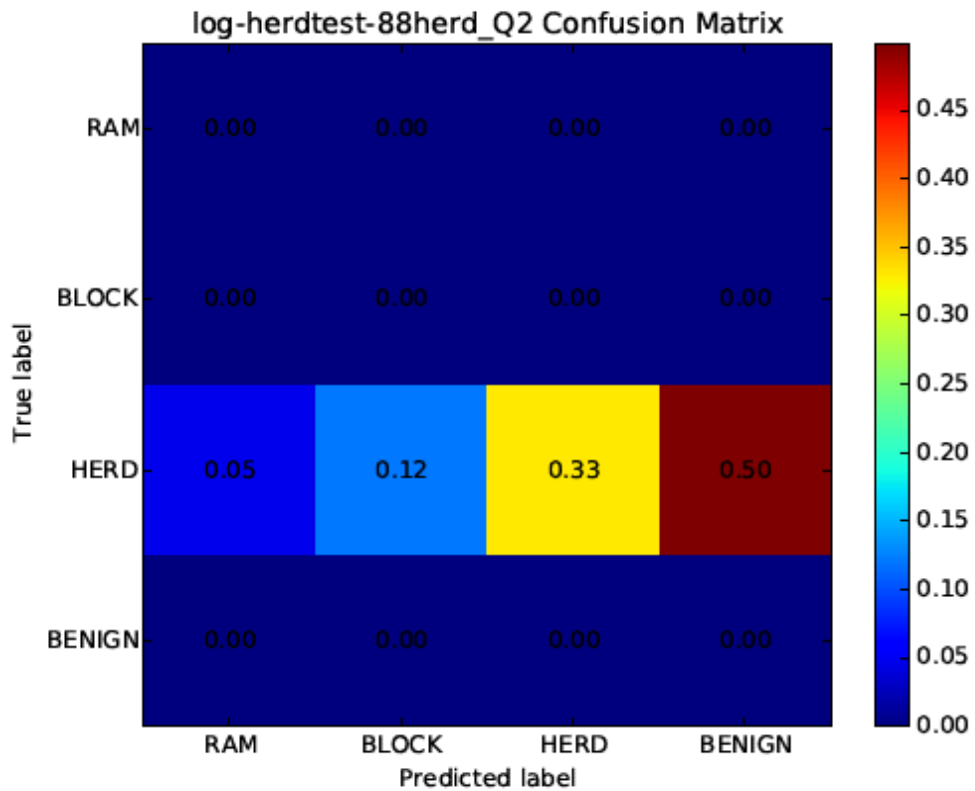
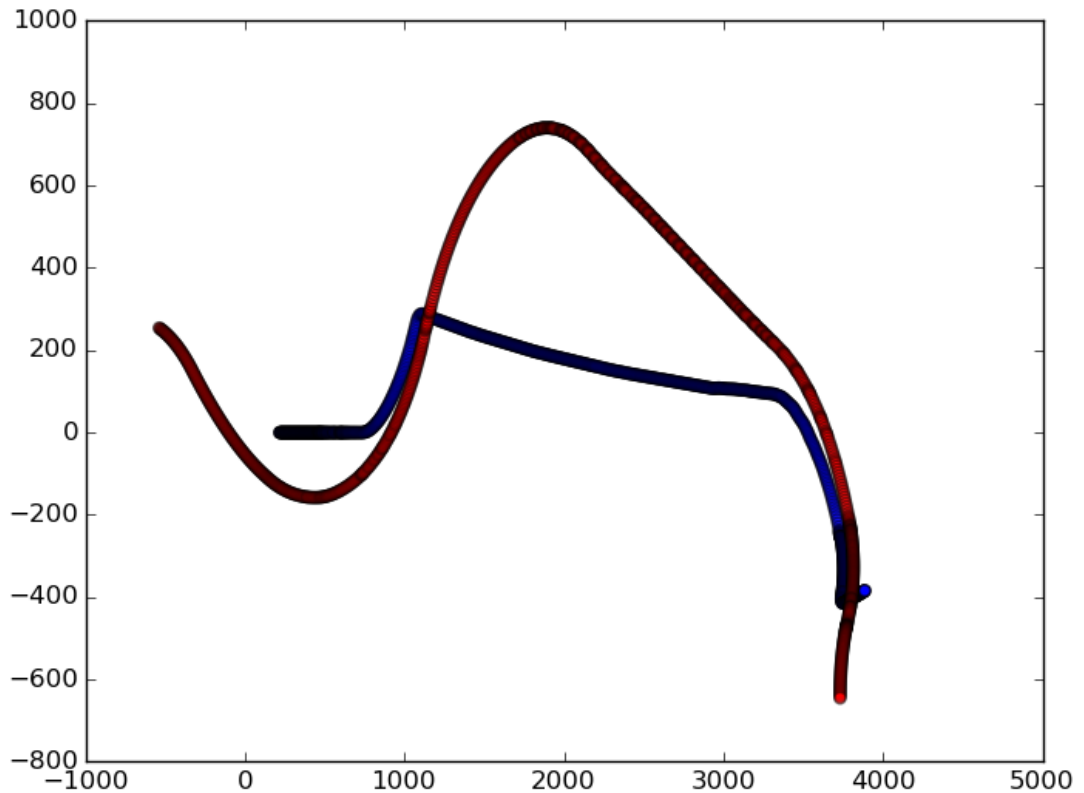


Figure 5.5.5: Confusion Matrix for Herd Test #2



**Figure 5.5.6:** Scatterplot of x/y positions of the own ship (blue) and the other ship (red), for Herd Test #2.

Test #2 shows an inversion of the results in test #1, with benign now the dominant behavior (50%), and herd a distant second (33%). This demonstrates once again the highly ambiguous nature of the data in the herd training files. Similar to the previous test cases, the recognized intent in this run is constantly switching between herd, benign and block, with very few periods of high consistency. Like test #1, this scenario was wrongly classified due to the training data for this HMM not being 100% herd intentions. This test case highlights the need for unambiguous data when training an HMM. Just like the previous example, the very final stretch of data in the run shows herd winning out as the dominant behavior.

## 5.6 Review of Results

The intent recognition system developed in this thesis performs well on three out of the four intent classifications. Ram is the intent that is most often classified correctly (in some cases 100% of the time). This is likely due to its simple, unambiguous nature lending itself well to intent classification. Benign had the second highest overall accuracy, averaging 85%. Benign activity was most often misclassified as a block by our intent recognition system, due to the two behaviors resembling one another in close proximity. Blocking came in third place according to overall accuracy, with an overall accuracy of 74%, and is most often confused with ram at close proximity, and benign when the two ships are relatively far away. Herding was the most difficult of the four intent classes to recognize, with an overall accuracy of 30%, which is only 5% better than a naive guessing approach.

The reason for the poor accuracy of the herding model turned out to be a problem with the input data not being 100% indicative of herding intent. A majority of the data generated for training and testing the herd model was virtually indistinguishable from benign behavior. As a result of this, herding behavior very often misclassified as benign behavior by our system. This sometimes led to benign becoming the dominant prediction in herd test cases. Unfortunately, this oversight was recognized too late in development to request new data.

# Chapter 6

## Conclusion

### 6.1 Closing Remarks

Intent recognition is a pivotal element of many aspects of computer science, including machine learning, artificial intelligence and robotics. An agent's ability to recognize and react to actions and intents is an integral aspect of many applications, some of which were discussed in Chapter 2 of this thesis. The ability for an intelligent agent to recognize intent allows for the development of high-level cognitive abilities, such as emulating behaviors learned from observations. It also allows for the development of reactive strategies, by incorporating knowledge of other agents' behaviors into a plan model.

In this thesis we presented a technique for recognizing the intents of ships in a highly realistic naval simulation. Using the simulator it was possible to produce large data sets describing realistic ship behaviors. This data was then used to generate a set of highly descriptive, discriminatory features which encapsulated information about a ship's current actions. These features were selected manually, based on what information would be informative and non-redundant to the intent recognition system. The features were then used for training a Hidden Markov Model based intent recognition system, capable of performing determination between different ship intents, on novel training data.

There were four intent classes which we sought to recognize: ramming, blocking, benign and herding. These behaviors were selected to provide realistic scenarios, which might be seen in a real world naval environment. Our approach worked well for three out of four of our intent classes. Ramming, blocking and benign behavior performed reasonably well on our testing data. The final behavior, herding, performed poorly due to an error in data generation, which was discovered too late to correct.

## **6.2 Improvements and Future Work**

### **6.2.1 Re-generate Herding Data**

The most needed change to our existing intent recognition framework is the substitution of the poor herding data we were provided, with data which is both highly descriptive of herding behavior as well as unambiguous from our benign samples. The herding data which we had access to was highly similar to our benign data; in some cases it was decided that more than 80% of a given herd file's content would be more correctly classified as benign behavior, thus hindering the accuracy of our herding model. An HMM's predictive power depends very much on the quality of the training data, so having access to new herd data would likely give our model a significant boost with regards to herd classification results.

### **6.2.2 Multi-agent Domains**

All the testing files produced for our research focused on a situation when there were exactly two ships on the water: the own ship, representing our point-of-view, and the other ship, whose intentions we were attempting to predict and recognize. It is likely, in a real world scenario, that there might be more than one other potentially adversarial ship on the water. One interested avenue of research would be the inclusion of multiple other ships in our test cases. For testing, this would require an HMM evaluation for every other ship within the environment, which would of course increase complexity linearly with respect to the number of other ships present, significantly slowing computation time in the presence of many ships. This complexity could be addressed by concepts mentioned in the next subsection.

### **6.2.3 Parallelization**

There exist many algorithms for the training and testing of HMMs in parallel, on a Graphical Processing Unit (GPU) [31]. In order to expand our research to include many additional ships in the simulation, a parallelization approach could help maintain real-time results. In addition to potentially improving computation speed in a multi-agent domain, there are also potential benefits to the single agent domain as well. A parallelization technique could improve performance in situations where the number of hidden states in an HMM is high, or in situations where the state space of the input vector is of high dimensionality.

### **6.2.4 High Level Intents**

Often times, in a multiple ship scenario ships will cooperate to perform complex naval maneuvers, such as forming a blockade. These maneuvers could potentially be recognized using techniques similar to those discussed in this thesis. One potential approach to solving this problem would be to recognize such high level, multi-ship intents as a collection of constituent single ship intents. For instance, it is likely that if a large number of single ships are attempting to block, there is likely a blockade being formed. This would introduce a hierarchical element to the HMM techniques discussed in this paper. Another idea would be to recognize these complex behaviors by training an HMM on features which describe the overall state configuration of other ships.

### **6.2.5 Additional Features**

There is no general method for finding a set of optimal features for a given classification problem. The manual extraction of descriptive features requires intuitive thought, in addition to testing and training to understand the effects that

new features have on classification performance. In an ideal situation, the features used for training and testing would be completely orthogonal to each other (no redundancy) and highly descriptive for separate classifications. A good feature is neither irrelevant nor redundant [33]. The features we used for our training and testing were manually determined to fit this criterion, however they are not necessarily the optimal set for the problem of intent recognition in our domain. One potential improvement is the search for new features to help separate classification distributions. This process can be aided by statistical methods, such as Principal Component Analysis or K-Means Clustering, used to analyze the distribution and covariance of the set of input features.

# Bibliography

- [1] Hovland, Geir E., Pavan Sikka, and Brenan J. McCarragher, "Skill acquisition from human demonstration using a hidden markov model", Proceedings on the International Conference on Robotics and Automation, vol. 3, 1996.
- [2] Pook, Polly K., and Dana H. Ballard. "Recognizing teleoperated manipulations", Proceedings on the International Conference on Robotics and Automation, 1993.
- [3] Scovanner, Paul, Saad Ali, and Mubarak Shah, "A 3-dimensional sift descriptor and its application to action recognition" Proceedings of the ACM 15th international conference on Multimedia, 2007.
- [4] Brand, Matthew, Nuria Oliver, and Alex Pentland, "Coupled hidden Markov models for complex action recognition", Proceedings of the IEEE computer Society Conference on Computer Vision and Pattern Recognition, 1997.
- [5] Kautz, Henry A, "A formal theory of plan recognition", Bell Laboratories, 1987.
- [6] Charniak, Eugene and Robert Prescott Goldman, "Probabilistic abduction for plan recognition", Brown University Department of Computer Science, 1991.
- [7] Demiris, Yiannis, and Bassam Khadhour. "Hierarchical attentive multiple models for execution and recognition of actions" Robotics and autonomous systems 54.5, pages 361-369, 2006.
- [8] Gallese, Vittorio, and Alvin Goldman. "Mirror neurons and the simulation theory of mind-reading" Trends in cognitive sciences 2.12, pages 493-501, 1998.
- [9] Demiris, Yiannis. "Prediction of intent in robotics and multi-agent systems" Cognitive processing 8.3, pages 151-158, 2007.
- [10] Wolpert, Daniel M., Kenji Doya, and Mitsuo Kawato. "A unifying computational framework for motor control and social interaction" Philosophical Transactions of the Royal Society of London B: Biological Sciences 358.1431, pages 593-602, 2003.
- [11] Geib, Christopher W. "Problems with intent recognition for elder care" Proceedings of the AAI-02 Workshop "Automation as Caregiver", 2002.
- [12] Han, Kwun, and Manuela Veloso. "Automated robot behavior recognition" Robotics Research International Symposium, vol. 9, 2000.
- [13] Richard Kelley, Christopher King, Alireza Tavakkoli, Mircea Nicolescu, Monica Nicolescu, George Bebis, "An Architecture for Understanding Intent Using a Novel Hidden Markov Formulation", International Journal of Humanoid Robotics - Special Issue on Cognitive Humanoid Robots, vol. 5, no. 2, pages 203-224, June 2008.
- [14] Richard Kelley, Alireza Tavakkoli, Christopher King, Monica Nicolescu, Mircea Nicolescu, George Bebis, "Understanding Human Intentions via Hidden Markov Models in Autonomous Mobile Robots", Proceedings of the ACM/IEEE International

Conference on Human-Robot Interaction, pages 367-374, Amsterdam, Netherlands, March 2008.

[15] Schmidt, Charles F., N. S. Sridharan, and John L. Goodson, "The plan recognition problem: An intersection of psychology and artificial intelligence", *Artificial Intelligence* 11.1-2, pages 45-83, 1978.

[16] Schank, Roger C., and Robert P. Abelson, "Scripts, plans, goals, and understanding: An inquiry into human knowledge structures", Psychology Press, 2013.

[17] Wilensky, Robert, "Why John married Mary: Understanding stories involving recurring goals", *Cognitive Science* 2.3, pages 235-266, 1978.

[18] Kautz, Henry A., and James F. Allen, "Generalized Plan Recognition", *AAAI-86 Proceedings*, 1986.

[19] Kautz, Henry A, "A formal theory of plan recognition", Bell Laboratories, 1987.

[20] McDermott, Drew, and Eugene Charniak, "Introduction to artificial intelligence", Reading: Addison-Wesley, 1985.

[21] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards "Interpretation as abduction" *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, 1988.

[22] E. Charniak and R. P. Goldman. "A Bayesian model of plan recognition", *Artificial Intelligence*, vol. 64, pages 53-79, November 1993.

[23] Vilain, Marc B. "Getting Serious About Parsing Plans: A Grammatical Analysis of Plan Recognition", *AAAI*, 1990.

[24] Pynadath, David V., and Michael P. Wellman. "Accounting for context in plan recognition, with application to traffic monitoring", *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann, 1995.

[25] Geib, Christopher W. "Delaying Commitment in Plan Recognition Using Combinatory Categorical Grammars", *IJCAI*, 2009.

[26] Geib, Christopher W., and Robert P. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars", *Artificial Intelligence* 173.11, pages 1101-1132, 2009.

[27] Prince, Simon, "Computer vision: models, learning, and inference", Cambridge University Press, 2012.

[28] Rabiner, Lawrence, "First Hand: The Hidden Markov Model", *IEEE global history network*, 2013.

[29] Baum, Leonard E, "An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes", *Inequalities*, vol. 3, pages 1-8, 1972.

- [30] Ellis, D, "What are delta features? How do you calculate them?", ICSI Speech FAQ, <http://www1.icsi.berkeley.edu/Speech/faq/deltas.html>, 2000.
- [31] Liu, Chuan. "cuHMM: a CUDA implementation of hidden Markov model training and classification", The Chronicle of Higher Education, 2009.
- [32] Taguchi, Y. H., and Yoshiki Murakami, "Principal component analysis based feature extraction approach to identify circulating microRNA biomarkers", PloS one, vol. 8, 2013.
- [33] Dash, Manoranjan, and Huan Liu, "Feature selection for classification", Intelligent data analysis 1.3, pages 131-156, 1997.
- [34] Duda, Richard O., Peter E. Hart, and David G. Stork, "Pattern classification", John Wiley & Sons, 2012.
- [35] Tanguay Jr, Donald O, "Hidden Markov models for gesture recognition", Massachusetts Institute of Technology, 1995.
- [36] Mann, Tobias P, "Numerically stable hidden Markov model implementation", An HMM scaling tutorial, pages 1-8, 2006.
- [37] Blunsom, Phil. "Hidden markov models", Lecture notes, pages 18-19, August 2004.
- [38] Jain LC, Halici U, Hayashi I, Lee SB, Tsutsui S, "Intelligent biometric techniques in fingerprint and face recognition", CRC press, vol. 10, 1999.
- [39] Peirce, Charles S, "Abduction and induction.", Philosophical writings of Peirce, vol. 11, 1955.