University of Nevada
Reno

Intent Recognition in Multi-Agent Domains

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

by

Daniel C. Bigelow

Dr. Monica Nicolescu, Thesis Advisor

May 2013

University of Nevada, Reno
Statewide • Worldwide

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

DANIEL CLAYTON BIGELOW

entitled

Intent Recognition In Multi-Agent Domains

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

MASTER OF SCIENCE

Dr. Monica Nicolescu, Advisor

Dr. Sushil Louis, Committee Member

Dr. Tomasz Kozubowski, Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

May, 2013

Abstract

Intent recognition is an extremely important aspect of social robotics. The ability to recognize and react to intentions is not only an integral part of social interaction, but is also useful in adversarial domains, as discussed in this thesis. It is especially important to be capable of performing intent recognition in multi-agent settings, both in terms of recognizing low-level intentions for individual agents, and of recognizing coordinated agents of multiple agents. This is because in real-world domains, it is rare to be performing tasks which require intent recognition in sparsely populated environments.

In this thesis, we present solutions to various aspects of the intent recognition problem. First, we introduce a framework for testing intent recognition systems in the multi-agent domain. This framework is modular in nature, making it easy to make changes to individual pieces of the system, and includes an open-source naval simulator, a low-level intent recognition module, a high-level intent recognition module, and a module for controlling the movements of the simulated agents.

This thesis also presents an extension of the work presented in [17, 18] to the multi-agent domain. We solve the problem of applying the hidden Markov formulation of the intent recognition problem to multiple agents while still operating in real-time by parallelizing various steps of the intent recognition algorithm, and relax the constraint that the topology of the HMMs must be designed by hand. In addition to this application of a low-level intent recognition system to the multi-agent domain, we also present a method for recognizing intentions which require cooperation between multiple agents. We do this by encoding high-level intentions in an activation network [1]. This approach addresses some of the drawbacks of traditional plan recognition techniques, including the difficulties presented by searching large plan libraries, and the difficulties in recognizing multiple plans which are being performed simultaneously.

Acknowledgments

Thank you to my advisor Dr. Monica Nicolescu for your generous support and patience. Thank you to my committee Dr. Sushil Louis and Dr. Tomasz Kozubowski for taking the time to see me through this final event of my degree. Thank you also to my colleagues Richard Kelley and Kyle Altemara for their help and input throughout my graduate career. Finally, thank you to my family, for their continued support throughout my life and school. I would not be here today without their help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In order to successfully interact with people and other robots in the real world, a robot must be able to watch one or more agents interact with the environment, recognize those actions, and use them to make inferences about future actions of the observed agents. Determining the action an agent is about to take, given a sequence of past actions, is known as the intent recognition (or sometimes the plan recognition) problem.

Intent recognition plays a crucial role in the potential success of many areas of robotics, ranging from human-robot interaction to competitive sports and military operations. One potentially useful application of intent recognition to human-robot interaction could be applied in assisted living facilities, where a robot would be required to predict the needs of the human it is helping [11]. Intent recognition has also been used in the context of prosthetics, where a powered prosthetic needs to recognize its user's intentions in order to perform the correct motions [27]. In fact, intent recognition is important for social robotics in general, as it is necessary for a robot to be able to predict human intentions in social situations in order for the robot to behave in a socially acceptable manner. For example, if a robot were to determine that two people walking towards each other intend to stop and converse, it could detour around them to avoid interrupting their conversation [17, 18]. Intent recognition plays a key role in competitive situations as well. In RoboCup [13], a key element of winning is a quick response to the other team's plays. Furthermore, the ability to actually *predict* the other team's future actions could provide a competitive edge. Potential applications of intent recognition extend to military operations as well. For instance, early warning systems could be devised to alert human operators to potentially hostile actions on the part of enemy forces. To this end, this thesis focused on applications of intent recognition in a naval domain.

There have been successful implementations of intent recognition algorithms, in fields ranging from pure mathematics to implementations on physical robots [16, 17, 18]. However, all of these methods are subject to strict constraints. In general, existing methods work only in very controlled environments. These methods also tend to scale poorly to scenes with a large number of agents present, or to require extensive pre-knowledge of other agents' plans. Relatively little work has been done in the area of multi-agent intent recognition, especially in high frame rate, real-time environments. Here, we present an initial solution to the problem of real-time intent recognition in crowded scenes, as well as preliminary work in the area of multi-agent plan recognition [3].

In this thesis, Chapter 2 gives a broad overview of related work that has been performed in the area of intent recognition. Chapter 3 shows a simulation system based on [21] which we have modified to support extensive testing of various types of intent recognition, and the overall infrastructure that has resulted from these modifications. Chapter 4 covers our implementation of low-level intent recognition in the multi-agent domain, as well as a quantitative analysis of our approach. Chapter 5 presents our preliminary work on using activation networks for high-level intent recognition. Finally, Chapter 6, we present some directions for future research.

Chapter 2

Background and Related Work

There is a body of existing work pretaining to intent recognition and some related fields. In this chapter, we shall explore some of the seminal works in the areas of action recognition, single-agent intent recognition, and multi-agent intent recognition, as well as some of the simulation systems which are more commonly used for testing research in multi-agent settings.

2.1 Action and Intent Recognition

2.1.1 Action Recognition

When examining the problem of intent recognition, we must distinguish between it and the problem of action recognition. Given a series of sensor inputs or world states, action recognition is the process of determining which action has just been *completed*, either by oneself, or by another agent. Solutions to this task are a key aspect learning by demonstration, as it is necessary for a robot to know which actions are being demonstrated in order for it to learn how to perform them. Intent recognition, on the other hand, is the problem of translating a series of sensor inputs into a prediction of the action that another agent is *about* to perform. This prediction can then be used to select an appropriate response before the other agent's action has been completed. This prediction of intent is useful in many settings, especially adversarial ones. While these problems have very different applications, they both require inference on temporal models of the world. Because of this, many current techniques for intent recognition draw inspiration from solutions to the action recognition problem.

There is a large body of work in the area of action recognition. In general, action recognition is used as part of a framework for robot training, as in [14, 23]. In [23], a robotic arm with high degrees of freedom is trained to manipulate primitives using examples given by a tele-operator. It uses k-nearest quantized pattern vectors to determine low-level actions and the corresponding controls, and uses sequences of these actions as inputs on a hidden Markov model in order to provide additional context and smoothing, which helps to reduce mis-classifications. In [14], a robot is trained in basic assembly skills by a human demonstrator. In this work, sensory data received by the robot during the demonstration phase are used to train the parameters of a hidden Markov model. When the training is finished, the states of the assembly task correspond to the states of the hidden Markov model, and the robot can use this model as a guide to determine which actions it needs to perform in order to complete the assembly task.

There has also been some work performed in the area of action recognition for its own sake (without a specific application in mind). References [26] and [5] provide examples of this. In [5], the concept of a coupled hidden Markov model is introduced. A traditional drawback to the use of hidden Markov models for temporal inference is the requirement of satisfying the Markov assumption, that is: the state at time t depends only on the state at time $t-1$. The coupled hidden Markov model approach introduced in [5] relaxes this constraint, and allows for the encoding of more complex information than traditional hidden Markov models. In [26], a 3D SIFT descriptor is introduced for classifying video streams or 3D imagery. This descriptor generates a signature for an entire segment of video (rather than a frame-by-frame signature like previous SIFT methods have used) and classifies the video using Support Vector Machines. This approach works quite well in comparison to previous SIFT methods, and has the benefit of capturing temporally dependent descriptors, which frame-by-frame methods tend to miss.

2.1.2 Intent Recognition

While the above methods work well for recognizing actions, and can be adapted to perform intent recognition, there is also a large body of work that focuses specifically on the problem of intent recognition. An early approach to this problem is introduced in Kautz's 1987 dissertation [16]. In [16], Kautz introduces a hierarchical library of event types, and plans which are composed of individual events. He then defines *minimum covering entailment*, which chooses plan models for which the library of observed events contains all of the events which make up the selected model, and for which the number of unrelated observations (events which occurred but are not part of the model) is minimized. This approach generalizes well to multiple intentions, but the structure of the event library has a significant impact on the performance of the classification algorithm, and the algorithm does not scale to the multi-agent case.

A more probabilistic approach to intent recognition was introduced in [7]. In this paper, Bayesian networks were used to represent intentions, with the highest-level node representing the overall intention and lower-level nodes representing sub-actions that contribute to that intention. From this, it is a simple matter of inference over the network to determine the probability of the high-level node being true, and thus achieve a level of belief about which intention is being carried out. Although this method performs well on the problems discussed in the paper, it can be difficult to represent complex intentions as nodes in a Bayesian network, and this method tends to scale poorly to large numbers of agents and/or recognizable intentions.

In [10], a biologically inspired approach to intent recognition is presented. In this paper, *forward models* are introduced which, given the state of a system and the control commands to be applied to it, produce the most likely next state of the system. In addition, *inverse models* are defined as models which, given the state of a system and the final goal of the system, produce the best set of controls to move towards that goal. These models can be used in a manner similar to *mirror neurons* [6]. The observing agent can take the current state of the system, and use the inverse models to compute the most likely set of controls to be used to achieve any given goal. At the next time step, the forward models can be used to determine which set of controls was applied, and thus which overall goal is most likely. This approach works quite well for a small number of models. However, the accuracy of the system decreases as the number of models (or recognizable intentions) increases.

Reference [9] provides a review of some of the approaches to the intent recognition problem which existed at the time of its publication. Specifically, it formalizes the ideas of *hierarchical representations* of intentions, and of *perspective taking*. With hierarchical representations of intent, lower-level actions feed into higher-level actions in a bottom-up approach. For example, [28] proposes a 3-level representation of intentions, where the lowest level is given by the dynamics of the system, the middle level is made up of sequences of elements, and the top level is comprised of symbolic representations of tasks. *Perspective taking* refers to the idea of putting oneself into someone else's shoes, or more formally, given the observed actions of another agent, the observer must determine what its own goal would be if it were performing the same actions.

In [11], a more general case of intent recognition problem is examined, and many individual tasks that must be performed in order to develop a fully working system for intent recognition are defined. This work also proposes a probabilistic model for plan recognition in elder care, and outlines how such a model could be implemented. However, at the time of publication of [11], the system had not been fully implemented or tested for accuracy.

In [13], hidden Markov models are used to represent and recognize strategic behaviors of robotic agents in a game of soccer. This method begins by computing state features from the sensor information of a robot, then using sequences of such features to train a set of hidden Markov models to recognize various actions. The trained Markov models are then used to determine how likely it is that a given set of state features was produced from each model. This approach can theoretically be generalized to recognizing any action whose performance can be discretized into a series of suitable observable variables for a hidden Markov model. However, in the scope of this paper, intent recognition is only performed in the context of a very controlled environment, where there are few agents and a very limited number of possible actions.

References [17, 18] also utilize the hidden Markov model based framework presented in [13]. In these papers, however, intent recognition is performed in far less structured environments. The authors are able to show that this method can work for recognizing the intentions of a human agent towards objects in a scene, or of two human agents towards each other or towards other objects, using only observable variables available from basic sensor information (provided by a laser range finder and color camera). However, while [17] and [18] show that the hidden Markov model based framework is generalizable to many application domains, they still suffer from some of the limitations of [13]. Although the recognition process is shown to work for less controlled environments, it is still only demonstrated in scenes where there are very few actors, and does not scale well to very crowded scenes, such as a robot may expect to encounter when acting in the real world.

2.2 Plan Recognition

So far, all of the methods for intent recognition (and action recognition as well) which have been discussed have focused on the single-agent intent recognition problem. Unfortunately, it is not a simple matter to extend these approaches to the multi-agent intent recognition domain. As stated in [9], it is not enough to simply recognize the intentions of each individual agent towards each other (although this is also a necessary step to solving the multi-agent intent recognition problem). To obtain a good prediction of multi-agent intent, it is also necessary to infer the joint intention, or the shared plan of the agents as a group.

The traditional approach to solving this problem is known as multi-agent plan recognition. In [3], Banerjee *et al.* present a formalization of this problem. Let $A = \{a_1, \ldots, a_n\}$ be a set of n agents. We are given a trace of activities over a period of time T, in a matrix $M = [m_{ij}]$, where m_{ij} is the action executed by agent a_j at time i. We are also given a vector of team plans L, where each plan $P = [p_{ij}] \in L$ is in the form of an $x \times y$ matrix with $1 \leq x \leq T$ and $1 \leq y \leq n$, and p_{ij} is the action expected from the jth team member at the ith step from the start of the plan. Given M, the plan recognition problem consists of determining the plan P which contains the sequence of actions p_{ij} that most closely matches M. Banerjee *et al.* also show that multi-agent plan recognition (using the above formalization) is NP-Complete, and introduce methods (such as branch and bound) for pruning the search space of the plan library. However, this method for plan recognition still does not scale well to large plan libraries, and does not handle goal abandonment or plan interleaving (the execution of multiple plans simultaneously).

The formalization presented in [3] is also used in [29]. In this paper, Zhuo *et al.* focus on the problem of partial observability. That is, they assume the actions executed at some point during the trace T are unknown, and attempt to marginalize on these unknown variables, while finding the plan for which all actions in the plan are also observed in the trace. This method is still not robust to goal abandonment or plan interleaving, and does not scale well to large plan libraries.

Works [15] and [2] move away from the formalization presented in [3]. In [15], it is assumed that the agents have all received similar training, and have full knowledge of the possible goals and actions to be executed. In this paper, plans are automatically mapped (as they are executed) to belief networks in an attempt to determine the most likely plan. While this approach works well, it has some major limitations. It operates under the assumption that all agents know the overall goal, and what each plan contributes to the overall goal. Thus, it is only necessary to determine which plan is currently being executed.

Finally, in [2], a template-based approach to multi-agent plan recognition is introduced. In this paper, a top-down approach is combined with a bottom-up approach. For the top down approach, observations about the state of the system are used to reason about possible global goals, and those global goals are decomposed into their component actions. This is then combined with a bottom-up approach, where the individual actions that are recognized are combined into plans. Any plans that are recognized can then be applied to the set of goals produced by the top-down approach to determine the most likely overall goal. While this approach seems theoretically sound and it addresses some of the problems that earlier methods do not (such as plan interleaving), no actual implementation or discussion of results was given in [2].

2.3 Simulation Environments

When performing intent recognition in a multi-agent system, it is important to be able to create consistent experiments, and to quickly and easily populate an environment with many agents. To this end, it is necessary to choose a good simulation environment in which to explore solutions to the intent recognition problem. Ideally, such an environment should be open-source, have a reasonable simulation of physics, be real-time, be 3D, and allow for the simulation of many agents at a time. Stage [12] and Gazebo [20] are a two open-source simulation systems that are widely used in the field of robotics. However, Stage [12] is not truly a 3D simulation engine, and does not incorporate a physics model. Gazebo [20], while being both 3D and having the capability of interfacing with a physics engine, does not allow for easy customization of the graphical display, making it difficult to display recognized intentions. PRACSYS [19] is a fully customizable, real-time, 3D simulation system with a built-in physics engine. However, this system is still under development, and the user interface is very difficult to use.

For our work, we chose to use Ecslent [21] as our simulation system. Ecslent is an open-source, 3D naval simulator with built-in physics, which allows waypoint-based control of the simulated ships, and has a fully customizable graphical interface, making it ideal for use with our intent recognition system.

Chapter 3

Infrastructure

In order to explore the problem of performing intent recognition in multi-agent settings, it is necessary to have an infrastructure in place that can easily provide scenarios with large numbers (at least 20) of agents, allow for replication of experiments, and which can perform in real-time. To this end, we developed a software infrastructure to support the testing of intent recognition systems in a simulated naval environment. We began with the simulator presented in [21] as a base, and built it into a fully functioning testbed for intent recognition systems. The remainder of this section shall outline the development process, and will discuss the architecture of the final system in some detail.

3.1 Development

We chose the naval simulator presented in [21] as a base for our project, as it supports the simulation of multiple ships, operates in real-time, and is deterministic (which allows for precise reproduction of experiments). In addition, this simulator is open-source, which made it easy for us to add functionality to support intent recognition. In our initial approach to developing an infrastructure, we integrated all of our systems directly into the simulation system. The intent recognition system was inserted into the existing system, which performed all of the physics calculations and graphics updates sequentially. Unfortunately, due to the high latency of the initial intent recognition system (which we shall discuss later in this work) and the added demand on the graphics system to display the calculated intentions, the frame rate of the simulator suffered, dropping to below .25 frames per second (fps) if more than 4 agents were present in the scene.

In order to improve the frame rate of the simulator, we re-implemented the display of the intentions by using less demanding graphics, although this required some loss of information in the display space. This improved the frame rate to around 5 fps for up to 8 simulated agents, but still didn't help as much as we had hoped. We then decided to modularize the system, so the simulator could either be run as a stand-alone system, or in conjunction with the intent recognition. Thus, the intent recognition system would not be dependent upon any specific simulation system. To this end, we separated the intent recognition into a separate, stand-alone system, as seen in Figure 3.1, and integrated ROS [24] nodes into both the simulator and the intent recognition system to allow for communication. Since ROS provides a publish/subscribe communication architecture (which, by nature, allows asynchronous communication), this significantly improved the performance of the simulator, as the simulator no longer had to wait for information from the intent recognition system.

Although modularizing the system improved the frame rate of the simulator, it still did not help with the underlying problem: the intent recognition system was too slow, and scaled very poorly to large numbers of agents. Because communication between the simulator and the intent recognition was asynchronous, the intent recognition system was dropping much of the data passed by the simulator, leading to classification errors. We remedied this problem by switching from the serial implementation of intent recognition [17, 18] to a parallel implementation, using CUDA (and the *cuhmm* library provided by [8]). This component will be discussed more in depth in Chapter 4. This parallelization led to a significant improvement in the throughput of the intent recognition system, and allowed us to add a visibility module to the system, which takes the global state space of the simulator and provides a locally visible state space from the perspective of each agent in the scene, for a more

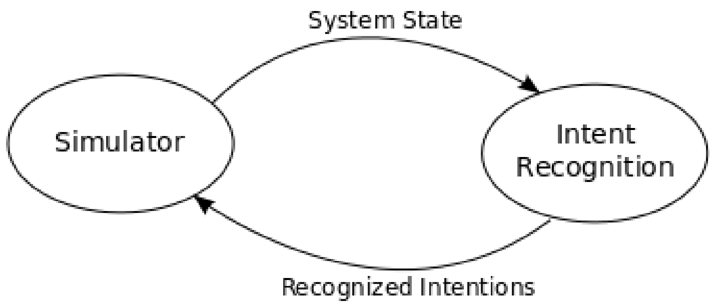

Figure 3.1: The simulator and intent recognition system architecture after the modularization process.

realistic simulated environment.

Finally, we added additional modules to the system to improve its overall functionality. First, we added a module to support external control of agents in the simulation. This allowed us to design controllers which can react to recognized intentions without having to integrate the controllers into the simulator itself, as this would have compromised the performance of the simulator. In addition, we created a second, higher-level intent recognition module, which takes the state of the simulator and the current recognized low-level intentions as inputs, and returns any recognized high-level intentions to the simulator. Figure 3.2 provides a diagram of the final system.

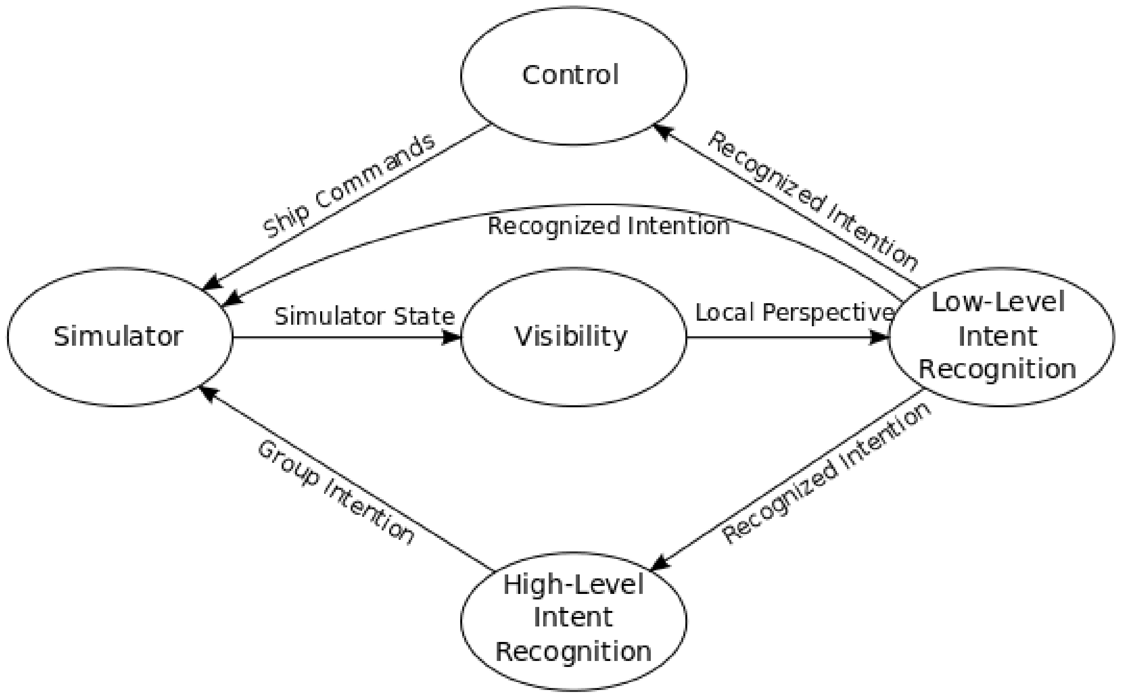

Figure 3.2: The final simulator and intent recognition system architecture.

3.2 System Description

Having outlined the process that led to the creation of our final system for simulation and intent recognition, we shall now examine each module of the system in more depth, including each module's dependencies, what each module provides, and a summary of the actions performed by each module.

3.2.1 Simulator

The simulator module of our system can act as a stand-alone, sandbox-style naval simulator, with the capabilities to create scenarios through either the GUI or through python code, and the ability to save and load created scenarios. It can also interface with an intent recognition system through ROS [24], and can receive and display information about recognized intentions, as well as displaying alerts when non-navy ships get too close to a naval vessel, and whenever group behaviors begin or end. The simulator can also receive navigation commands for actors in the current scene over the network (again using ROS).

In order to perform the actions above, this system requires the following information from external modules:

- Commands for agents.
- List of intentions (optional).
- High-level intentions (optional).

The information that this module publishes and makes available to external modules is:

- Current state of each agent.
- Position of land masses.

3.2.2 Visibility Module

Although this module is not strictly necessary for the intent recognition system to work, it does help to add realism to the simulated scenarios. Given the world state generated by the simulation module, a straightforward application of the low-level intent recognition module will result in intentions as recognized by an agent with global knowledge. However, in the real world, it is very rare for agents to have such knowledge. Thus, the visibility module takes the current state of the simulator and transforms it to the local perspective of each agent in the scene by calculating which agents are visible to each other.

In order to perform the actions above, this system requires the following information from external modules:

- Current state of each agent.
- Position of land masses.

The information that this module publishes and makes available to external modules is:

- Current state of each agent as seen by each other agent.

3.2.3 Intent Recognition Module

The intent recognition module is based on the intent recognition system presented in [17, 18]. In this module, a hidden Markov model is trained for each low-level intention that needs to be recognized. The data from the visibility module is then used to calculate the log-likelihood of each intention, given the current and previous states of the system. The intention with the maximum likelihood is then selected and returned to the simulator. The structure of this module can be seen in Figure 3.3.

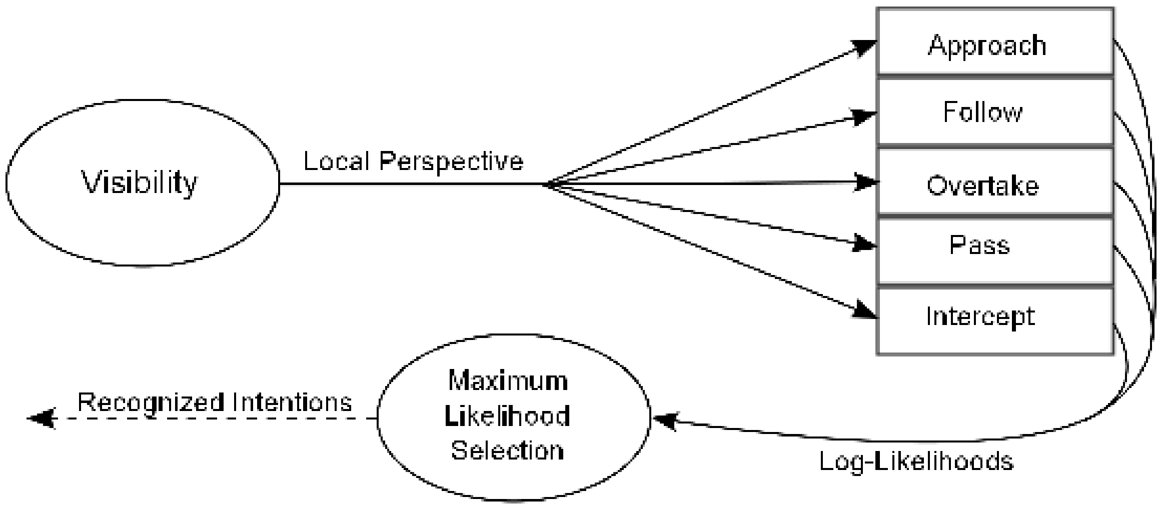

Figure 3.3: A close-up look at the structure of the intent recognition module

In order to perform the actions above, this system requires the following information from external modules:

- Current state of agents for which intentions are being computed.

The information that this module publishes and makes available to external modules is:

- Array containing most likely intention for each pair of agents.
- Log-likelihoods of each recognizable intention for each pair of agents.

3.2.4 Control Module

This module is not necessary for the operation of the system as a whole, but can be useful in implementing controllers that take into account current intentions. This module receives the current set of recognized intentions, the current state of each ship, and topological layout of the world, and is able to issue waypoint-based navigation commands to agents acting in the scene.

In order to perform the actions above, this system requires the following information from external modules:

- List of agents.
- Location of land masses.
- Current recognized intentions (optional).

The information that this module publishes and makes available to external modules is:

- Navigation commands for agents in the current scene.

3.2.5 High-level Intent Recognition Module

While it is possible to calculate low-level intentions (such as approach, pass, or follow) using the hidden Markov Model approach given in [17, 18], representing higher-level plans is more difficult, and often requires knowledge of the low-level intentions as well. Because of this, we also implemented a high-level plan recognition module which uses both context information from the simulator and the current recognized low-level intentions. This module recognizes high-level plans, and also determines an overall threat level (for naval scenarios).

In order to perform the actions above, this system requires the following information from external modules:

- Current state of agents.
- Current recognized intentions.

The information that this module publishes and makes available to external modules is:

- High-level plan recognition.
- Overall threat level.

Chapter 4

Low-level Intent Recognition

Markov models have been shown to work well when modeling processes that evolve over time. Consider a time-homogeneous Markov model representing a system, consisting of a set of N discrete states $\{s_i\}$. At any time, the system can be in any one of these states, and it transitions among the states with probabilities a_{ij}, where a_{ij} is the probability of being in state s_j at time $t+1$, given that the system was in state s_i at time t. We can see how this might be useful when trying to represent an intention such as *overtaking* another agent. In this case, the states might be {match heading, approach, pass to side, pull away}, with {pull away} being a terminal state (a state s_i such that $a_{ii} = 1$). However, in practice, it is often difficult to observe the state of a Markov model directly. Thus, we use an extension of Markov models known as hidden Markov models (HMMs) in order to perform intent recognition. With this extension, the system is modeled as a set of N hidden states $\{s_i\}$ with transition probabilities $\{a_{ij}\}$, as before. However, in this formulation the state of the system at time t is not directly observable. Instead, there is a set of observable variables $\{v_i\}$, which depend on the hidden states. For each state s_i, the probability of observing any given variable v_j is given by $b_{ij} = P(v_j|s_i)$.

The HMM formulation for this problem works well, as it allows us to compute observable variables based on globally available data such as an agent's heading, its distance to a target, etc, rather than attempting to directly ascertain which sub-action an agent is performing as part of an overall intention. In addition, we can use the HMM framework to represent intentions in a general manner by using context-free observable variables (such as whether or not the distance between two agents is increasing, decreasing, or remaining constant) rather than context-dependent variables (such as an agent's absolute speed or position). We base our method of using HMMs to infer intent on the methods presented in [17, 18], which require the implementation of two different processes: activity modeling and pattern classification. We then extend these methods to work efficiently in the multi-agent domain.

4.1 Activity Modeling

In order to recognize intentions using HMMs, we must trained models which correspond to each type of intention that we want to be able to recognize. We first generated scenarios showing each type of intent using the naval simulator described in Chapter 3. We generated 20 of each type of scenario, each with randomized starting configurations. The system state was then logged at each time step, for each scene, in order to generate data for training hidden Markov models. From these logs, we were able to compute observable variables from each frame generated by the simulator. The observable variables used for our models were distance to target and angle to target, each of which could be increasing, decreasing, or constant, as well as orientation of actor towards target and vice versa (facing or not facing), and difference in headings (facing the same direction or not). The alphabet of observable variables used consists of all possible combinations of these variables. So, for example, a visible symbol might be: (*distance:* increasing, *angle to target:* decreasing, *orientation:* facing target, *target's orientation:* not facing agent, *difference in headings*: none).

Once the observable variables have been calculated for each frame of each generated scene, the transition probabilities a_{ij} and emission probabilities b_{ij} for the HMMs that represent each intention can be trained using the Baum-Welch [4] algorithm. This algorithm also requires that the topology of the HMM be defined beforehand (i.e., number of hidden states). The methods for intent recognition presented in [17, 18] suggest that the hidden states be determined based on a human-understandable model of the activity being recognized, as in the example of overtaking provided above. However, empirical results suggest that this is not necessarily the best approach, as often the hidden states, which might make sense to a human, are not actually easy to discriminate mathematically. For this reason, we used cross-validation to determine the number of hidden states which resulted in the best performance for each model, and used these models when performing our classification.

4.2 Pattern Classificiation

Once HMMs have been trained for each intention we want to recognize, the actual recognition becomes a problem of pattern classification. At runtime, we must calculate the observable variables for each agent in a scene with respect to each other agent, and determine the HMM which is most likely to have generated a given sequence of observables. This can be done using the forward algorithm [25], which, given an HMM and a sequence of observable variables, returns the log-likelihood of the given HMM generating that particular sequence of variables. If we do this for each HMM that we have trained, we can simply choose the HMM with the highest probability of generating our observed sequence, and classify the intention accordingly.

This approach for intent recognition is shown to work quite well by the work presented in [17, 18]. However, it is subject to some constraining assumptions. First, while the algorithm will work as-is for scenarios involving multiple agents, the forward algorithm is computationally complex and the intent recognition process can not scale to scenarios involving many agents while still performing in real-time. Second, the HMM-based approach to intent recognition assumes that agents' intentions towards one another are statistically independent. (For example, given a scene involving three agents, the intention of agent 1 towards agent 2 is independent of agent 2's intention towards agent 3, and both are independent of agent 2's intention towards agent 1.) While this simplifying assumption is not a problem for low-level intentions such as passing, overtaking, or approaching, it presents difficulties when attempting to model intentions which involve coordinated efforts between agents. The scaling problem will be addressed in the next section, and a method for handling more complicated intentions will be presented in Chapter 5.

Another challenge is that the observations come as a continuous stream of measurements, spanning the lifetime of the simulated scenario. However, when using the forward algorithm to compute the liklihood of a sequence, the probability of that sequence drops to zero as the size of the sequence increases. We addressed this problem by only performing inference over the most recent chunk of k observations, as in [17, 18]. In our results, we used a chunk size of $k = 30$.

4.3 Extension to Multiple Agents

While the methods described above theoretically work for scenes containing any number of agents, they are quite slow in practice. Early experiments show that these methods tend to stop performing in real-time when presented with any more than 5 agents in a scene. However, there is a simple solution to this problem, which takes advantage of the assumption that each intention is independent of each other intention. We can simply parallelize each step of our algorithm, rather than running it sequentially. We calculate the observable variables independently and in parallel for each pair of agents in the scene, using a simple CUDA kernel. Once each sequence of observable variables has been calculated, we use the parallel implementation of the forward algorithm presented in [8] to calculate the log-likelihood for each intention, for each pair of agents simultaneously. Once this has been done, another simple CUDA kernel is used to pick the most likely intention for each pair of agents (or no intention, if all likelihoods are below a certain threshold). This parallelization significantly increases the speed of the intent recognition process.

4.4 Experimental Results

We have presented an extension of the intent recognition techniques developed by [17, 18] to the multi-agent domain. Thus, in order to analyze the performance of our system, we must not only verify the baseline accuracy of the HMM-based approach, but also the performance of our parallelized algorithm in terms of run-time, and the accuracy of the intent recognition process in more complex scenarios.

4.5 Baseline Accuracy

To test the baseline accuracy of the HMM-based approach to low-level intent recognitions, we trained models for 5 different intentions: approach, pass, overtake, follow, and intercept. We then generated 200 two-agent scenarios using the simulation system discussed in Chapter 3, resulting in 40 test scenarios for each of the trained intentions. All of our statistics represent the average performance of the intent recognition system over the 40 relevant scenarios. For a quantitative analysis of the intent recognition system, we used three standard measures for evaluating HMMs [22]:

- *Accuracy rate:* The proportion of test scenarios for which the final recognized intention was correct.

- *Average early detection:* $\frac{1}{N}\sum_{i=1}^{N}\frac{t_i^*}{T_i}$, where N is the number of test scenarios, T_i is the total runtime of test scenario i, and t_i^* is the earliest time at which the correct intention was recognized consistently until the end of scenario i.

- *Average correct duration:* $\frac{1}{N}\sum_{i=1}^{N}\frac{C_i}{T_i}$, where C_i is the total time during which the correct intention was recognized for scenario i.

For reliable intent recognition, we want *accuracy rate* and *average correct duration* to be close to 100%, and *average early detection* to be close to 0%. The results of our experiments are shown in Table 4.1. As can be seen, the intent recognition system performs comparably with that presented in [17, 18] in terms of both accuracy rate and correct duration, with 100% accuracy rate, and over 88% correct duration for all but the "overtake" intention. It also performs well in terms of early detection for the approach, intercept, and follow behaviors, recognizing them consistently within the first 12% of the completion of the action. These results are consistent with the current state of the art for single-agent intent recognition methods [17, 18].

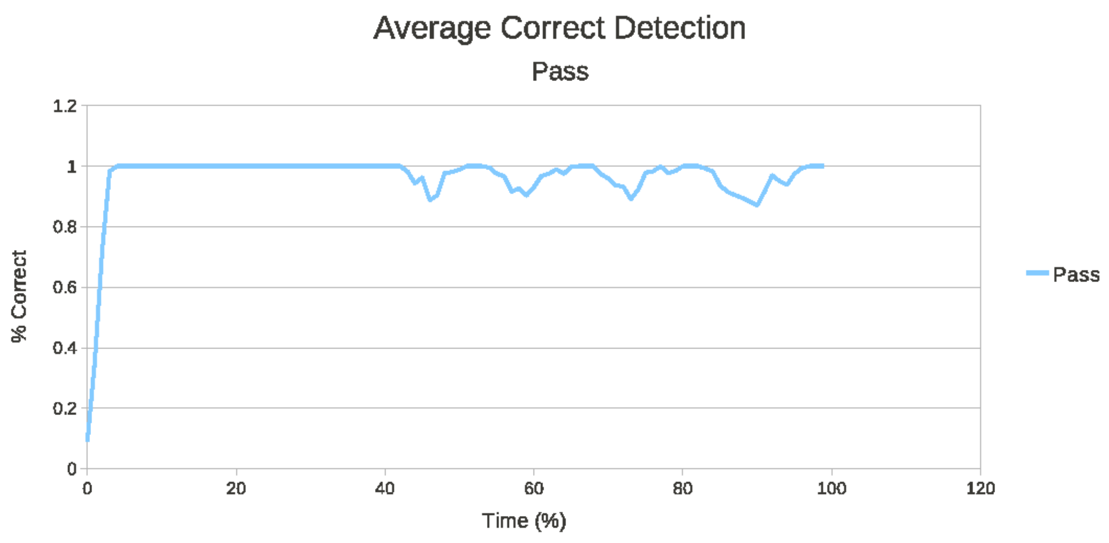

Figure 4.1: The average correct detection of pass over time.

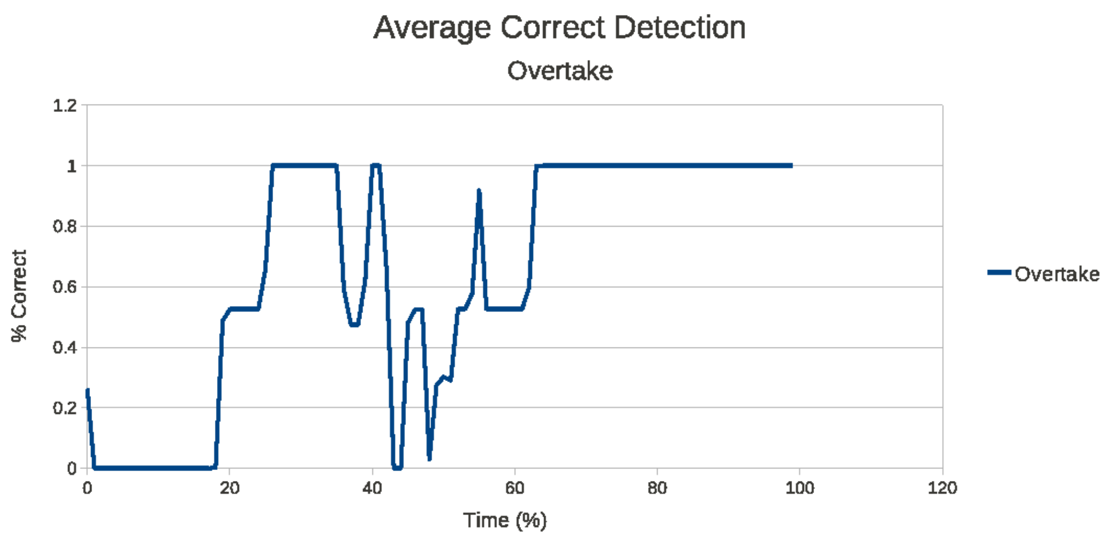

Figure 4.2: The average correct detection of overtake over time.

Figure 4.2 and Figure 4.1 provide an explanation for the poor performance of the pass and overtake behaviors. These figures shows the percent accuracy of each intention over the duration of the run. For instance, if 20 out of the 40 runs correctly recognized “pass” 50% of the way through a scenario, then the value of the graph in Figure 4.1 at $t = 50$ will be .5. From this analysis, we can see that both pass

Table 4.1: Quantitative evaluation of low-level intent recognition module

Scenario	Accuracy Rate (%)	Avg. Early Detection (%)	Avg Correct Duration (%)
Approach	100	8.95	90.9
Pass	100	68.0	96.5
Overtake	100	56.8	64.6
Follow	100	1.92	99.3
Intercept	100	11.3	88.8

and overtake are correctly recognized for the majority of the duration of each scene (as borne out in Table 4.1), but consistently fail to be recognized when the agents have drawn abreast of each other. This is likely due to a lack of distinguishing evidence variables at this time. Given the evidence variables discussed above, the only difference between pass and overtake at this point would be "change in angle from target agent to acting agent," which is likely not enough to result in a distinct classification.

4.6 Parallelization

To evaluate the effectiveness of parallelizing the intent recognition process, we implemented both serial and parallel versions of the intent recognition algorithm and ran them on scenes containing varying numbers of agents (provided by the simulation system discussed in Chapter 3). We then recorded the average frame rate over each scene (with one frame defined as a single iteration of the intent recognition algorithm, from symbol generation to selection of most likely intent), with the results shown in Figure 4.3. We can see that while the performance of the serial implementation of the intent recognition process quickly drops below an acceptable frame rate for real-time systems, the parallel implementation maintains a speed of about 40 frames per second, which is definitely adequate for performing in real-time. The intent recognition problem as presented in this thesis has a computational complexity of $O(n^3)$ (intentions must be calculated for each pair of agents, from the perspective of each agent). Thus, we can expect that the intent recognition system will continue to perform in the neighborhood of 40 fps as long as $n < \sqrt[3]{m}$, where n is the number of agents and

m is the maximum number of threads provided by the GPU. On our system (which uses the Tesla C2050), this means that we should be able to continue performing intent recognition in real-time as long as $n < 30,000$.

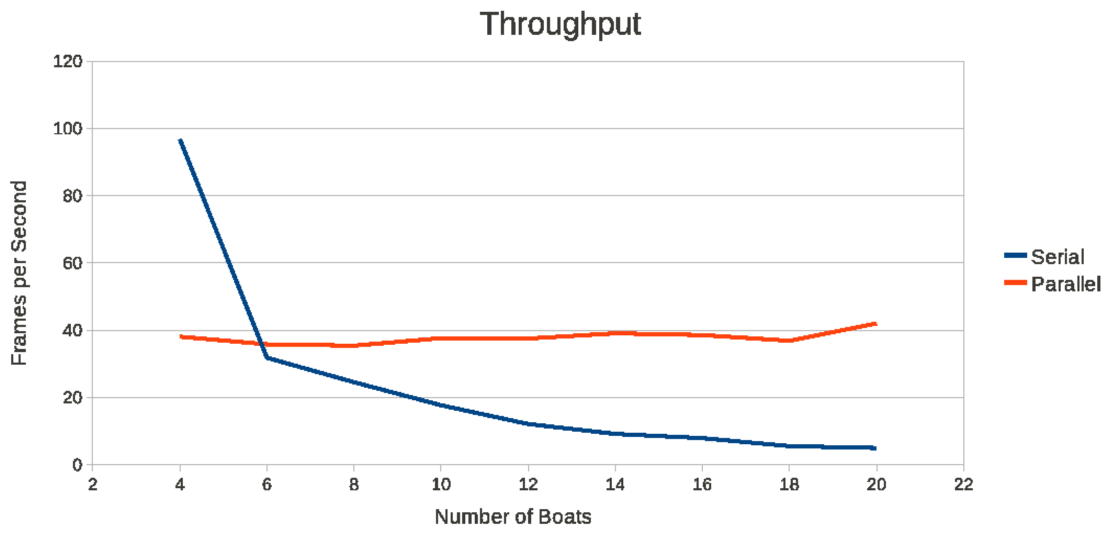

Figure 4.3: Performance of serial implementation of intent recognition vs parallel implementation.

4.7 Complex Scenarios

Since the main factor motivating the extension of [17, 18] to the multi-agent domain is the desire to be able to recognize intentions in complex scenarios involving multiple agents, it is necessary to evaluate the performance of our system on some scenes involving more than two agents. To this end, we created 6 different scenarios in our simulator in which naval vessels needed to recognize potentially hostile intentions (approach and intercept) as enemy ships maneuvered to attack.

In the *Straits of Hormuz* scenario, a convoy of naval vessels is attempting to traverse the straits. As they do so, a pair of other ships pass close by the convoy, creating a distraction. Shortly after this, more ships break free of a group of trawlers, and begin a suicide run towards the convoy in an attempt to damage it. The *San Diego* scenario is constructed similarly. Here, a group of naval vessels is attempting to exit the San Diego harbor. As they travel towards the harbor mouth, a ship that had been behaving like a fishing boat comes about and begins a run towards the navy vessels.

In *hide,* the naval vessels are traveling through a channel, while passing some container ships. As this happens, a small boat accelerates to a position behind one of the container ships and hides there until it is abreast of the navy vessels. At this point, it breaks from hiding and attacks the navy vessels.

Blockade and *hammer and anvil* are examples of some scenarios in which more complex intentions (in which agents must cooperate to perform a task) may occur. In *blockade*, a naval vessel is attempting to pass through a channel when some other ships emerge from hiding behind nearby islands and intercept it, forming a blockade. *Hammer and anvil* begins similarly, but once the channel is blocked by the blockading ships, an additional pair of ships approaches from behind the naval vessel in order to attack and cut off escape. We discuss the recognition of such complex intentions in depth in Chapter 5. In order for these techniques to work, we must also be able to accurately recognize the low-level intentions (intercept and approach) which make up the overall attacks.

In performing a quantitative analysis of the more complex scenarios, we first define *key intentions* as those intentions that make up actions which are threatening to the naval vessels in the scene. For instance, in the *hide* scenario, the container ships may have the intention of “passing” the naval vessels, but this would not be a key intention. However, the aggressive ship must “overtake” a container ship in order to hide behind it, and must “approach” the navy vessels in order to attack them, and both of these would be considered key intentions. For the purposes of determining the performance of the intent recognition in the complex scenes, we will focus on the average early detection for key intentions in each scene, and the accuracy rate for those key intentions as well.

The accuracy rate for our system is 100% for key intentions in the complex scenarios – in each of the 5 scenarios all of the key intentions were correctly identified.

Table 4.2: Intent recognition in complex scenarios

Scenario	Early Detection (%)
Straits of Hormuz	1.50
San Diego	2.31
Hide	3.03
Blockade	0.0
Hammer and Anvil	5.04

In addition, it can be seen in Table 4.2 that the early detection rate for the key intentions is below 13%. In every case, the key intentions were recognized almost as soon as they began.

Chapter 5

High-Level Intent Recognition

As we saw in the previous chapter, although hidden Markov models work well for detecting low-level intentions, it is difficult to use them for the detection of higher-level, cooperative intentions. This is particularly true in our case, as the parallelization used to allow the HMM algorithms to run in real-time enforces the assumption of independence between intentions. However, this independence is not always a reasonable thing to assume. For example, an intention such as "form a blockade" would require multiple agents to simultaneously perform an "intercept" action. Thus, it is necessary to find some way of modeling and detecting plans and intentions which require cooperation between agents. One potential approach to this is to use the intentions generated by the low-level intent recognition module as evidence variables for new Hidden Markov Models, with the new HMMs trained to recognize more complicated intentions. However this (and other model-based approaches) is difficult to implement in practice, as it can be impractical to generate enough training data to obtain a good model of a complicated action. Because of this, other approaches to high-level intent recognition must be explored.

5.1 Early Approaches

One approach to recognizing higher-level intentions is to look for abnormal (or in our case, threatening) behaviors, and use the recognition of those to make claims about the overall intentions which are currently being carried out. To this end, we introduced modules capable of recognizing *hiding* and *group formation*, as well as modules which trigger a warning whenever a ship in the simulation broke the maritime laws regarding the approach of a naval vessel (no ship is allowed to approach within 100 meters of a naval vessel, and any ship within 500 meters must be traveling at a low enough speed to avoid creating a wake). Group formation was recognized using a simple application of disjoint sets. At each timestep of the simulation, each ship belongs to its own disjoint set. Next, for any two ships for which the "follow" intention is recognized, the corresponding disjoint sets are joined. If any ships belong to a disjoint set at time t which did not belong to that set at time $t-1$, then a group is said to be forming.

To recognize *hide*, we started with our projection to a local perspective (discussed in Section 3.2.2). From this, we gain information about which ships are visible to each other. For each ship, we check to see if each other ship is visible. If not, we project an estimated location based on the last known speed and heading of the obscured ship. If at any point the projected location of the ship is visible, but the ship itself is not, we say that the ship is hiding.

While the recognition of these abnormal behaviors worked well, they did not end up providing any useful information about specific plans which might be happening. The recognition of these behaviors was useful for providing a warning that some sort of (threatening) high-level plan was taking place. In order to recognize specific plans, however, a different approach was needed.

5.2 Activation Networks

As noted in Chapter 2, there has been some work done already on multi-agent plan recognition. The general approach to this problem, as defined by [3], is to maintain a library of "plans" where each plan is represented as a sequence of actions which must be completed in order to achieve some overall goal. In our "create a blockade" example, the plan might be "at least three agents must perform an intercept". In addition to this library, a list of completed actions is also maintained. Each time an action is detected, it is added to the list of completed actions. Then, the plan library is searched for plans which require sequences of actions that are subsets of the current list.

While plan recognition techniques are also extremely applicable to intent recognition (substituting “intention” for “action” in the process described above), there are some major drawbacks to this family of approaches. First, as more plans get added to the plan library, it becomes more expensive to look up plans and compare them to the current list of actions. In addition, it is possible for multiple plans to be executed simultaneously by different groups of agents, which can result in the list of recognized actions/intentions not actually matching any overall plan in the plan library. Thus, it is necessary to consider all possible combinations of actions for all possible agents, which quickly becomes prohibitive in cost. Finally, it is usually necessary for the plan library to be created by hand, which introduces the possibility of human error (i.e., the person creating the plan forgets a necessary action, or includes actions that are not necessary), and generally requires plans to be tailored to individual problem domains (as there is no way to learn new plans on the fly). To handle these problems, we propose a new approach to the problem of multi-agent intent recognition, using the concept of spreading activation [1] to encode and recognize intentions in a hierarchical manner.

5.2.1 Overview

An activation network can be represented by a directed graph with N vertices, or nodes, and k edges, or links, connecting the nodes. Each node i will have an activation value A_i. A $link_{ij}$ is a directed edge from node i to node j, and will have a weight W_{ij}. In addition to weights and activation values, there will also be a firing threshold F and a decay factor D associated with the network.

The typical procedure for spreading activation is as follows. First, the graph is initialized by setting all $A_i = 0; i \in [1, .., N]$. Next, an activation level is set for one or more origin nodes. Next, for each unfired node $[i]$ in the graph having an activation level $A_i > F$, for each $link_{ij}$ connecting such an unfired node to some node $[j]$, adjust $A_j = max((A_j + (A_i * W_{ij})) * D, 0)$. A node may only fire once. Repeat until some halting criterion is met (i.e. there are no nodes left to fire, or activation has reached a specific node). We will now show that with a few adjustments to the basic algorithm, activation networks can be used to both encode and recognize complex intentions.

5.2.2 Intent Recognition

Complex intentions can be encoded in the topology of an activation network. In order to do this, origin nodes are created which correspond to the low-level intentions returned by the module discussed in Chapter 4. Next, nodes are created which correspond to higher-level intentions. For our "create a blockade" example, we would have origin nodes corresponding to the low-level intention "intercept recognizing agent," which would all link to a node representing "create a blockade". The topology for such a network can be seen in Figure 5.1. It is also relatively simple to encode multiple intentions in a single network. For example, if we wanted to recognize both "form a blockade" and "perform a hammer and anvil attack", we can simply modify the network for "form a blockade" by adding origin nodes corresponding to the low-level intention "approach recognizing agent", and adding a node representing "hammer and anvil attack". We can then simply add a link from the new origin nodes to "hammer and anvil attack", as well as a link from "blockade" to "hammer and anvil", as seen in Figure 5.2.

Now that we can encode intentions in the topology of an activation network, we must also be able to recognize said intentions when they occur in real-time. The standard spreading activation algorithm is not suitable for this, as it runs only a single time and does not allow nodes to fire more than once. However, we can adjust the algorithm by running in a continuous loop, where at each step activation is set on the origin nodes (based on the low-level intentions currently being recognized at this time-step) and propagated up the graph in the standard fashion. By doing this, activation will accumulate in the various intention nodes until the incoming activation is balanced by the decay factor, and the network stabilizes. Thus, detecting an intention can be done by simply thresholding on the activation level of the node corresponding to said intention.

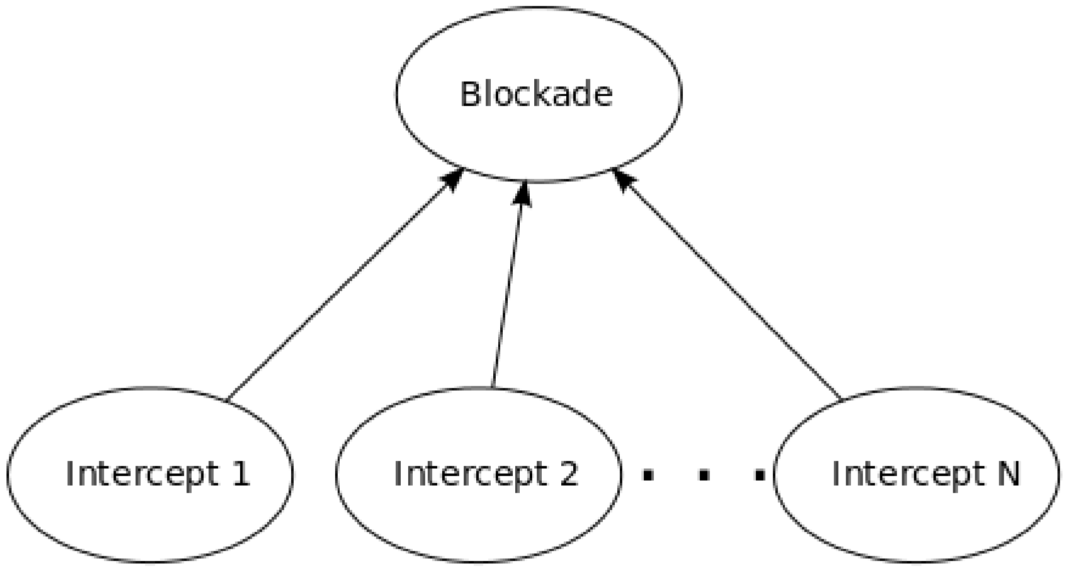

Figure 5.1: The graph structure for an activation network designed to recognize a blockade behavior.

5.2.3 Threat Level

Encoding intentions in activation networks addresses the problem of maintaining a library of plans to recognize, as well as the problem of the time required to perform an exhaustive search of said plan library. Using a topology-based approach to encoding intentions, it is often possible to add new intentions without a dramatic increase in the size of the network (as seen in our examples above). In addition, the use of spreading activation means that we do not have to explicitly search a library to see if a plan matches the sequence of recognized low-level intentions. Instead, activation is propagated automatically as new intentions are recognized and cause the corresponding origin nodes in our network to fire. Thus, we must simply monitor

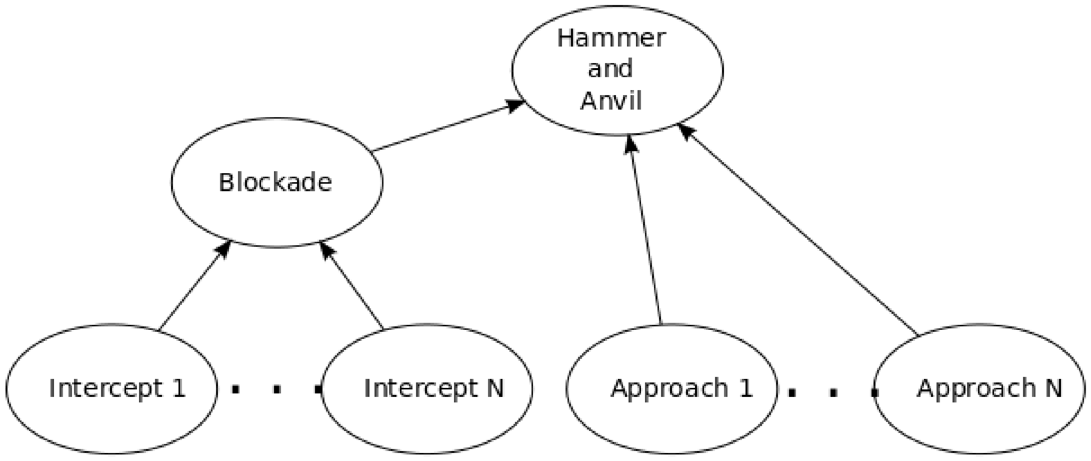

Figure 5.2: The graph structure for an activation network designed to recognize a hammer and anvil behavior.

the activation levels of the nodes corresponding to high-level intentions to determine if such an intention is likely. This also solves the problem of simultaneous execution of plans, as the structure of the network will encode all possible recognizable intentions, and activation will spread appropriately from any firing origin nodes. However, it is still necessary to encode each high-level intention by hand, and specify the topology of the network accordingly.

While it is unlikely that a method for automatically generating plans for high-level intentions will be found, it is possible to use our spreading activation approach to determine a “general threat level”. While this may not be as useful as recognizing a specific intention, it is possible to generate the network for recognizing this threat level with minimal human input. It is simply necessary to define a subset of the low-level intentions which can be deemed “hostile” (such as approach, intercept, etc). Once this is done, origin nodes can be generated for each possible hostile intention towards the recognizing agent and links can be generated connecting these to a “threat level” node. In addition to these low-level behaviors, the abnormal behaviors discussed in Section 5.1 can be used as inputs to this network. The degree of threat will then be proportional to the activation level of the “threat level” node.

Table 5.1: High-Level Intent Recognition

Scenario	Early Detection (%)	Accuracy Rate (%)
Blockade 1	3.12	100
Blockade 2	3.10	100
Hammer and Anvil	5.23	100

5.3 Experimental Results

5.3.1 Activation Networks

In order to quantitatively evaluate our activation network based approach to high level intent recognition, we designed two scenarios, *blockade* and *hammer and anvil*, in which cooperative intentions were taking place. A description of each of these scenarios can be found in Section 4.7. In the *blockade* scenario, we want our system to be able to recognize the blockade behavior before it is actually completed, and continue to recognize it as long as it is happening. Similarly, for the *hammer and anvil* scenario, our system must first recognize the blockade which happens, and then recognize the hammer and anvil attack as the other ships begin to approach from behind. Thus, we shall use the early detection and the accuracy rate metrics to evaluate our results.

Table 5.1 shows the results for each of the three intentions we wanted to recognize. Blockade 1 refers to the blockade executed in the *blockade* scenario, and blockade 2 refers to the blockade executed as the first step of the *hammer and anvil* scenario. As shown in the table, the activation network based approach performed very well for each intention. All of them were classified correctly, and each was recognized within the first 13% of the scene.

5.3.2 Threat Level

It is difficult to perform a quantitative analysis of the threat level indicator discussed in Section 5.2.3, due to the ambiguity inherent in defining a “level of threat” for any given scenario. However, it can be seen in Figure 5.3 and Figure 5.4 that the activation level of the “threat level” node does behave as expected. That is, as the number of hostile low-level intentions increases, so does the threat level, and the threat level spikes whenever abnormal behaviors (such as group formation) are detected. Thus, we feel that it is reasonable to say that the threat level indicator shows promise for general high-level intent recognition, though there is definitely room for further development in this direction.

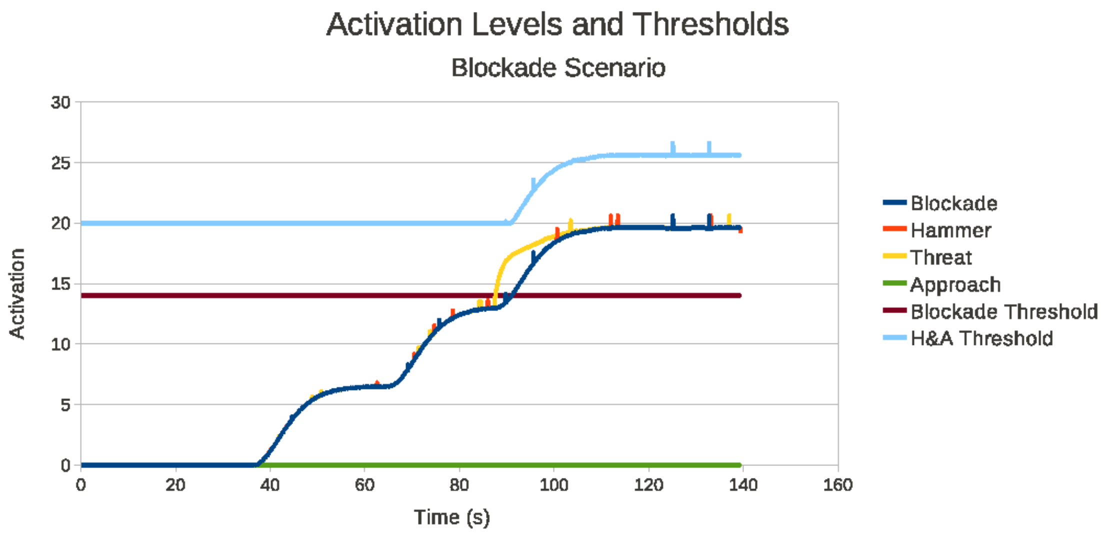

Figure 5.3: Activation levels for the blockade scenario.

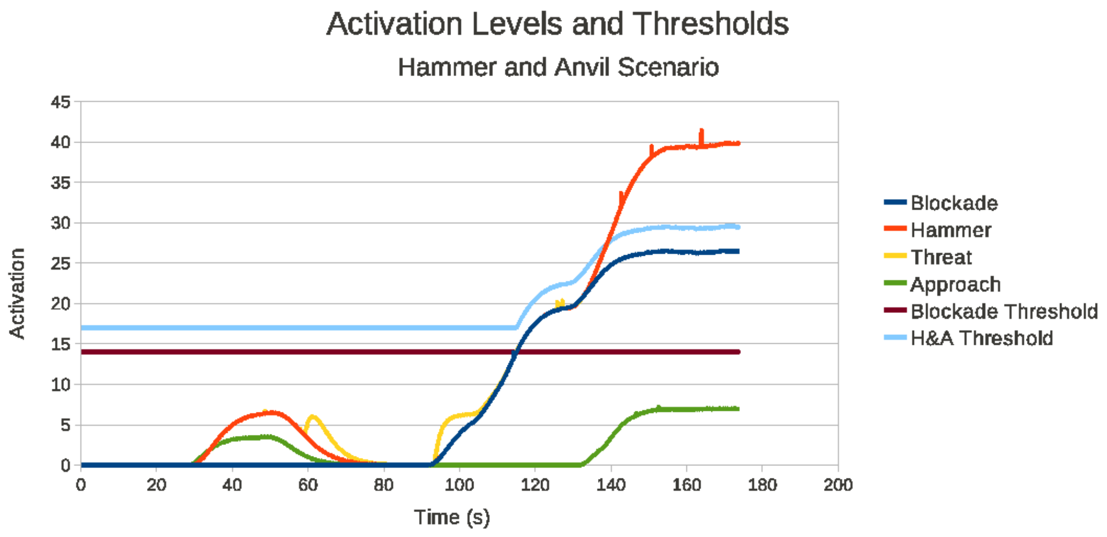

Figure 5.4: Activation levels for the hammer and anvil scenario.

Chapter 6

Conclusions and Future Work

6.1 Remarks

Intent recognition is an extremely important aspect of social robotics. The ability to recognize and react to intentions is not only an integral part of social interaction, but is also useful in adversarial domains, as discussed in this thesis. It is especially important to be capable of performing intent recognition in multi-agent settings, both in terms of recognizing low-level intentions for individual agents, and of recognizing coordinated agents of multiple agents. This is because in real-world domains, it is rare to be performing tasks which require intent recognition in sparsely populated environments.

In this thesis, we presented solutions to various aspects of the intent recognition problem. First, we introduced a framework for testing intent recognition systems in the multi-agent domain. This framework was modular in nature, making it easy to make changes to individual pieces of the system, and included an open-source naval simulator, a low-level intent recognition module, a high-level intent recognition module, and a module for controlling the movements of the simulated agents.

This thesis also presented an extension of the work presented in [17, 18] to the multi-agent domain. We solved the problem of applying the hidden Markov formulation of the intent recognition problem to multiple agents while still operating in real-time by parallelizing various steps of the intent recognition algorithm, and relaxed the constraint that the topology of the HMMs must be designed by hand. In addition to this application of a low-level intent recognition system to the multi-agent domain, we also presented a method for recognizing intentions which requires cooperation between multiple agents. We did this by encoding high-level intentions in an activation network [1]. This approach addressed some of the drawbacks of traditional plan recognition techniques, including the difficulties presented by searching large plan libraries, and the difficulties in recognizing multiple plans which are being performed simultaneously.

6.2 Directions for Future Work

Although Hidden Markov Models have been shown to work well for the recognition of low-level intentions (both in this work and in [17, 18]), there are some significant drawbacks to using them as well. The accuracy of the classification depends greatly on the selection of representative evidence variables, and it can often be difficult to select sufficiently discriminative variables (as in the case of overtaking and passing in Chapter 4). In addition to this problem, it is difficult to make use of the temporal aspects of Hidden Markov Models in the simulated environment, due to the sheer bulk of information being observed in fairly short time-frames. It may be the case that other pattern recognition techniques (such as mixtures of Gaussians) would not suffer in performance due to a lack of temporal modeling, and would alleviate some of the problems involved with selecting representative evidence variables.

It should also be possible to extend the activation network approach to high-level intent recognition in some interesting ways. In this work, no high-level plans with temporal dependencies (where an overall task depends on a specific sequence of actions) were encoded. In addition, the networks in this work used uniform firing thresholds and edge weights. It may be possible to encode more complex intentions by varying either the edge weights, the firing thresholds, or both.

Bibliography

[1] Allan and Elisabeth. A spreading activation theory of semantic processing. *Psychological Review*, 82, 1975.

[2] J. Azarewicz, G. Fala, and C. Heithecker. Template-based multi-agent plan recognition for tactical situation assessment. In *Fifth Conference on Artificial Intelligence Applications, 1989. Proceedings.*, pages 247–254, 1989.

[3] B. Banerjee, L. Kraemer, and J. Lyle. Multi-agent plan recognition: Formalization and algorithms. 2010.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):pp. 164–171, 1970.

[5] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 994–999, 1997.

[6] G. Buccino, F. Binkofski, and L. Riggio. The mirror neuron system and action recognition. *Brain and Language*, 89(2):370 – 376, 2004.

[7] E. Charniak and R. P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53 – 79, 1993.

[8] L. Chuan. cuhmm: a cuda implementation of hidden markov model training and classification. May 2009.

[9] Y. Demiris. Prediction of intent in robotics and multi-agent systems. *Cognitive Processing*, 8(3):151–158, 2007.

[10] Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. In *Robotics and Autonomous Systems*, pages 361–369, 2005.

[11] C. W. Geib. Problems with intent recognition for elder care. In *Proceedings of the AAAI-02 Workshop Automation as Caregiver*, pages 13–17, 2002.

[12] B. P. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[13] K. Han and M. Veloso. Automated robot behavior recognition applied to robotic soccer. In *Robotics Research: the Ninth International Symposium*, pages 199–204. Springer-Verlag, 1999.

[14] G. E. Hovland, P. Sikka, and B. J. McCarragher. Skill acquisition from human demonstration using a hidden markov model, 1996.

[15] M. J. Huber, E. H. Durfee, and M. P. Wellman. The automated mapping of plans for plan recognition. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, UAI'94, pages 344–351, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[16] H. A. Kautz. *A formal theory of plan recognition.* PhD thesis, Bell Laboratories, 1987.

[17] R. Kelley, C. King, and A. Tavakkoli. An architecture for understanding intent using a novel hidden markov formulation, 2008.

[18] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. Understanding human intentions via hidden markov models in autonomous mobile robots. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, HRI '08, pages 367–374, New York, NY, USA, 2008. ACM.

[19] A. Kimmel, A. Dobson, Z. Littlefield, A. Krontiris, J. Marble, and K. E. Bekris. Pracsys: An extensible architecture for composing motion controllers and planners. In *Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, Tsukuba, Japan, 11/2012 2012.

[20] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154 vol.3, 2004.

[21] N. M. N., O. A., L. R., L. S., D. S., M. C., Q. J. C., and A. R. A training simulation system with realistic autonomous ship control. *Computational Intelligence*, 23(4):497–519, 2007.

[22] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 955–960 vol. 2, 2005.

[23] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 578–585, 1993.

[24] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *IEEE international conference on robotics and automation (ICRA)*.

[25] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[26] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 357–360, New York, NY, USA, 2007. ACM.

[27] H. A. Varol, F. Sup, and M. Goldfarb. Multiclass real-time intent recognition of a powered lower limb prosthesis. *Biomedical Engineering, IEEE Transactions on*, 57(3):542–551, 2010.

[28] D. M. Wolpert, K. Doya, and M. Kawato. A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society of London*, 358:593–602, 2003.

[29] H. H. Zhuo and L. Li. Multi-agent plan recognition with partial team traces and plan libraries. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 484–489. AAAI Press, 2011.