Intent Recognition Using an Activation Spreading Architecture

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

by

Mohammad Taghi Saffar

Dr. Mircea Nicolescu, Primary thesis advisor
Dr. Monica Nicolescu, Secondary thesis advisor

August 2015
We recommend that the thesis prepared under our supervision by

MOHAMMAD TAGHI SAFFAR

Entitled

Intent Recognition Using An Activation Spreading Architecture

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

MASTER OF SCIENCE

Mircea Nicolescu, Ph. D., Advisor

Monica Nicolescu, Ph. D., Committee Member

Birant Ramazan, Ph. D., Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

August, 2015
Abstract

Intent understanding is the problem of recognizing people’s goals by passively observing them perform some activities and predicting their future actions. In this thesis, we designed and implemented an efficient intent recognition system with real-time capabilities for two different applications. One of the applications is to recognize intentions in highly populated multi-agent environments with real-time constraints. Low-level intention between all pairs of agents are detected through a novel formulation of Hidden Markov Models with perspective taking capabilities. This layer of our framework only captures interactions between pairs of agents. To move to a higher level and also detect intentions involving multiple agents we use a distributed representation of connecting intentions based on the idea of Activation Spreading Networks (ASN). We utilized an open source naval ship simulator and showed that our system is able to detect intentions reliably and early. Another application is to recognize intentions of humans by observing their activities with an RGB-D camera. Activities and goals are modeled as a distributed network of inter-connected nodes in an ASN. Inspired by a formalism in hierarchical task networks, the structure of the network captures the hierarchical relationship between high-level goals and low-level activities that realize these goals. Our approach can detect intentions before they are realized and it can work in real-time. We also extend the formalism of ASNs to incorporate contextual information into intent recognition. We further augment the ASN formalism with special nodes and synaptic connections to model ordering constraints between actions, in order to represent and handle partial-order plans in our ASN.
Acknowledgements

I would like to thank my wonderful and helpful supervisors Dr. Mircea Nicolescu and Dr. Monica Nicolescu. You made my time in UNR very rewarding and made the transitions into a new environment very easy and I am always grateful to have you as my supervisors. Thank you to Dr. Birant Ramazan for taking the time to see me through this final event of my degree. Finally I would like to thank my great parents. You are the heroes of my life. Everything that I have and every achievement is because of your selfless support and your extraordinary devotion towards me.
Table of Contents

1. Introduction ........................................................................................................................................... 1
   1.1 Plan Recognition ................................................................................................................................. 1
   1.2 Spatiotemporal Pattern Classification ................................................................................................ 2
   1.3 Thesis Goals ......................................................................................................................................... 3
       1.3.1 Intent Recognition in Multi-Agent Domains .................................................................................... 3
       1.3.2 High-Level Intent Recognition in RGB-D Videos ........................................................................... 4
2. Previous Work .......................................................................................................................................... 6
   2.1 Single-Agent Plan/Intent Recognition .................................................................................................. 6
   2.2 Multi-Agent Plan/Intent Recognition .................................................................................................... 8
   2.3 Symbolic Approaches ........................................................................................................................... 9
   2.4 Probabilistic Approaches ..................................................................................................................... 10
3. Multi-Agent Intent Recognition ............................................................................................................. 12
   3.1 Infrastructure ........................................................................................................................................ 12
   3.2 Low-Level Intent Recognition .............................................................................................................. 14
       3.2.1 Hidden Markov Models .................................................................................................................. 15
       3.2.2 Extension to Multiple Agents ......................................................................................................... 20
   3.3 High-Level Intent Recognition ............................................................................................................ 21
       3.3.1 Activation Spreading Networks (ASNs) ........................................................................................... 21
       3.3.2 Intent Encoding and Recognition .................................................................................................. 22
       3.3.3 Threat Level Assignment .............................................................................................................. 24
4. Experimental Results for Multi-Agent Intent Recognition ...................................................................... 26
   4.1 Evaluating the HMM Approach ........................................................................................................... 26
   4.2 Complex Scenarios for Joint Intentions .................................................................................................. 28
   4.3 Evaluating the ASN approach ................................................................................................................ 31
5. High-Level Intent Recognition ................................................................................................................ 35
   5.1 Vision-Based Capabilities ..................................................................................................................... 35
       5.1.1 Background Subtraction .................................................................................................................. 37
       5.1.2 Blob Detection .................................................................................................................................. 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.3 Object Detection</td>
<td>38</td>
</tr>
<tr>
<td>5.1.4 Object Tracking</td>
<td>39</td>
</tr>
<tr>
<td>5.2 Intent Recognition with Activation Spreading Networks</td>
<td>39</td>
</tr>
<tr>
<td>5.2.1 Preliminaries</td>
<td>40</td>
</tr>
<tr>
<td>5.2.2 From Hierarchical Task Network to Activation Spreading Network</td>
<td>43</td>
</tr>
<tr>
<td>5.2.3 Intent Recognition in Activation Spreading Networks</td>
<td>46</td>
</tr>
<tr>
<td>5.2.4 Context-Based Intent Recognition</td>
<td>48</td>
</tr>
<tr>
<td>5.2.5 Partial-Order Modeling in Activation Spreading Networks</td>
<td>50</td>
</tr>
<tr>
<td>6. Experimental Results for High-Level Intent Recognition</td>
<td>56</td>
</tr>
<tr>
<td>6.1 Experiment Domain</td>
<td>56</td>
</tr>
<tr>
<td>6.2 Activation Spreading Networks</td>
<td>60</td>
</tr>
<tr>
<td>6.3 Scenarios</td>
<td>62</td>
</tr>
<tr>
<td>6.4 Experiments</td>
<td>62</td>
</tr>
<tr>
<td>6.4.1 Eating Scenario</td>
<td>63</td>
</tr>
<tr>
<td>6.4.2 Reading Scenario</td>
<td>67</td>
</tr>
<tr>
<td>6.4.3 Drinking Scenario</td>
<td>70</td>
</tr>
<tr>
<td>7. Conclusion and Future Work</td>
<td>75</td>
</tr>
<tr>
<td>8. References</td>
<td>77</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1. Screen-shot of the naval ship simulator environment ........................................... 13
Figure 2. The simulator and intent recognition system architecture ........................................ 14
Figure 3. Low-level intentions in naval ship domain ............................................................... 17
Figure 4. Local state vector for a sample configuration of entities (observable states are changes in these measurements, not their absolute value) .............................................. 18
Figure 5. HMM-based intent recognition .................................................................................. 19
Figure 6. Typical activation spreading inference algorithm ....................................................... 22
Figure 7. Augmented activation network with hammer and anvil intentions ........................... 23
Figure 8. Comparing the speed of serial and parallel implementation of HMM-based intent recognition .................................................................................................................. 28
Figure 9. Activation levels for the blockade scenario .............................................................. 33
Figure 10. Activation levels for the hammer and anvil scenario .............................................. 34
Figure 11. Video parser architecture ......................................................................................... 36
Figure 12. Background subtraction module .............................................................................. 37
Figure 13. Activation message processing algorithm in ASN .................................................. 43
Figure 14. Activation spreading algorithm in ASN .................................................................... 43
Figure 15. HTN to ASN conversion algorithm ......................................................................... 44
Figure 16. Intent recognition algorithm in ASN ....................................................................... 47
Figure 17. Activation message processing algorithm in CASN. The text in blue represents modifications to the original procedure for ASN presented in Figure 13 .................. 50
Figure 18. Activation message processing procedure in POCASN. The text in blue represents modifications to the original procedure for CASN presented in Figure 17 ..... 52
Figure 19. HTN to POCASN conversion algorithm. The text in blue represents modifications to the original algorithm for ASN presented in Figure 15 .......................... 54
Figure 20. Training images used for the object detector in the video parser module ......... 57
Figure 21. Screenshot of our intent recognition system ............................................................. 57
Figure 22. (A) A portion of the ASN generated from the HTN. (B) The same network but in CASN form. (C) The same network in POCASN form. Red lines are max edges, green lines are ordering edges, black lines are min edges, blue lines represent sum edges, grey lines show positive context edges, and orange lines represent negative context edges. Red boxes represent method nodes, grey boxes show context nodes and green boxes represent dummy nodes used to connect ordering edges to sum edges .................. 61
Figure 23. Activation values for the eating scenario with CASN .......................................... 65
Figure 24. Activation values for the eating scenario with ASN .............................................. 65
Figure 25. A subset of activities for the eating scenario with CASN ......................... 66
Figure 26. A subset of activities for the eating scenario with ASN ................................ 66
Figure 27. Activation values for the reading scenario with CASN ............................ 68
Figure 28. Activation values for the reading scenario with ASN ................................. 69
Figure 29. A subset of activities for the reading scenario with CASN ......................... 69
Figure 30. A subset of activities for the reading scenario with ASN ............................ 70
Figure 31. Activation values for the drinking scenario with POCASN ......................... 72
Figure 32. Activation values for the drinking scenario with ASN ............................... 72
Figure 33. A subset of activities for the drinking scenario with POCASN ..................... 73
Figure 34. A subset of activities for the drinking scenario with ASN ........................... 73
List of Tables

Table I. Performance of low-level intent recognition system ........................................... 27
Table II. HMM-based intent recognition in complex scenarios ........................................ 31
Table III. ASN-based intent recognition ........................................................................... 32
Table IV. Kinect camera stream properties ........................................................................ 36
Table V. The effect of contextual information on compound tasks ................................... 58
Table VI. A summary of domain knowledge in the form of HTN ..................................... 59
Table VII. Early detection rates and average confidence of detections for eating scenario ......................................................................................................................... 67
Table VIII. Early detection rates and average confidence of detections for reading scenario ................................................................................................................................. 70
Table IX. Early detection rates and average confidence of detections for drinking scenario ................................................................................................................................. 74
1. Introduction

Intent understanding is the problem of recognizing people’s goals by passively observing them perform some activities and predicting their future actions [1]. This is a crucial capability of an artificial intelligent system which helps provide a proper level of social behavior when interacting with other intelligent agents, including humans. Intent recognition is very similar to the problem of plan/activity recognition, which has been extensively studied [1-3]. Plan recognition is the process of selecting the most suitable plan that an agent is undertaking, based on an observed sequence of atomic actions [4]. Intent recognition aims to predict a high-level activity or goal well before it is realized, in contrast with plan/activity recognition, which addresses the problem of recognizing activities after they are finished. Due to the prediction aspect, a natural requirement of intent recognition systems is to work in real-time. This is especially true for vision-based systems, which have to recognize intentions by analyzing a video stream of a person performing some activities in real-time. Such a capability would be very valuable in many application domains including surveillance, learning by observation and human-robot interaction (HRI).

1.1 Plan Recognition

A plan is normally defined as a set of low level actions with a partial order constraint to represent ordering of actions [5]. Actions that precede another action should be performed prior to that action in order to satisfy its pre-conditions. Actions that have no ordering constraints with respect to each other can be performed in arbitrary order. The
observable evidence for a plan recognizer is one of many possible linearizations of the plan that satisfies the existing partial ordering constraints. Plan/activity recognizers are generally not designed to predict future activities and require observing the whole or a large part of a sequence of low-level actions to robustly recognize the plan. However, early detection is a key requirement for intent recognition systems and this is usually ignored in previous research on plan recognition. Any automatic system for intent recognition should process incoming sensory data and recognize the goal/intention in real-time to support early detection. The mainstream approach in plan recognition is based on searching in a plan library to find a match with the observed evidence [1]. A recent theoretical analysis on the complexity of this problem [6] shows that single-agent plan recognition with a library of known plans and an evaluation function is in class $P$ and multi-agent plan recognition is $NP$-complete. This suggests that the running time requirement of single-agent plan recognizers is related to the size of the plan library; therefore any such system cannot perform in real-time if the plan library is large enough.

### 1.2 Spatiotemporal Pattern Classification

Intent understanding is also related to spatiotemporal pattern classification. Hidden Markov Models (HMMs) and their extensions are widely used for classification of activities in many applications, including vision-based systems such as [7]. Evidence is usually encoded as observable random variables and activities are represented as the hidden states of a Markov process, which models the behavior of the observed agent. However, most vision-based previous work on activity recognition is centered on detecting a particular activity in a specific scenario. Simple activities such as “moving an
object” or “waving the hand”, which usually have a very short time-span, have been the focus of study in prior work [8]. Such activities are normally considered as atomic actions in the planning literature, since they are short and simple enough for the entire system to execute without breaking them down into even simpler actions. Understanding high-level activities and detecting intentions are, however, more challenging problems. Representing high-level activities as hidden states in an HMM is not trivial because they usually consist of several low-level actions and have a long time-span resulting in long and complex temporal patterns that are hard to recognize with Bayesian-based approaches like HMMs.

1.3 Thesis Goals

In this section we describe the contributions of this thesis and the problems we address in two different application domains. First we describe the problem of multi-agent intent recognition and how we utilized Activation Spreading Networks (ASNs) for high level intent recognition in crowded multi-agent systems, and later we describe the problem of intent recognition when activities have a longer time span and are a complex composition of lower level activities in a vision-based intent recognition domain.

1.3.1 Intent Recognition in Multi-Agent Domains

In this part of this thesis we propose a new distributed hierarchical architecture to detect intentions of an individual, or collective intentions in a crowded simulated multi-agent system. Our approach introduces several key contributions: (i) the system has predictive power, which means it can detect intentions well before they are being realized, (ii) the system works in real-time, processing data while it is being gathered and (iii) the system
can detect intentions of individual agents towards other agents, as well as collective and joint intentions in which a team of agents are collaborating on a shared intention. This is a natural multi-agent extension to intent recognition in single-agent systems.

1.3.2 High-Level Intent Recognition in RGB-D Videos

A vision-based intent recognition system is introduced in this section, in order to address the problems mentioned above. We assume that domain knowledge about how certain activities are performed is available in the form of Hierarchical Task Network (HTN). The central idea of the proposed approach is to model intentions, high-level activities and low-level actions in a distributed network of inter-connected nodes where activation spreads in the network via synaptic connections. A low-level activity is a basic action with a short time span such as grabbing an object. A high-level activity/intention such as making tea, is a more complex activity with longer time span and is usually a composite of low-level actions. We show that Activation Spreading Networks (ASN) are suitable for intent recognition, because their graph-based representation can naturally model the hierarchical structure of low-level, high-level activities and goals/intentions. Inference in ASNs is done with spreading activation through the network, which is inherently a distributed task. This feature makes ASNs particularly suitable for real-time intent recognition. Intuitively, nodes represent activities and different edges represent different relationships between activities, such as the decomposition of high-level intentions into low-level actions and partial order constraints between actions. These edges are designed to spread activation messages from low-level actions to high-level intentions. Activation values show the likelihood of each intention. Our approach brings together several key
contributions: (i) early detection of intentions/high-level activities before they are realized; (ii) processing of RGB-D video streams in real-time to detect and track objects, as well as the human subject and its skeleton joint positions in 3D, which are used as basic features in our recognition network; (iii) introducing an ASN-based approach to model hierarchical activities similar to the formalism used in Hierarchical Task Networks (HTNs); (iv) extending the formalism of ASNs to incorporate contextual information into intent recognition; (v) further augmenting the ASN formalism with special nodes and synaptic connections to model ordering constraints between actions, in order to represent and handle partial-order plans in our ASN. The effect of using contextual information and ordering constraints to distinguish between similar intentions has been experimentally tested and the results show that incorporating contextual and ordering information in the network significantly helps in identifying the correct intentions.

The rest of this thesis is organized as follows. Existing methods related to intent/plan/activity recognition are discussed in Chapter 2. In Chapter 3 we explain our multi-agent intent recognition approach. Chapter 4 shows our experimental results for multi-agent intent recognition approach. In Chapter 5 we describe our high-level intent recognition approach followed by its experimental results in our vision-based domain in Chapter 6. We conclude the thesis and provide directions of future work in Chapter 7.
2. Previous Work

In this section we describe work related to our system, by considering plan and intent recognition frameworks and symbolic and probabilistic approaches.

2.1 Single-Agent Plan/Intent Recognition

Plan recognition is an important step in learning by demonstration techniques. In [9] a robotic arm with high degrees of freedom is trained for manipulation. In this work the authors used a k-nearest quantized pattern vector to determine low-level actions and corresponding controls and used a sequence of low-level actions as input to an HMM in order to provide additional context and smoothing. In [10] robot sensory data is used to train an HMM specifically designed to the given task, where each hidden state corresponds to a state in completing the task.

In [11] authors introduced the concept of a coupled HMM that relaxes the Markov assumption by coupling and factoring two or more simple HMMs and used it to recognize two-handed gestures. In [12] a 3D SIFT descriptor is introduced and used with a Support Vector Machine (SVM) classifier to detect the action that is happening in a video. This method performs better than other approaches with the simple SIFT descriptor because of the ability to capture temporal relations in the pattern of the 3D SIFT descriptor. All these approaches are specifically addressing the problem of action recognition, i.e. recognizing actions after the entire sequence has been observed, with no explicit modeling of prediction required for intent recognition.

One of the earliest approaches to intent recognition is introduced in [4], which proposes a
hierarchical library of event types, and plans that are composed of individual events. The method then defines the minimum covering entailment, which chooses a plan for which the library of observed events contains all the events in that plan, and for which the number of unrelated observations (events which occurred but are not part of the plan) 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. In [13] a Bayesian network is used to model intentions, with the highest level node representing the overall intention and the lower level nodes representing sub-actions that contribute to that intention. 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 [14], a biologically-inspired approach to intent recognition is presented. In this paper forward models are introduced that, given the state of a system and the control commands, produce the most likely next state of the system. In addition, inverse models are defined as models that, given the state of a system and its final goal, produce the best set of controls to move towards that goal. These models can be used in a manner similar to mirror neurons [15]. 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. [16] provides a review of some approaches to intent recognition, which existed at the time of its publication. Specifically, it formalizes the ideas of hierarchical representations of intentions, and of perspective taking. For example, [17] 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.

2.2 Multi-Agent Plan/Intent Recognition

As stated in [16], it is not enough to simply recognize the intentions of each individual agent toward 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. In [18], Banerjee et al. present a formalization of multi-agent plan recognition by representing a plan library as a set of matrices and observation of actions as another matrix. The matching in this formalism is NP-complete and can’t handle goal abandonment or plan interleaving. In [19], Zhuo et al. focus on the problem of partial observability. Using [18]’s formulation, they assume that some actions are unknown and try to marginalize on these unknown actions while finding the best match in the plan base. This work still cannot handle plan interleaving and does not scale well to large plan libraries. In [20], the authors proposed a theoretical framework with a combined top-down and bottom-up approach. In the top-down approach the system reasons about global goals and their decomposition into plans. In bottom-up the system observes atomic actions and merges them into plan segments. The main issue in this work is that no experimental evaluations were provided.

Some new approaches in goal/plan recognition are using dynamic networks of connected nodes such as Dynamic Neural Fields [21-24] or semi-Markov decision process graphs [25] to represent tasks for different applications like active learning, intent inference and
even joint intention understanding. Dynamic Neural Fields (DNF) [26, 27] have first been introduced as a simplified mathematical model for neural processing based on recurrent interactions, which neglects the temporal dynamics of individual neurons and uses average firing rate. The main idea behind DNF-models for intent representation is that a distributed network of reciprocally connected neurons forms a complex dynamical system, which can process intent-related information in its activation patterns.

2.3 Symbolic Approaches

Symbolic approaches to goal or behavior recognition are based on abductive reasoning, which – unlike deduction – cannot guarantee a conclusion. Abduction is the process of generating a hypothesis that best explains the observed evidence [28]. This problem has been well studied in multi-agent domains and in applications such as cooperative systems, opponent modeling and agent programming paradigms. In [29], a general framework for plan recognition in the Belief/Desire/Intention (BDI) [30] paradigm is introduced. The authors use a general logical abstraction of plans and Answer Set Programming (ASP) [31] techniques to formalize a non-monotonic reasoning scheme for abduction. In a similar work [32], the same authors formalized incomplete observations to expand the mental state abduction framework in the presence of incomplete or missing information. Another similar work [33] uses situation-sensitive Causal Bayesian Networks (CBNs) for intent recognition in an application for the care of the elderly. Graphical representations of CBNs are translated into a special form of declarative language. Abduction and probabilistic reasoning are combined by using ASP on logical terms obtained from CBNs and probabilistic analysis on CBNs. In a recent work [34],
authors use predicate logic to encode a knowledge base containing a set of decomposition rules describing domain knowledge in the form of hierarchical task networks. A bottom-up abductive reasoning scheme tries to match the observed sequence of actions with corresponding tasks and methods, resulting in recognition of a plan that best describes the evidence. A theoretical study in ASP [35] shows that finding a stable model for a logic program is \textit{NP-complete}. This suggests that such logic programming-based and abductive methods are not good candidates for real-time analysis. In addition, the significant challenge of relating predicates to sensory data makes such symbolic systems of limited use in real-world applications, as they remain mostly theoretical and not appropriate for robotics and particularly computer vision domains.

\section*{2.4 Probabilistic Approaches}

A large body of work uses HMMs to perform temporal inference for action recognition. For example, in [36] Hidden Markov Models are used to represent and recognize strategic behaviors of robotic agents in a game of soccer. After training HMMs corresponding to different actions, it is possible to determine how likely a given sequence of observations was produced from each model. In the scope of that work, 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. [37, 38] also utilize the HMM-based framework presented in [36]. In these approaches, however, intent recognition is performed in far less structured environments. The authors show that this method can work for recognizing the intentions of a human agent toward objects in a scene, or of two human agents toward each other or toward other objects, using only
observable variables from basic sensor information. More complex models, such as parameterized-HMM [39], entropic-HMM [40], variable-length HMM [41], coupled-HMM [11], and hierarchical-HMM [42] have been used to recognize more complex activities. In [43], the authors use stochastic context-free grammars to compute the probability of a temporally consistent sequence of primitive actions recognized by HMMs. Brand and Kettnaker in [40] introduce an entropic-HMM approach to organize the observed video activities (such as office activity and outdoor traffic) into meaningful states. In [13] a Bayesian network is used to model intentions, with the highest level node representing the overall intention and the lower level nodes representing sub-actions that contribute to that intention. In [44] authors use a Bayesian network and propose a two-stage inference process to predict the next activity features and its label in the context of a smart room.

There are many other recent efforts using more complex Bayesian net approaches for recognizing human activities [45-49]. Bayesian nets and specifically HMMs are shown to be working very well for recognizing activities that are simple and short. However, they tend to suffer from overfitting training data if they are used to recognize complex and long activities. There has been little research on real-time probabilistic and specifically Bayesian-based intent recognition systems in a hierarchical manner. Probabilistic approaches usually have difficulties recognizing complex, long spatiotemporal patterns. It is not straightforward to model hierarchical relationships between low-level actions and high-level intentions in such systems. Also it is not easy to incorporate domain knowledge into such frameworks and they rely on extensive datasets for training.
3. Multi-Agent Intent Recognition

In this chapter we describe our proposed method for intent recognition in multi-agent systems. We also introduce the domain application used for our experiments and we describe the development of the implemented system. The proposed intent recognition approach is capable of recognizing the intentions before they are being realized. It works in real-time and it can detect both single-agent intentions as well as joint intentions of a group of agents.

3.1 Infrastructure

We evaluate our approach for multi-agent intent recognition in the context of detecting threatening intentions against Navy ships, using an open-source naval ship simulator [50] as the base for our simulation framework. We built a fully functioning test-bed for intent recognition systems on top of this simulator. The open source naval simulator [50] is very appropriate for our problem, as it supports the simulation of multiple ships, operates in real-time, and is deterministic (which allows for precise reproduction of experiments). Within this environment we implemented behaviors that correspond to our list of intentions of interest and also the capability to create and autonomously run complex scenarios involving large number of boats. A snapshot of the simulation environment is shown in Figure 1.
Figure 1. Screen-shot of the naval ship simulator environment

For better modularity and for decoupling the intent recognition system from the simulator, we created separate processes for controlling the simulation and for intent recognition. These two components communicate via the Robot Operating System (ROS) [51], using its publish/subscribe approach.

To make the experiments realistic we implemented a visibility module to filter out any information that is hidden from the viewpoint of an entity in the simulator (ships, oil platforms, airplanes, helicopters, etc.). This module considers any landscape in the line of sight between two entities and publishes local state vectors for each pair of entities. The state vector contains the following local information: the distance between the entities (\(d\text{ist}\)), the angle of the line connecting them (\(\text{cross angle}\)), the direction (\(\text{orientation}\)) and speed of movement for each entity, and finally a visibility flag showing if the second entity is visible from the first entity’s viewpoint. In the domain of
naval ships, it is also very important to appropriately react to any malicious intentions as soon as they are detected. This however is mostly out of the scope of this thesis, but we implemented a simple controller module for the system, which overwrites the actions of the entities if some specific intentions are detected. This module is simply a collection of pre-written behaviors and their corresponding intention stimuli. Finally, another component was added to the simulator for visualizing detected intentions. The components and their connections are shown in Figure 2.

![Diagram of simulator and intent recognition system architecture](image)

Figure 2. The simulator and intent recognition system architecture

### 3.2 Low-Level Intent Recognition

In this section we formally describe our proposed approach for intent recognition in the naval ship domain. This part of the system contains two separate layers. In the lower layer, we are using Hidden Markov Models (HMM) formalism for detecting low-level intentions of each entity towards another entity in the environment. Following previous
work in using HMMs for intent recognition [37, 38], our HMM formalism is based on the idea of *perspective taking*. As shown in Figure 2, the input to this layer is simply the local perspective state generated by the visibility module and the output is the recognized intention for each pair of entities. The second layer is responsible for detecting multi-agent (joint) intentions. This part of the system mainly focuses on collaborative behaviors that are a sign of joint intentions. This layer is based on the idea of a distributed model of computation, Activation Spreading Networks (ASN), inspired by [52].

### 3.2.1 Hidden Markov Models

As previously mentioned, we use HMMs to model low-level activities in our system in order to capture pairwise interactions between entities. An intention that an agent can have toward any other agent is represented as a set of Hidden Markov Models. An HMM is a probabilistic method to model a Markov process with unobserved states, and consists of a set of hidden states, a probability distribution on transitions between hidden states (transition probability), a set of visible or observable states, and a probability distribution on observing visible states given that the system is in a particular hidden state (emission probability). In our particular low-level intent recognition problem, each possible low-level intention is modeled as a separate set of HMMs. In each HMM, we are trying to recognize the intention of only a single entity (actor) toward another entity (target). We denote the intention of actor *A* towards target *T* as $I^T_A \in \bigcup_{k=1}^{m} i_k$ in which $i_k$ represents the $k^{th}$ low-level intention in our system and $m$ is the total number of low-level intention types that the system can recognize. Any low-level intention $i_k$ has a set of corresponding HMMs $M_k = \bigcup_{j=1}^{s} m^j_k$ that have different number of hidden states starting from 1 to $s$. 
Previous work [38] suggests that to accurately model an intention as a HMM, we need to have a meaningful interpretation of hidden states. The interpretation of hidden states used in [38] is that they encode representative sub-activities of the particular intention being modeled. This break-down typically requires expert knowledge about the domain at hand, which is not always available. In addition, in many domains it is possible to have multiple ways of accomplishing an intention, which is precluded if we allow only a single model for each intention. Following these observations we therefore employ a set of HMMs $M_k$ for each low-level intention $i_k$ in our system. These models have different numbers of hidden states, all of them representing the same intention. At recognition time, to detect the intention of an actor towards a target, we use cross-validation to select the best model for each intention. In the next section we describe how the HMMs were used to model activities in the naval ship domain and in Section 3.2.1.2 we describe how intent recognition is performed.

3.2.1.1 Activity Modeling

For our particular naval ship domain we are interested in detecting the following 5 different low-level intentions: approach (one boat heads directly to another), pass (one boat passes another in opposite directions), overtake (one boat passes another in the same direction), follow (one boat maintains the same distance and heading with respect to another) and intercept (one boat heads toward a point of another boat’s trajectory). These represent the intention of an actor agent toward a single other target agent. A simple schematic representation of these intentions is shown in Figure 3.
The observable states in our approach come from the local perspective information that is generated by the visibility module described in Section 3.1. Formally the set of observable states is a set of tuples of the form $T = \langle d, \alpha, o_A, o_T, \delta \rangle$ in which $d \in \{increasing, decreasing, constant\}$ is the change in distance between actor and target, $\alpha \in \{increasing, decreasing, constant\}$ is the change in the angle between the actor’s current direction of movement and the straight line from actor toward target, $o_A, o_T \in \{facing, not facing\}$ are the orientations of actor toward target and orientation of target towards actor respectively, and finally $\delta \in \{the same, not the same\}$ is the difference in headings for the actor and the target. Figure 4 shows how an actual configuration of two entities relates to the observable states defined above. The set of all possible tuples $T$ forms the alphabet of observable states and contains a total of 72 different tuples. As described here, an observable state encodes changes to visible variables that are relevant to the intentions we want to detect from the actor’s perspective. This is in agreement with findings in [53] about how humans take the perspective of others to correctly infer intentions.
For each one of the 5 intentions, we created 4 different models with different number of hidden states ranging from 1 to 4. We chose 4 because we believe that the low-level intentions we are trying to detect at this stage are not complex enough to need more than 4 hidden states. Clearly this number is domain-dependent and can be changed for other problems and intentions. To train the HMM models, we generated 20 different examples for each intention in our simulator, each with randomized starting boat configurations. The system state was then logged at each time step for each example, in order to generate data for training. We then computed observable variables for each frame of each generated example, and then used it as the training set. We trained our models with the standard Baum-Welch algorithm [54]. The only requirement for this algorithm is a known topology of the HMM, which is satisfied by specifying the number of hidden states.
3.2.1.2 HMM-Based Intent Recognition

Once the HMMs for each intention have been trained, recognizing intent becomes a problem of pattern classification. At run time, we calculate the observable variables for each agent in a scene with respect to every other agent, and we determine the model that is most likely to have generated a given sequence of observables. This is done using the forward algorithm [55], which, given an HMM and a sequence of observable variables, returns the log-likelihood of the given HMM generating that particular sequence of variables. We compute this likelihood for each trained model and we choose the one with the highest probability of generating the observed sequence, and classify the intention accordingly. Figure 5 shows a schematic view of this HMM-based intent recognition approach in our system.

![Figure 5. HMM-based intent recognition](image-url)
3.2.2 Extension to Multiple Agents

The approach to intent recognition presented so far is shown to work quite well in the work presented in \cite{37, 38}. However, it is subject to some constraining assumptions. First, while the algorithm will work in its current form for scenarios involving multiple agents, the forward algorithm is computationally complex and the intent recognition process cannot scale to scenarios involving many agents while still performing in real-time. Since this approach is based on the idea of perspective taking, for \( n \) number of entities in the environment we need to detect \( n^2 \) number of intentions. If we have \( m \) number of low-level intentions and for each low-level intention we have \( s \) different HMMs, then for detecting each intention we need to apply the forward algorithm for \( m \times s \) HMMs, which means that for each frame we have to run the forward algorithm \( m \times s \times n^2 \) times, which becomes infeasible for real-time intent recognition. Second, the HMM-based approach to intent recognition assumes that each agent's intention toward each other agent is completely independent of all other intentions. 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 that involve coordinated efforts among multiple agents.

In our early experiments with the ship simulator, we discovered that the HMM method tends to become too slow for performing in real-time when more than 5 agents are present in the scenario. However we can take advantage of the independence assumption and simply parallelize each step of our algorithm. We calculate the observable variables independently and in parallel for each pair of agents in the scene, using a CUDA kernel. Once each sequence of observable variables has been calculated, we use the parallel
implementation of the forward algorithm presented in [56] to calculate the log-likelihood for each intention, for each pair of agents simultaneously. Once this has been done, another CUDA kernel is used to choose 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.

3.3 High-Level Intent Recognition

As discussed in Section 2.2, the general approach to multi-agent plan recognition defined by [18] is to maintain a library of joint plans and search for plans that contain the list of observed actions as a subsequence. In the naval ship domain, a joint plan for creating a blockade might involve at least three agents performing an intercept. The problem with using a plan library is twofold. First, plan libraries might become quite large, slowing the search process, which becomes an issue if real-time computation is desired. Second, goal abandonment or plan interleaving is not easily supported.

In this section we propose to address these problems by using activation spreading in a hierarchical intent network inspired by [57] for the task of high-level intent recognition.

3.3.1 Activation Spreading Networks (ASNs)

An ASN can be represented as a set of neurons that are connected to each other via synaptic connections. Mathematically, an ASN is a directed graph $G = (V, E)$ in which $V = \{v_1, ..., v_n\}$ is a finite set of vertices (neurons) and $E = \{e_1, ..., e_k\}$ is a set of directed edges (synapses) connecting two vertices. For simplicity, we use the $e_{ij}$ notation to represent an edge from $v_i$ to $v_j$. Each vertex $v_i$ has an activation value $A_i$ and each edge
$e_{ij}$ has a weight $w_{ij}$. In addition to weights and activation values, there is also a firing threshold $F$ and a decay factor $D$ associated with the network. We will extend the typical definition of ASN in Chapter 5 to make it more suitable for complex intent recognition applications.

The typical approach for activation spreading inference is shown in Figure 6. In the next section we modify this basic algorithm and we show how to represent and recognize high-level joint intentions with activation networks.

$$\begin{align*}
&\text{for } i = 1..n \\
&\quad A_i = 0 \\
&\text{set } A_i = x; \ x > F \text{ for some origin nodes} \\
&\text{for each } i \text{ so that } A_i > F \text{ and } v_i \text{ is unfired} \\
&\quad \text{for each } j \text{ so that } e_{ij} \in E \\
&\quad \quad A_j = \max(A_j + (A_i \times w_{ij} \times D), 0) \\
&\quad \text{repeat} \\
&\text{repeat}
\end{align*}$$

Figure 6. Typical activation spreading inference algorithm

### 3.3.2 Intent Encoding and Recognition

The activation spreading network allows us to encode complex, possibly hierarchically structured intentions, through the topology of the network. In the network, low-level nodes correspond to the low-level intentions recognized by the HMM approach discussed in Section 3.2.1. We create one low-level node for each pair of agents and each low-level intention that can occur between them. In our work, this means that for each pair of nodes $(i, j)$, we will have 5 low-level nodes associated with the 5 different intentions. Nodes that represent higher-level intentions receive activations from the low-level nodes or other
high-level intentions. For simplicity we use 1 as the weight for the connecting edges, the decay factor $D$ is set to 0.95 and the firing threshold $F$ is set to 0. For example, for the blockade example, we would connect all the low-level nodes corresponding to the intercept intention (from all entities to a given target) to a higher-level node, representing blockade. It is also relatively simple to encode multiple intentions in a single network. For example, if we wanted to recognize both a blockade and a hammer and anvil attack, we can simply modify the blockade network by adding low-level nodes corresponding to the approach intention, and adding a node representing hammer and anvil attack. We can then add a link from the new origin nodes to hammer and anvil attack, as well as a link from blockade to it, as shown in Figure 7.

![Figure 7. Augmented activation network with hammer and anvil intentions](image)

To recognize high-level intentions we run the algorithm from Figure 6 in a continuous loop, where at each step activation is set on the low-level nodes (based on the maximum likelihood selection of forward probabilities of HMMs with different number of hidden states for each low-level intention at this time step) and propagated through the graph as described above. At each time step activation values decay based on the following
formula. \( A_i = A_i \times D \). By doing this, activation will accumulate in the various intention nodes and detecting an intention can be done by checking that the activation level of a node is above a certain detection threshold.

This simple approach has some problems and it works only for simple ASNs. Since higher level nodes in the network accumulate activation values from numerous lower level nodes, they usually tend to have higher activation values compared to lower level nodes. For example the node for *hammer and anvil* always has a higher activation value than the *blockade* in Figure 7. If the activation value of *blockade* is above the detecting threshold, then *hammer and anvil* is also above that threshold. To avoid this problem we are automatically adjusting the detecting threshold values in the network as explained in Section 4.3. However even with adjustable thresholds we cannot reliably detect high-level intentions if the network contains several possible intentions at the same level in the structure of the network. The activation values of the nodes in the network can be above the adjustable detecting threshold and this makes the system detect several possibly conflicting intentions. A more suitable formalization of ASN and high-level intent recognition for networks with lots of possible intentions at different levels is introduced in Chapter 5 that addresses these issues and makes ASNs work reliably to detect the correct intention among many different options.

**3.3.3 Threat Level Assignment**

Encoding intentions in activation networks addresses the problem of maintaining a library of plans to recognize, as well as the problem of performing an exhaustive search in that plan library. Using a topology-based approach to encoding intentions, it is possible
to add new intentions without a significant increase in the size of the network (as seen in our examples above). In our method, we simply need to monitor 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 low-level nodes. However, it is still necessary to encode each high-level intention by hand, and specify the topology of the network accordingly.

In the naval ship domain, it is very desirable to also assign a general threat level to the state of the scene, in addition to particular high-level intentions. For this purpose we add a threat level node to our activation network for each target agent and we connect it to all hostile intentions (low-level intentions like intercept, approach, etc. and high-level intentions like blockade and hammer and anvil) toward that target. The degree of threat for each agent will then be proportional to the activation level of its corresponding threat level node.
4. Experimental Results for Multi-Agent Intent Recognition

In this chapter we describe how the naval ship simulator introduced in Section 3.1 was used for the experimental evaluation of our proposed method for multi-agent intent recognition described in previous chapter.

4.1 Evaluating the HMM Approach

To test the baseline accuracy of the HMM-based approach to low-level intent recognition, we trained models for 5 different intentions: approach, pass, overtake, follow, and intercept. We then generated 200 two-agent scenarios using the simulation system, resulting in 40 test scenarios for each of the individual 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 [58]:

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 from 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 the accuracy rate and the average correct duration to be close to 100%, and the average early detection to be close to 0%. The results of our experiments are shown in Table I. As can be seen, the system is able to detect all intentions correctly (accuracy) and for most of the simulation time (average correct duration). 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.

Table I. Performance of low-level intent recognition system

<table>
<thead>
<tr>
<th>Intention</th>
<th>Accuracy (%)</th>
<th>Average early detection (%)</th>
<th>Average correct duration (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>approach</td>
<td>100</td>
<td>8.95</td>
<td>90.9</td>
</tr>
<tr>
<td>pass</td>
<td>100</td>
<td>68.0</td>
<td>96.5</td>
</tr>
<tr>
<td>overtake</td>
<td>100</td>
<td>56.8</td>
<td>64.6</td>
</tr>
<tr>
<td>follow</td>
<td>100</td>
<td>1.92</td>
<td>99.3</td>
</tr>
<tr>
<td>intercept</td>
<td>100</td>
<td>11.3</td>
<td>88.8</td>
</tr>
</tbody>
</table>

The poor performance of the pass and overtake intentions on early detection rate is mainly because of the lack of distinguishing observable variables in our HMM-based module for these two intentions. This is especially true when the agents have come close to each other. Given the observable variables discussed in Section 3.2.1.1, the only difference between pass and overtake at this point would be change in angle from target to actor which is likely not enough to result in a distinct classification.

To evaluate the effectiveness of parallelizing the intent recognition process, we implemented both serial and parallel versions of the HMM-based intent recognition
algorithm and ran them on 17 scenarios containing varying numbers of agents. We then recorded the average frame rate over each scene with the results shown in Figure 8. While the performance of the serial implementation 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 adequate for performing in real-time.

![Throughput](image)

Figure 8. Comparing the speed of serial and parallel implementation of HMM-based intent recognition

### 4.2 Complex Scenarios for Joint Intentions

To evaluate the performance of our proposed approach for multi-agent intent recognition, we first need to create more complex scenarios involving multiple agents. To this end, we created 7 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 all of the scenarios there are high-value target ships (vessels, aircraft carriers and oil platforms) which could be the target of coordinated attacks by smaller boats. For the scenarios we have a representation of land.
In scenario 1 (16 ships in total), a convoy of naval vessels is attempting to traverse the straits. As they do so, a pair of other ships passes close by to 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 scenario 2 (17 ships) is constructed similarly: a group of naval vessels is attempting to exit the 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 naval vessels.

In scenario 3 (7 ships), 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.

In scenario 4 (14 ships, 1 oil platform and a helicopter), two naval vessels are patrolling around an oil platform. There is a helicopter on the platform. A small boat starts a suicide run towards one of these vessels to create a diversion and at the same time another boat on the other side of the platform breaks free from a fishing boat swarm to attack the oil platform.

In scenario 5 (an aircraft carrier, a jet fighter and 16 ships), an aircraft carrier and another naval vessel are moving in a congested area with big tankers and fishing boats. A small boat hides behind one of the tankers waiting for the carrier to get close to attack it. Another boat which was behaving like a fishing boat starts a suicide run towards the carrier from the other side at the same time.

Scenario 6 and scenario 7 are examples of scenarios in which more complex intentions (in which agents must cooperate to perform a task) may occur. In scenario 6 (4 ships), 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. Scenario 7 (7 ships) 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 its escape.

In performing a quantitative analysis of the more complex scenarios, we first define key intentions as those intentions that make up actions that are threatening to the naval vessels in the scene. For instance, in scenario 6, 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 focus on the average early detection for key intentions in each scene, and on the accuracy rate for those key intentions as well.

The accuracy rate for our system is 100% for key intentions in complex scenarios, which means all key intentions were correctly recognized in all complex scenarios. In addition, it can be seen in Table II that the early detection rate for the key intentions is below 9%. In every case, the key intentions were recognized almost as soon as they began. It is worth mentioning that we have the ground truth for intentions in each of the scenarios. The total time $T_i$ that a particular intention is active is the total time for that particular intention to be active in the ground truth.

It is important to note that the system is able to detect the intention with the highest likelihood without any pre-segmentation of the observation trace: as new behaviors are
performed in the scenario, the system is capable of transitioning from one detected intention to another. We only use the ground truth information (exact duration and timing of each intention) for the qualitative evaluation.

Table II. HMM-based intent recognition in complex scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Key intentions</th>
<th>Average early detection ratio for key intentions (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>approach, intercept</td>
<td>1.50</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>approach, intercept</td>
<td>2.31</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>approach, overtake</td>
<td>3.03</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>intercept</td>
<td>0.00</td>
</tr>
<tr>
<td>Scenario 5</td>
<td>approach, intercept</td>
<td>5.04</td>
</tr>
<tr>
<td>Scenario 6</td>
<td>approach, intercept</td>
<td>4.23</td>
</tr>
<tr>
<td>Scenario 7</td>
<td>approach, overtake,</td>
<td>8.40</td>
</tr>
<tr>
<td></td>
<td>intercept</td>
<td></td>
</tr>
</tbody>
</table>

4.3 Evaluating the ASN approach

In order to quantitatively evaluate our activation network-based approach to high-level intent recognition, we designed two scenarios, scenario 6 and scenario 7, in which cooperative intentions were taking place. A description of each of these scenarios can be found in Section 4.2. In scenario 6, we want the system to recognize the blockade behavior before it is completed, and continue to recognize it as long as it is happening. Similarly, for scenario 7, our system must first recognize the blockade when it happens, and then recognize the hammer and anvil attack as the other ships begin to approach from
behind. We will use the early detection and the accuracy rate metrics to evaluate our results.

Table III shows the results for each of the three intentions we wanted to recognize. Blockade 1 refers to the *blockade* executed in *scenario 6*, and blockade 2 refers to the *blockade* executed as the first step of the *hammer and anvil* in *scenario 7*. As shown in the table, the activation network-based approach performed very well for each intention. All of them were classified correctly, therefore the accuracy rate of our ASN-based approach to multi-agent intent recognition is 100% and each was recognized within the first 6% of the scene.

<table>
<thead>
<tr>
<th>Intention</th>
<th>Early detection rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>blockade 1</em></td>
<td>3.12</td>
</tr>
<tr>
<td><em>blockade 2</em></td>
<td>3.10</td>
</tr>
<tr>
<td><em>hammer and anvil</em></td>
<td>5.23</td>
</tr>
</tbody>
</table>

Figure 9 and Figure 10 show activation levels for nodes in our ASN corresponding to high-level intentions, the threat level and the detection threshold values we are using to detect high-level intentions. It is important to note that the absolute value as thresholds for detecting *hammer and anvil* is actually equal to the initial threshold value for that intention plus any positive difference between the activation level of *blockade* and the *blockade* threshold. This is because *hammer and anvil* is placed higher in the topology of the network compared to *blockade* and the weight of their connection is 1, therefore the minimum activation level for *hammer and anvil* is always greater than or equal to the *blockade* activation level. If we used a fixed threshold we might wrongly detect a
hammer and anvil if the activation level of the blockade node becomes greater than hammer and anvil fixed threshold.

It is difficult to perform a quantitative analysis of the threat level indicator, due to the ambiguity inherent in defining a “level of threat” for any given scenario. However, it can be seen in Figure 9 and Figure 10 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.

![Activation Levels and Thresholds](image)

**Figure 9. Activation levels for scenario 6**
Figure 10. Activation levels for scenario 7
5. High-Level Intent Recognition

In this chapter we describe the method we developed for ASN-based high level intent recognition in RGB-D video streams. We first introduce the vision-based system that we used for extracting features from RGB-D video streams. We then formally define our high-level intent recognition system. The proposed intent recognition approach is capable of recognizing intentions before they are finalized, works in real-time and can detect complex intentions. It will also address the shortcomings of the intent recognition system proposed in Chapter 3: unlike that system, it can reliably choose the correct intention among numerous and even conflicting options under the same framework with a single ASN. It can distinguish between different intentions that only differ in their ordering of lower level activities, and it can also take advantage of contextual information to disambiguate between similar intentions to detect the correct one.

5.1 Vision-Based Capabilities

In order to detect human intentions, we process a video stream from a scene where a person is performing various activities, including both short and long time span actions in accordance to high-level intentions. This data stream is fed into a video parser to extract useful features for intent recognition. The scene is observed with a Microsoft Kinect camera, which provides both depth and RGB frames for the scene. In addition, we use the Microsoft Kinect skeleton tracking package available in the Microsoft Kinect SDK for Windows [59]. The camera position and orientation does not change during recording. Table IV shows the properties of the different Kinect streams used in our experiments.
Table IV. Kinect camera stream properties

<table>
<thead>
<tr>
<th>Stream</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB frame</td>
<td>640x480 pixels, 30 fps</td>
</tr>
<tr>
<td>Depth frame</td>
<td>640x480 pixels. Format=13 bit depth in mm + 3 bit player index. Min 8 cm, max 4 m</td>
</tr>
<tr>
<td>Skeleton</td>
<td>20 joints. Format=(x, y, z) metric 3D joint position in the camera coordinate system</td>
</tr>
</tbody>
</table>

The basic features that we use for intent recognition are the human pose and object locations in the 3D metric camera coordinate system. Obviously, detecting more features (such as object pose) can help the recognition system, but estimating these more complex features requires more processing time and they are not usually as robust as the simpler ones. Human joint locations are obtained in real-time with the skeleton tracking package from Microsoft. In order to robustly estimate object locations we developed a processing pipeline that works on RGB and depth frames in real-time and estimates object locations in 3D. Figure 11 shows the overall architecture of the video parser. We describe the individual stages of the pipeline in the following subsections.

![Figure 11. Video parser architecture](image-url)
5.1.1 Background Subtraction

As the first step in our video parser, background subtraction helps reduce the computation time significantly in later stages. By taking advantage of the fixed camera assumption in our scenarios, we probabilistically model the background and use foreground regions for object detection. Figure 12 shows the steps for background subtraction in our system. We use both RGB and depth frames to build a robust background subtraction module with little sensitivity to aperture changes in the RGB camera. We use the Codebook background modeling [60] on RGB frames and a Mixture of Gaussians (MoG) background modeling [61] on depth frames to independently build two background models in RGB and depth space, and consequently determine two foreground masks. An aggregated mask is obtained by applying a bitwise AND operation between the two separate foreground masks. Finally, to remove small pixel-level noise in the foreground mask, we use a contour-based segmentation algorithm to refine foreground regions by filling out small holes and removing isolated pixels. We used the OpenCV library in most parts of our video parser.

Figure 12. Background subtraction module
5.1.2 Blob Detection

The output of the background subtraction stage is a binary foreground mask. In blob detection, we detect individual candidate blobs in the foreground mask that contain foreground objects, by finding connected components in the mask. The connectivity is simply defined by neighboring pixels with the same binary value in the foreground mask. It is important that blobs contain only a single foreground object before initializing the object tracker, since we are using a holistic object descriptor for detection. We assume that the foreground objects do no overlap in the RGB image when they initially appear in the scene. This is no longer required after initializing the object tracker. An additional issue is caused by the foreground regions corresponding to people in the scene, rather than objects. We use the human body segmentation information provided by Microsoft Kinect (on the depth image) to discard foreground regions that overlap with the segmented body, in order to avoid considering them as candidate blobs for object detection.

5.1.3 Object Detection

We use the detected blobs in the foreground as inputs to a multi-class Support Vector Machine (SVM) [62] classifier, which was trained using images for objects of interest in our scenarios. Since real-time processing is a crucial feature of video-based intent recognition, we decided to use normalized color histograms as features for classification, which are significantly faster to compute compared to more sophisticated features such as SIFT [63]. Since color histograms are holistic properties, a segmentation of objects in the RGB space is required, and the blob detection module discussed in Section 5.1.2 provides the segmentation we need. We used 8 bins for hue and 8 different quantized values for
saturation in each bin. The distance measurement used for the classification of color histograms is the Bhattacharyya distance [64]; if the minimum distance of a blob’s color histogram from any of the training color histograms is above a threshold, then no class will be assigned to that blob, showing no known object is present in that region.

### 5.1.4 Object Tracking

After robustly detecting an object in one of the blobs for several consecutive frames, we start tracking the object in the scene and remove the corresponding class from the classifier in the object detector to reduce the computational requirements of the system, since running the object detector on every blob in every frame is a time-consuming process. To track objects we use the Continuously Adaptive Meanshift (Camshift) [65] algorithm. For simplicity, we assume that no objects would leave the scene once they are detected. This makes tracking easier, avoiding the need to stop or re-initialize the tracker when objects leave the field of view and then re-enter. The output of the object tracker is a region in the RGB frame corresponding to each detected object. We use the centroid of that region and the depth information from the depth image to estimate the location of the objects in the 3D camera coordinate system.

### 5.2 Intent Recognition with Activation Spreading Networks

In this section we formally define our ASN-based approach to intent recognition from RGB-D videos. Activation Spreading Networks provide a parallel, distributed and fast search mechanism for intent recognition. Through spreading activation messages to other nodes in the network and accumulating activation by receiving messages from neighbor nodes, we can robustly detect the intention of a subject of interest and the corresponding
plan that the subject is following to realize that intention. We can predict a set of future activities of the subject based on the detected plan. These capabilities depend on designing an activation spreading network that captures the real structure of activities, plans and intentions related to the system. This is the domain knowledge, which (in different forms) is incorporated in every planning system. One of the most widely used forms of representing planning knowledge is Hierarchical Task Networks (HTN) [66]. The main idea behind HTNs is to store mini-plans to achieve common goals in a database of reusable methods, and to use them while planning – for fast processing. Theoretical studies [67] show that HTNs in their unrestricted form are actually more complex than Partial Order Planning (POP) [5]. Only after enforcing some limitations on the expressivity of HTNs, this form of planning becomes tractable. In the following subsections we formally define ASN and HTN, provide an algorithm to build an ASN from an HTN, and describe an inference algorithm based on ASN for intent recognition.

5.2.1 Preliminaries

We adapt the definition of HTNs from SHOP² [68] which is a well-known HTN planner.

**Operator:** an operator \( R(v_1, ..., v_n) = (name(R), pre(R), add(R), del(R)) \) is a parameterized strips-like atomic action where \( v_1, ..., v_n \) are variables used in precondition, add and delete lists. Each variable \( v_i \) has a set of all possible substitutions or domain \( D(v_i) \).

**Task:** A task \( T(x_1, ..., x_m) \) is either a primitive or a compound task where \( T \) is task symbol along with a list of terms \( x_1, ..., x_m \) as arguments. If task \( T(x_1, ..., x_m) \) is primitive then \( T \) is an operator name and the task arguments are used as the operator
parameters.

**Method:** A method $M = (\text{name}(M), \text{task}(M), \text{pre}(M), \text{subtasks}(M))$ is a possible expansion of the non-primitive task $\text{task}(M)$ and it is only applicable in situations satisfying the precondition $\text{pre}(M)$. Intuitively, a method represents a particular way of achieving a given task.

**Task Network:** a task network $N$ is a tuple of the form $(U, <)$ where $U$ is a set of tasks and $<$ is a partial order constraint on $U$. If $U$ contains only primitive tasks, it is called a primitive task network, otherwise it is called a non-primitive task network.

**Hierarchical Task Network:** a hierarchical task network is a set of operators, methods, task networks and tasks. Intuitively, a hierarchical task network is a representation of the planning domain knowledge.

The ultimate goal in HTN planning is to complete a task. Usually the goal task is compound and the planner should choose a suitable method from the set of available methods to breakdown the goal task into smaller tasks. This recursive procedure continues until all tasks in the network are primitive. HTN planners are equivalent to context-free grammars in their set of possible solutions [67], but to simplify our intent recognition problem, we restrict the HTN formalism to avoid recursion in the methods. Put differently, we assume that no compound task can be a member of the subtasks of itself. The intention of completing a compound task is the focus of our intent recognition system. From now on when we mention “detecting a task” we implicitly mean to detect the intention to complete a task. Next, we formally define an activation spreading network.
Activation Spreading Network: An activation spreading network \( G = (V, E_S, E_M, cl, F, d) \) is a directed acyclic graph.

- \( V \) is the set of nodes. Each node \( v \in V \) has an activation value \( ac(v) \) that is a positive real number.
- \( E_S \) is the set of sum edges connecting nodes in the graph. Each sum edge \( e \in E_S \) has a weight \( w(e) \leq 1 \). A sum edge is an edge through which activation messages spread in the network and the receiving node processes it by updating its activation value with a summation.
- \( E_M \) is the set of max edges connecting nodes in the graph. A max edge is another type of edge through which activation messages spread in the network and the receiving node processes it by updating its activation value with a maximum value selection.
- \( cl \) is the internal clock sending periodic signals to all nodes in the graph.
- \( F \) is a firing threshold.
- \( d \) is a decay factor that is a real number between 0 and 1.

A node would update its activation value by multiplying it with the decay factor \( d \) on every clock tick. Upon receiving an activation message, a node would update its activation value by summing the activation message multiplied by the edge weight, with its own activation value if the message was received via a sum edge. A node would update its activation value by choosing the maximum activation message on all ingoing max edges. Upon receiving a tick from the clock, a node would send activation messages equal to its activation value on outgoing edges, if its activation value is above \( F \). Figure
13 shows the algorithm of processing activation messages in the ASN. This procedure is called for a node upon receiving an activation message. Figure 14 shows the algorithm of activation spreading in the network. This procedure is called for a node upon receiving a periodic signal from the clock.

**Activation Processing Procedure**

let $v \in V$ be the node receiving activation message from $s$

let $S_{\text{max}} = \{n|(n, v) \in E_M \text{ and } n \text{ sent message to } v \text{ in recent clock signal}\}$

if $S_{\text{max}} \neq \emptyset$

$$ac(v) = \max_{v' \in S_{\text{max}}} ac(v')$$

else

$$ac(v) = ac(v) + ac(s) \times w((s, v))$$

**Activation Spreading Procedure**

let $v \in V$ be the node receiving periodic signal from $cl$

let $ac(v) = ac(v) \times d$

if $ac(v) > F$

send activation messages to all nodes $v'$ in which $(v, v') \in E_S \cup E_M$

**Figure 13. Activation message processing algorithm in ASN**

**Figure 14. Activation spreading algorithm in ASN**

### 5.2.2 From Hierarchical Task Network to Activation Spreading Network

We define an activation spreading network as an acyclic graph to simplify the design by avoiding recurrent ASNs. This is in line with our simplifying assumption about not having recursions in the HTN formalism that we adapted for our work. Each task in HTN can be seen as a potential intention. Tasks in an HTN form a hierarchy according to the definition of methods. Intuitively, this means that intentions can be sub-goals for a higher
level intention. Different methods of the same task describe different ways of achieving the goal. We now describe how to instantiate an ASN from the domain knowledge in the form of HTN. Figure 15 shows the conversion algorithm.

**HTN to ASN Conversion Procedure**

for every operator \( R(v_1, ...v_n) \) in HTN

for every substitution \((val_1, ...val_n)\) in the domain of \( D(v_1) \times ... D(v_n) \)

add node \( R(val_1, ...val_n) \) to the ASN

for every compound task \( T(x_1, ..., x_m) \) in HTN

for every substitution \((val_1, ...val_m)\) of \((x_1, ..., x_m)\)

add node \( T(val_1, ...val_m) \) to the ASN if not already present

for every method \( M \in methods(T) \) and \((val_1, ...val_m) \vdash pre(M)\)

add node \( T^M(val_1, ...val_m) \) to the ASN

add a max edge from \( T^M(val_1, ...val_m) \) to \( T(val_1, ...val_m) \)

for every task \( T'(x'_1, ..., x'_m) \) in \( subtasks(M) \)

let \((val'_1, ...val'_m)\) be a substitution for \((x'_1, ..., x'_m)\)

in agreement with \((val_1, ...val_m)\)

if \( T'(val'_1, ...val'_m) \) is compound then

add \( T'(val'_1, ...val'_m) \) if not already present

add a sum edge from \( T'(val'_1, ...val'_m) \) to \( T^M(val_1, ...val_m) \) with \( \frac{1}{\mid subtasks(M) \mid} \) as weight

else if \( T'(val'_1, ...val'_m) \) is primitive then

let \( R \) be the operator corresponding to \( T' \)

add a sum edge from \( R(val'_1, ...val'_m) \) to \( T^M(val_1, ...val_m) \) with \( \frac{1}{\mid subtasks(M) \mid} \) as weight

Figure 15. HTN to ASN conversion algorithm
The conversion algorithm first adds all possible instantiations of operators in the HTN as nodes \( R(val_1, \ldots, val_n) \) in the ASN. These nodes will be the leaves of the hierarchical structure of the obtained ASN. With a similar procedure, we also add new nodes \( T(val_1, \ldots, val_m) \) to the network for each unique instantiation of compound tasks. Each instance of a compound task can be realized in different ways represented by a set of methods. To capture this property of HTNs in our network, we add additional nodes such as \( T^M(val_1, \ldots, val_m) \) for each method \( M \) and connect these nodes to the parent node \( T(val_1, \ldots, val_m) \) with max edges. With this configuration, the activation value of node \( T(val_1, \ldots, val_m) \) would be the maximum activation value among all of its methods.

Activation values of nodes in the ASN provide a comparative measure for the likelihood of their corresponding tasks happening in the scene and are used for inference. Sum edges are not a suitable choice for connecting method nodes to task nodes because several methods with low activations should not accumulate a high activation in the task node, since the likelihood of a high-level task happening in the scene is only as high as the maximum likelihood of its methods.

Any method \( M \) in our HTN breaks down a high-level task into lower-level tasks (either compound or primitive). This is captured in our ASN by connecting nodes of lower-level tasks to their parent node corresponding to the method, which in turn is connected to the high-level task. A method should have a higher likelihood if a larger number of its subtasks have higher activation values. For instance, a method with only a single subtask node with activation value greater than zero is less probable than another method with two or more subtasks with the same activation values. This is why we chose to use sum
edges to connect subtasks to their parent method. The edge weights are \( \frac{1}{|\text{subtasks}(M)|} \). This is a normalization factor in order to make the activation values in method nodes comparable to each other, regardless of their subtask size. A sample ASN created with this conversion algorithm is shown in Figure 22(A). It is important to note that the algorithm in Figure 15 does not use the partial order relation in the task networks. At this stage, we simply ignore the ordering constraints between tasks and cannot distinguish between two methods (perhaps not even for the same high-level task) that have exactly the same subtasks but in different order. We will extend our ASN approach to handle partial order constraints in Section 5.2.5.

5.2.3 Intent Recognition in Activation Spreading Networks

As explained in Section 5.2.2, the activation values are a comparative measure for selecting the most probable intention based on the observed evidence. The hierarchical structure of tasks in HTN suggests that at any given time, a subject is actively pursuing a set of intentions that are in agreement with each other but are at different levels in the hierarchy. Lower-level tasks that normally correspond to short time span activities that are a part of a larger and longer activity. This hierarchical structure is preserved in the process of converting HTN to ASN in the form of connected nodes. We use this hierarchical structure and the activation values of the nodes in the network to robustly detect the intention of completing a set of tasks but at different levels in the hierarchy. More precisely, we start the search from the highest level nodes corresponding to the highest level intentions and choose the one with the largest activation value above a threshold. If none of the nodes have activation values greater than the threshold, then the
system detects the idle state for the subject. We then continue our search by only considering the children of that node. The highest activation value is chosen at each stage iteratively until we reach the lowest level, containing only operators. To disregard very low activation values, any node with activation below a threshold cannot be chosen even if it has the highest activation. The intent recognition algorithm is presented in Figure 16.

**Intent Recognition Procedure**

| let s = {v|v is a node in ASN and has no outgoing arrow} |
| repeat until s = ∅ |
| let \( v_{\text{max}} = \text{argmax}_{v \in s} ac(v) \) |
| add \( v_{\text{max}} \) to the set of recognized intentions if ac(v) > F |
| s = {v|v is a node in ASN and has outgoing arrow to \( v_{\text{max}} \)} |

Figure 16. Intent recognition algorithm in ASN

It is important to note that activation values of nodes are only comparable if they are on the same level. Nodes in higher levels usually have smaller activation values, since they only receive activation values from lower-level nodes and edges have weights less than one, which reduces the activation values. This is why we only compare nodes at the same level in each stage of the recognition process. It is also possible that the node with the highest activation value in a lower level conflicts with the node with the highest activation in a higher level. The lower level node might belong to a task that does not contribute to the higher level task with the largest activation value. To have a coherent recognition of intentions on different levels, at each stage of the process we limit our search space to nodes that are the children of the selected higher level task.

Since the network is acyclic, an external stimulus (i.e., observation of a low-level activity) starts the activation spreading process in the network. Low-level activities
correspond to operator nodes in the network. These nodes are leaves in the network, and they can propagate activation values up in the hierarchy, but no other nodes can send activation messages to them. We use simple formulas to compute the activation values for the operator nodes, based on the features extracted by the video parser. This part of the system will be defined in Section 6.2.

5.2.4 Context-Based Intent Recognition

The precondition properties of operators and methods in HTN allow us to choose suitable methods to reduce a task network to a fully formed plan. These preconditions describe a context or situation in which that operator or method is suitable for achieving a goal. Planning approaches usually need to know about preconditions to successfully develop a plan suitable for the current circumstances by choosing suitable methods and operators. Similarly for intent recognition, we also face the problem of choosing the hypothesis that best describes the observed evidence. This suggests that having some information about the actual context of the observed scene can help in intent recognition, by analyzing what method or task is more probable for the subject to undertake, given the known circumstances. Unlike preconditions in planning, which model the required conditions, contextual information for intent recognition in our framework works as a favoring mechanism that makes some tasks more probable, and others less probable.

In order to incorporate contextual information in our intent recognition system, we modify the definition of ASN to include another type of nodes to represent contextual information and two special types of edge to connect contextual information to the relevant task nodes in the network. The formal definition of a contextual ASN (CASN) is
as follows:

**Contextual Activation Spreading Network:** A CASN is an ASN with an additional set of:

- Context nodes $V_C$ representing different contextual information. A context node $v_c \in V_C$ has an activation value $ac(v_c)$ representing the level of certainty for that context.

- Positive context edges $E_{C+}$ connecting nodes in $V_C$ to nodes in $V$.

- Negative context edges $E_{C-}$ connecting nodes in $V_C$ to nodes in $V$.

Nodes in $V_C$ represent contextual information and their activation values represent the level of certainty about that information. It is important to note that the activation values do not represent probabilities and should not be interpreted as such. Nodes in $V_C$ do not have any ingoing edges of any type and cannot send activation messages to any other nodes. Nodes in $V_C$ cannot have activation values greater than 1. Nodes in $V_C$ do not decay by clock ticks. Upon receiving activation messages, the receiving node $v$ would update its activation value by first applying the procedure in Figure 13 and then multiplying the activation value by $\left( \sum_{(v_c,v) \in E_{C+}} \frac{1+ac(v_c)}{|[(v_c,v)\in E_{C+}]} - \sum_{(v_c,v) \in E_{C-}} \frac{1-ac(v_c)}{|[(v_c,v)\in E_{C-}]} \right)$.

Figure 17 shows the algorithm for processing activation messages in the CASN.

Edges in $E_{C+}$ and $E_{C-}$ show the positive and negative effect of contextual information on the tasks. Having additional contextual information about the subject or the environment being observed by the system should not increase or decrease the activation value of any tasks, unless we are observing some activities in the scene. In other words, we should not detect any intentions when no activity is being observed, even if all contextual
information is in favor of a particular activity. That is why we chose the above formula to update activation values. If no contextual information is available, then the activation values of nodes in $V_C$ are zero and the multiplication factor is one. If contextual information in favor of a task is stronger than contextual information against a task, then the multiplication factor would be greater than 1, and it will be less than 1 otherwise. A sample of CASN is shown in Figure 22(B).

<table>
<thead>
<tr>
<th>Activation Processing Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>let $v$ be the node receiving activation message from $s$</td>
</tr>
<tr>
<td>let $S_{max} = {n</td>
</tr>
<tr>
<td>if $S_{max} \neq \emptyset$</td>
</tr>
<tr>
<td>$ac(v) = \max_{v' \in S_{max}} ac(v')$</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>$ac(v) = ac(v) + ac(s) \times w((s,v))$</td>
</tr>
<tr>
<td>$ac(v) = ac(v) \times \left(\sum_{(v_c,v) \in E_{C^+}} \frac{1+ac(v_c)}{</td>
</tr>
</tbody>
</table>

Figure 17. Activation message processing algorithm in CASN. The text in blue represents modifications to the original procedure for ASN presented in Figure 13

5.2.5 Partial-Order Modeling in Activation Spreading Networks

As previously discussed in Section 5.2.2, the ASN and CASN cannot model partial-order constraints in the hierarchical task network formalism, and the intent recognition procedure in ASN and CASN cannot distinguish between methods of two different tasks that are only different in their partial-order constraints. We now propose an extension to ASN in order to model partial-order constraints by allowing edges to receive activation messages and defining a special type of edge (ordering edges) to connect nodes to other edges in the network.
Partial-Order Contextual Activation Spreading Network: A Partial-Order Contextual Activation Spreading Network (POCASN) is a CASN with an additional set of:

- Ordering edges $E_{PO}$, connecting nodes in $V$ to edges in $E_S$. Every sum edge $e \in E_S$ has an activation value $ac(e)$ in addition to its weight $w(e)$ and can receive activation messages from ordering edges in $E_{PO}$.

- Min edges $E_m$, connecting nodes in $V$ to each other. A min edge is another type of edge in which an activation messages spreads in the network and the receiving node processes it by updating its activation value with a minimum value selection.

Upon receiving an activation message from node $n$, a sum edge $e \in E_S$ updates its activation value with $ac(e) = max\{ac(e) + ac(n), 1 - w(e)\}$. With every clock tick, activation values of all sum edges like $e \in E_S$ decay according to: $ac(e) = ac(e) \times d$.

Upon receiving an activation message, a node updates its activation value by summing the activation message multiplied by the edge weight plus the edge activation value, with its own activation value if the message was received via a sum edge. A node would update its activation value by choosing the minimum activation message on all ingoing min edges. Figure 18 shows the algorithm of processing activation messages in the POCASN.
**Activation Processing Procedure**

let $v$ be the node receiving activation message from $s$

let $S_{max} = \{n|(n,v) \in E_M \text{ and } n \text{ sent message to } v \text{ in recent clock signal}\}$

let $S_{min} = \{n|(n,v) \in E_m \text{ and } n \text{ sent message to } v \text{ in recent clock signal}\}$

if $S_{max} \neq \emptyset$

$$ac(v) = \max_{v' \in S_{max}} ac(v')$$

else if $S_{min} \neq \emptyset$

$$ac(v) = \min_{v' \in S_{min}} ac(v')$$

else

$$ac(v) = ac(v) + ac(s) \times \left( w((s,v)) + ac((s,v)) \right)$$

$$ac(v) = ac(v) \times \left( \sum_{(v_c,v) \in E_{C^+}} \frac{1+ac(v_c)}{|(v_c,v)\in E_{C^+}|} - \sum_{(v_c,v) \in E_{C^-}} \frac{1-ac(v_c)}{|(v_c,v)\in E_{C^-}|} \right)$$

Figure 18. Activation message processing procedure in POCASN. The text in blue represents modifications to the original procedure for CASN presented in Figure 17

The main idea behind POCASN is to allow nodes to strengthen edges that connect subsequent tasks (in the task partial ordering) to the common parent node, by sending activation messages to those edges. If a task receives high activation values (showing a detection of that task), it cannot significantly affect its parent node, unless the preceding task in the partial-order has been detected and strengthened the edge connecting that task to its parent. The procedure of activation spreading in POCASN suppresses spreading of activation among nodes, if the order of observed tasks is not in agreement with the partial order constraints. The edges in $E_{PO}$ connect task nodes to edges from the subsequent task nodes in the partial order to their common parent. The reason behind defining min edges is to model tasks that have no ordering constraints with respect to each other. For such tasks there is no requirement on their order of execution. However, these tasks can all be
the immediate prerequisite of another set of tasks. All the prerequisite tasks should have
been detected (have high activation values) to strengthen edges on the subsequent set of
tasks. We use min edges to connect all unrelated nodes to an extra node which in turn, is
connected to the edges of the subsequent set of tasks. This ensures that all prerequisite
tasks are detected before expecting to observe the subsequent tasks in the task network.
We limit the activation value of edges in $E_S$ to $1 - w(e)$, mainly because any activation
values more than that would amplify the activation value of the sender node by
multiplying it with a number greater than 1, which is not desirable. Figure 22(C) shows a
sample POCASN used in our work.
We now introduce the algorithm to convert HTN to POCASN by modifying the original
conversion algorithm presented in Figure 15. The new conversion algorithm that creates a
POCASN from an HTN is shown in Figure 19.
The conversion procedure is similar to the original algorithm in Figure 15 with some
modifications. Recall that partial-order constraints in HTN are a part of a task network
which itself is the body of a method for accomplishing a compound task. While
processing different methods for a compound task, we first need to topologically sort the
set of tasks in the method body, according to their partial-order constraints. While
processing each task $T'$ in the body of a method, we first find a chain containing that
particular task to find out all the immediate prerequisite tasks $pre(T')$. Then we add a
dummy node to the network for collecting activation values of all tasks in the $pre(T')$ via
min edges. This dummy node in turn spreads activation to the sum edge from $T'$ to the
method, in order to model sequencing.
HTN to POCASN Conversion Procedure

for every operator $R(v_1, ..., v_n)$ in HTN
  for every substitution $(val_1, ... val_n)$ in the domain of $D(v_1) \times ... D(v_n)$
    add node $R(val_1, ..., val_n)$ to the ASN

for every compound task $T(x_1, ..., x_m)$ in HTN
  for every substitution $(val_1, ... val_m)$ of $(x_1, ..., x_m)$
    add node $T(val_1, ..., val_m)$ to the ASN if not already present

  for every method $M \in \text{methods}(T)$ and $(val_1, ..., val_m) \vdash \text{pre}(M)$
    add node $T^M(val_1, ..., val_m)$ to the ASN

  add a max edge from $T^M(val_1, ..., val_m)$ to $T(val_1, ..., val_m)$

let $\text{sorted}(M)$ be an topological sort of $\text{subtasks}(M)$

for every task $T'(x'_1, ..., x'_m)$ in $\text{sorted}(M)$
  let $(val'_1, ..., val'_m)$ be a substitution for $(x'_1, ..., x'_m)$
    in agreement with $(val_1, ..., val_m)$

let $\text{chain}(T')$ be the longest chain containing $T'$ and
$\text{pos}(T')$ be the position of $T'$ in $\text{chain}(T')$

let $\text{pre}(T') = \{ \text{task}| \text{task} < T' \land \exists \text{task}' : (\text{task} < \text{task}' \land \text{task}' < T') \}$

add node $\text{pre}(T')$ to ASN if not exists and $\text{pre}(T') \neq \emptyset$

add min edges from every node $n \in \text{pre}(T')$ to node $\text{pre}(T')$

if $T'(val'_1, ..., val'_m)$ is compound then
  add node $T'(val'_1, ..., val'_m)$ if not already present
  add a sum edge from $T'(val'_1, ..., val'_m)$ to $T^M(val_1, ..., val_m)$ with $\frac{1}{\text{subtasks}(M) \times \text{pos}(T')}$ as weight

add ordering edge from $\text{pre}(T')$ to edge $(T', T^M)$ if $\text{pre}(T') \neq \emptyset$

else if $T'(val'_1, ..., val'_m)$ is primitive then
  let $R$ be the operator corresponding to $T'$
  add a sum edge from $R(val'_1, ..., val'_m)$ to $T^M(val_1, ..., val_m)$ with $\frac{1}{\text{subtasks}(M) \times \text{pos}(T')}$ as weight

add ordering edge from $\text{pre}(T')$ to edge $(R, T^M)$ if $\text{pre}(T') \neq \emptyset$

Figure 19. HTN to POCASN conversion algorithm. The text in blue represents modifications to the original algorithm for ASN presented in Figure 15.
Weights of sum edges in POCASN represent a minimum effective value for the edges, since they can receive activation values from ordering edges to make them stronger. Unlike the original ASN in which sum edges would connect task nodes to method nodes in the network with a shared normalization weight of $\frac{1}{|\text{subtasks}(M)|}$, in POCASN we need to assign smaller weights to outgoing edges from tasks that come after other tasks in the chain. This is because observing a task that comes after another task in the chain should not be sufficient for detecting the intention by itself. This observation is only important, if we previously detected preceding tasks (high activation values for preceding tasks). Weights of sum edges should be normalized by the size of the subtasks in order for different methods on the same level to have comparable activation values. $\frac{1}{|\text{subtasks}(M)| \times \text{pos}(T')}$ was chosen as the weight of sum edge connecting node $T'$ to its parent. $\frac{1}{|\text{subtasks}(M)|}$ is the normalization factor for a method and $\frac{1}{\text{pos}(T')}$ is the effect of ordering of task $T'$ among other tasks in the network.
6. Experimental Results for High-Level Intent Recognition

In this section we describe our implemented system and introduce the experiment domain, the HTN domain knowledge we used for experiments, the contextual information incorporated in the system, the resulting ASN, CASN and POCASN and the experimental results in three different tested scenarios.

6.1 Experiment Domain

The domain chosen for the experiments consists of 16 high-level daily activities and 2 different low-level activities. All these activities are defined upon 8 different objects. *Book, cup, bowl, kettle, lettuce, tea bag, bottle and instant coffee* are the objects of interest in our intent recognition system. Figure 20 shows our training images for the video parser. We developed our real-time intent recognition system in C++ using Microsoft Kinect SDK and OpenCV as our two main software tools. The system captures live feed from a Kinect camera and outputs the results of intent recognition in the form of activation values for the important task nodes in our network. For a more natural interaction, we also used a NAO robot (by Aldebaran Robotics) to provide audio feedback to the user to inform him/her about the detected intention. The robot controller is designed to interpret detected intentions from our intent recognition system and make the robot behave appropriately by having a real-time interaction with the user. Figure 21 shows a captured screenshot of the developed software.
Our low-level activities (corresponding to operators in the HTN formalism) are grabbing and moving. The high-level activities (corresponding to compound tasks in the HTN formalism) are drink [what] from [where] in which [what]={tea, water, coffee} and [where]={cup, bottle}, eat from bowl, pour from [src] to [dest] in which [src]={kettle, bottle} and [dest]={cup, bowl, kettle}, make salad, organize desk, make tea, make coffee,
read book and eat lettuce. The set of all contextual information about the scene contains 8 different items: person has exam soon, person has meeting soon, it is morning, it is night, person is thirsty, person is hungry, weather is cold, and weather is hot. We use two different states for contextual information, and we do not use the full spectrum of activation values of contextual nodes. We either know a particular context is true, or it is either false or unknown. If it is known, then the activation value is set to 0.2, otherwise it is set to 0. The effect of this contextual information on the compound tasks is shown in Table V.

<table>
<thead>
<tr>
<th>Compound Task</th>
<th>Positive Context</th>
<th>Negative Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>drink tea from cup</td>
<td>weather is cold, it is morning</td>
<td>weather is hot</td>
</tr>
<tr>
<td>drink water from cup</td>
<td>person is thirsty, weather is hot</td>
<td>-</td>
</tr>
<tr>
<td>drink coffee from cup</td>
<td>it is morning, weather is cold</td>
<td>weather is hot, it is night</td>
</tr>
<tr>
<td>drink water from bottle</td>
<td>person is thirsty, weather is hot</td>
<td>-</td>
</tr>
<tr>
<td>eat from bowl</td>
<td>person is hungry</td>
<td>-</td>
</tr>
<tr>
<td>pour from kettle to cup</td>
<td>weather is cold, it is morning</td>
<td>-</td>
</tr>
<tr>
<td>pour from kettle to cup</td>
<td>person is thirsty</td>
<td>-</td>
</tr>
<tr>
<td>make salad</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>organize desk</td>
<td>person has meeting soon</td>
<td>-</td>
</tr>
<tr>
<td>make tea</td>
<td>weather is cold, it is morning</td>
<td>weather is hot</td>
</tr>
<tr>
<td>make coffee</td>
<td>weather is cold, it is morning</td>
<td>weather is hot, it is night</td>
</tr>
<tr>
<td>pour from bottle to kettle</td>
<td>weather is cold</td>
<td>-</td>
</tr>
<tr>
<td>pour from bottle to cup</td>
<td>person is thirsty</td>
<td>-</td>
</tr>
<tr>
<td>pour from bottle to bowl</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>read book</td>
<td>person has exam soon</td>
<td>-</td>
</tr>
<tr>
<td>eat lettuce</td>
<td>person is hungry</td>
<td>-</td>
</tr>
</tbody>
</table>
Table VI shows the HTN-based representation of high-level tasks and their decomposition into lower-level tasks and operators. The operator *grab* has a single parameter with the domain containing all possible objects, and the operator *move* has two parameters: the first parameter is the object with the domain containing all objects; the second parameter is the destination and it can be an object, a special value *self* (the person is moving the object towards him/herself), or another special value *not known*. Compound tasks do not have parameters for instantiation and have a single method of decomposition.

<table>
<thead>
<tr>
<th>Compound Task</th>
<th>Method Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>drink tea from cup</td>
<td>make tea &lt; grab cup &lt; move cup to self</td>
</tr>
<tr>
<td>drink water from cup</td>
<td>pour from bottle to cup &lt; grab cup &lt; move cup to self</td>
</tr>
<tr>
<td>drink coffee from cup</td>
<td>make coffee &lt; grab cup &lt; move cup to self</td>
</tr>
<tr>
<td>drink water from bottle</td>
<td>grab bottle &lt; move bottle to self</td>
</tr>
<tr>
<td>eat from bowl</td>
<td>grab bowl &lt; move bowl to self</td>
</tr>
<tr>
<td>pour from kettle to cup</td>
<td>grab kettle &lt; move kettle to cup</td>
</tr>
<tr>
<td>pour from kettle to bowl</td>
<td>grab kettle &lt; move kettle to bowl</td>
</tr>
<tr>
<td>make salad</td>
<td>grab lettuce &lt; move lettuce to bowl</td>
</tr>
<tr>
<td>organize desk</td>
<td>grab object x &lt; move object x to object y (for all possible x, y where x≠y)</td>
</tr>
<tr>
<td>make tea</td>
<td>pour from kettle to cup, grab tea bag &lt; move tea bag to cup</td>
</tr>
<tr>
<td>make coffee</td>
<td>pour from kettle to cup, grab coffee &lt; move coffee to cup</td>
</tr>
<tr>
<td>pour from bottle to kettle</td>
<td>grab bottle &lt; move bottle to kettle</td>
</tr>
<tr>
<td>pour from bottle to cup</td>
<td>grab bottle &lt; move bottle to cup</td>
</tr>
<tr>
<td>pour from bottle to bowl</td>
<td>grab bottle &lt; move bottle to bowl</td>
</tr>
<tr>
<td>read book</td>
<td>grab book &lt; move book to self</td>
</tr>
<tr>
<td>eat lettuce</td>
<td>grab lettuce &lt; move lettuce to self</td>
</tr>
</tbody>
</table>
6.2 Activation Spreading Networks

The HTN introduced in Section 6.1 can be converted to an ASN. After instantiating all operators, we have 80 operator nodes in the network. Figure 22(C) shows a portion of the resulting POCASN without all the operators and context nodes for simplicity. The network contains 80 different operator nodes accounting for all possible instantiations of grab and move. The clock in our ASN increments with receiving a new frame from the camera, therefore the clock frequency is equal to the frame rate (25 to 30 fps for our configuration). We chose 0.98 as the decay factor in our network and the firing threshold is set to zero for simplicity.

To determine the activation value of operators, we used simple formulas that can be directly computed from the features extracted by the video parser. The activation value of move nodes is computed with equation (1) and the activation value of grab nodes is computed with equation (2).

\[
ac(R) = \min \left\{ \frac{-\text{distance change between object and destination}}{\text{actual distance between object and destination}} \right\} \quad (1)
\]

\[
ac(R) = \min \left\{ \frac{-\text{distance change between hands and the object}}{\text{actual distance between hands and the object}} \right\} \quad (2)
\]
Figure 22. (A) A portion of the ASN generated from the HTN. (B) The same network but in CASN form. (C) The same network in POCASN form. Red lines are max edges, green lines are ordering edges, black lines are min edges, blue lines represent sum edges, grey lines show positive context edges, and orange lines represent negative context edges. Red boxes represent method nodes, grey boxes show context nodes and green boxes represent dummy nodes used to connect ordering edges to sum edges.
6.3 Scenarios

For experimental evaluation we performed three different scenarios, during which the intent recognition system was running and the robot in the scene was interacting with the subject at run time. These scenes are also recorded and can later be fed into our system as a live stream to simulate the situation where the program is actually processing frames as they stream in. This is useful for repeating the same experiments given different contextual situations. The first scenario (eating) consists of a person eating salad (lettuce) and then drinking water from a bottle. It is 60 seconds long and *lettuce, bowl, bottle, tea bag* and *cup* are present in the scene. The second scenario (reading) consists of a person reading a book, then making tea and finally drinking tea from a cup. *Tea bag, coffee, book, kettle* and *cup* are visible in this scene and the video is 65 seconds long. In the final scenario (drinking) the subject makes tea and drinks tea from the cup. *Kettle, tea bag, cup* and *coffee* are present in the scene. The drinking scenario is 36 seconds long. We manually segmented the videos into partitions for different high-level intentions and used that as the ground truth. We experimented with ASN, CASN and POCASN to see how well they can handle these three scenarios. We also investigated the effect of considering contextual information and partial-order constraints on the system performance. Section 6.4 provides a discussion of the results obtained during the experimental evaluation.

6.4 Experiments

A relevant performance metric for intent recognition is how early the system is able to detect an intention reliably. Recall that for intent recognition we compare activation values of nodes at the same level in the hierarchical structure of the network, in order to
choose the node with largest activation value as the recognized intention. The difference between the highest and the second highest activation values represents the level of confidence for recognition. For a quantitative analysis of the intent recognition with ASNs, we used two metrics:

**Early detection rate:** $\frac{t_i^*}{T_i}$ where $T_i$ is the total runtime of the segment for intention $i$ in a scenario, and $t_i^*$ is the earliest time (from the start of the segment) from which the correct intention was recognized consistently until the end of segment for intention $i$.

**Confidence of detection:** $\frac{ac(i)}{ac(max-1)}$ where $ac(i)$ is the activation value of the correct intention $i$ at any given time and $ac(max-1)$ is the second highest activation value of the nodes in the same level as $i$.

For effective intent recognition we want the early detection rate to be close to 0, which means that the system was able to detect that intention immediately. An average of early detection rate for a scenario is simply computed over all intentions present in that scenario. For the confidence of detection, larger values show a better (more confident) recognition of intentions. Any values greater than 1 show a correct recognition at that particular time. An average confidence of detection for an intention in a scenario is computed over all time-steps in a scenario in the intention’s ground truth segment.

### 6.4.1 Eating Scenario

Figure 23 shows the activation values of high-level (compound) tasks in our CASN. Figure 24 shows the same results but with ASN. In the full context version, we assumed the system knows the following: *it is hot, it is night, the person is hungry* and *person is thirsty*. We annotated the graph with how the robot responded during each case. We also
show the ground truth segmentation of intentions.

A comparison of Figure 23 with Figure 24 shows how CASN can distinguish between similar activities better than the ASN. To highlight this improvement, Figure 25 and Figure 26 show a subset of related tasks for these two networks. These activities are at the same level in the structure of the network, therefore their activation values are comparable. In Figure 25, we can see a significant disambiguation of similar activities including *drinking water from bottle, pouring from bottle to cup* and *pouring from bottle to bowl*. The ASN was not able to choose the correct intention among these very similar tasks. Figure 26 shows how the activation values of these three activities are very close, which makes a reasonable detection impossible. The early detection results and average confidence of detections for this scenario in CASN and ASN are shown in Table VII. CASN improves ASN by correctly detecting *drink water from bottle* with 51.12% early detection rate, compared to no detection in ASN. The confidence of detection is also improved for *eat from bowl* and *drink water from bottle* tasks. However ASN worked slightly better for *make salad*, because in CASN contextual nodes are strengthening the *eat from bowl* task, which has the second highest activation value in the *make salad* segment of this scenario.
Figure 23. Activation values for the *eating* scenario with CASN

Figure 24. Activation values for the *eating* scenario with ASN
Figure 25. A subset of activities for the *eating* scenario with CASN

Figure 26. A subset of activities for the *eating* scenario with ASN
Table VII. Early detection rates and average confidence of detections for eating scenario

<table>
<thead>
<tr>
<th>Intention</th>
<th>Early Detection Rate</th>
<th>Average Confidence of Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>make salad</td>
<td>62.25%</td>
<td>1.02</td>
</tr>
<tr>
<td>eat from bowl</td>
<td>4.85%</td>
<td>1.60</td>
</tr>
<tr>
<td>drink water from bottle</td>
<td>51.12%</td>
<td>0.80</td>
</tr>
<tr>
<td>ASN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>make salad</td>
<td>57.84%</td>
<td>1.08</td>
</tr>
<tr>
<td>eat from bowl</td>
<td>5.42%</td>
<td>1.39</td>
</tr>
<tr>
<td>drink water from bottle</td>
<td>no detection</td>
<td>0.68</td>
</tr>
</tbody>
</table>

6.4.2 Reading Scenario

Figure 27 shows the activation values of high-level (compound) tasks in our CASN. Figure 28 shows the same results but with ASN. In the full context version, we assumed the system knows the following: person has exam soon, it is morning and weather is cold.

A comparison of Figure 27 with Figure 28 shows that nodes in CASN accumulate higher activation values. This is expected since contextual information can have positive effects on the activation values of some task nodes in the network. In CASN, we would have no activation values spreading from context nodes. This is visible in Figure 27, since we have no activation values at the beginning of this graph. Context would only amplify the effect of activation spreading in the network in favor of some related tasks. Figure 29 and Figure 30 show a subset of related tasks for these two networks. As shown in these figures, ASN is not able to detect the make tea intention. The early detection results and average confidence of detections for this scenario in CASN and ASN are shown in Table VIII. It is clear that CASN is performing better than ASN for this scenario by detecting make tea with 58.80% early detection rate compared to no reliable detection of this
intention in ASN. No detection for this task does not imply that it was not recognized in any time-step. This means that the network could not correctly recognize that intention at the last time-step of its ground truth time segment. However, *make tea* had the highest activation values for the most parts of the ground truth. An interesting property of our ASN is shown in this experiment. Although the system was not able to detect *make tea* reliably (not until the end of ground truth segment) it could correctly recognize the next task *drink tea from cup*. This shows that our ASN-based intent recognition is able to recover from detection errors in the previous time-steps. The use of context also improved the early detection rate for *drink tea from cup* by 16%.

Figure 27. Activation values for the *reading* scenario with CASN
Figure 28. Activation values for the *reading* scenario with ASN

Figure 29. A subset of activities for the *reading* scenario with CASN
Figure 30. A subset of activities for the *reading* scenario with ASN

Table VIII. Early detection rates and average confidence of detections for *reading* scenario

<table>
<thead>
<tr>
<th>Intention</th>
<th>Early Detection Rate</th>
<th>Average Confidence of Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CASN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>read a book</em></td>
<td>21.22%</td>
<td>1.23</td>
</tr>
<tr>
<td><em>make tea</em></td>
<td>58.80%</td>
<td>1.31</td>
</tr>
<tr>
<td><em>drink tea from cup</em></td>
<td>22.66%</td>
<td>1.24</td>
</tr>
<tr>
<td><strong>ASN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>read a book</em></td>
<td>23.58%</td>
<td>1.24</td>
</tr>
<tr>
<td><em>make tea</em></td>
<td>no detection</td>
<td>1.18</td>
</tr>
<tr>
<td><em>drink tea from cup</em></td>
<td>38.66%</td>
<td>1.23</td>
</tr>
</tbody>
</table>

6.4.3 Drinking Scenario

Figure 31 shows the activation values of high-level (compound) tasks in our POCASN. Figure 32 shows the same results but with ASN. The *drinking* scenario is designed to evaluate the performance of POCASN and its comparison with the original ASN. No contextual information is used in POCASN, in order to see the effect of partial-order modeling for intent recognition. The first major difference between POCASN and the
original ASN is that in POCASN, activation values are much higher. This is because of spreading activation values to edges in addition to nodes. A node corresponding to a previous task would strengthen the edges of the next task, and this results in a higher spreading of activation to the parent method node. *Drink tea from cup* and *drink coffee from cup* are similar tasks in our network, with only a difference on *grabbing* and *moving coffee* instead of *moving tea*. In the *reading* scenario we could detect this activity with the help of contextual information, which here is absent. POCASN is able to detect this intention reliably, as shown in Figure 33. A similar graph for ASN in Figure 34 shows that ASN cannot disambiguate between these two tasks. Table IX shows detection rates and average confidence of detection for the *drinking* scenario, for both POCASN and ASN. In POCASN, the network is expecting to observe the next correct task – the edges connecting the expected task to its method node are already strengthened, which greatly helps decrease the early detection rate. Even for *drink tea from cup* the accumulated activation values received from *make tea* are enough for a detection of that intention. The 0% early detection rate for *drink tea from cup* in POCASN illustrates this situation.
Figure 31. Activation values for the *drinking* scenario with POCASN

Figure 32. Activation values for the *drinking* scenario with ASN
Figure 33. A subset of activities for the *drinking* scenario with POCASN

Figure 34. A subset of activities for the *drinking* scenario with ASN
Table IX. Early detection rates and average confidence of detections for *drinking* scenario

<table>
<thead>
<tr>
<th>Intention</th>
<th>Early Detection Rate</th>
<th>Average Confidence of Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>POCASN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>make tea</td>
<td>31.84%</td>
<td>1.63</td>
</tr>
<tr>
<td>drink tea from cup</td>
<td>0%</td>
<td>1.50</td>
</tr>
<tr>
<td><strong>ASN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>make tea</td>
<td>76.01%</td>
<td>1.09</td>
</tr>
<tr>
<td>drink tea from cup</td>
<td>no detection</td>
<td>1.06</td>
</tr>
</tbody>
</table>
7. Conclusion and Future Work

In this thesis we described, implemented and evaluated an intent recognition system by utilizing Activation Spreading Networks (ASN). In the first section of this thesis we introduced a novel approach to multi-agent intent recognition for the real-time domain of naval ships. The proposed approach consists of two components - for detecting low-level single actor intentions and for recognizing high-level multi-agent (joint) intentions. The low-level intent recognition system is based on the idea of perspective taking for intent recognition, by utilizing Hidden Markov Models (HMMs) with local perspective observable states. In order to be suitable for real-time processing, this part of the system is implemented by parallelizing different parts of the recognition algorithm as CUDA kernels. We also showed that the HMM-based approach is not enough to represent and recognize joint plans and intentions, and for this purpose we proposed using Activation Spreading Networks (ASNs). We showed how ASNs can be used to encode multi-agent intentions and how the activation spreading algorithm can be modified for intent recognition. We conducted several experiments to evaluate the performance of our approach in the naval ship simulator environment. Our experimental results show that the proposed approach is able to detect multi-agent intentions reliably under different circumstances, while also being able to detect these intentions very early.

In the second part of this thesis we proposed a novel real-time vision-based intent recognition system based on ASNs. We formally defined ASNs and showed how we can create a hierarchical ASN from Hierarchical Task Networks (HTNs). We then described
an algorithm for intent recognition with ASNs by selecting the maximum activation value from sets of comparable nodes in the network. We extended this ASN formalism to handle contextual information in our network. We formally defined how to modify the original ASN to obtain a Contextual ASN (CASN) and to process this type of information, and we showed how to relate context to different tasks. Finally, we extended the ASN formalism to model partial-order constraints in the HTN to obtain a Partial-Order CASN (POCASN). We implemented all three ASN, CASN and POCASN approaches in a fully functioning system. The resulting system can process RGB-D video streams in real-time to detect, recognize and track objects, extract features from video, and recognize intentions while observing a person performing daily tasks. Our experiments showed that the system is able to efficiently and reliably recognize intentions, even before activities are finished. In our experiments we also compared ASN, CASN and POCASN and showed how CASN and POCASN can improve the performance of the original ASN.

As future work, we plan to extend our ASN-based approach in order to learn the structure of the network by analyzing a training set of observed activities. For now, our system relies on having an HTN-based description of the domain knowledge to detect intentions. It is possible to investigate how this kind of domain knowledge could be extracted by machine learning techniques. Another direction of future work is related to handling missing information, which is frequent in real-world problems due to partial observability of the environment (e.g., occlusion) or failing sensors. Furthermore, it is possible to analyze how well our approach is able to recover from errors in different modules of the system, especially the video parser.
8. References


[19] H. H. Zhuo and L. Li, "Multi-agent plan recognition with partial team traces and


[36] K. Han and M. Veloso, "Automated robot behavior recognition," in *ROBOTICS*


[44] E. Nazerfard and D. J. Cook, "Using Bayesian Networks for Daily Activity


[53] A. Gopnik, V. Slaughter, and A. Meltzoff, "Changing your views: How


