Intention from Motion

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Christopher Crick

Dissertation Director: Brian Scassellati

December 2009

Abstract

Intention from Motion

Christopher Crick

2009

I present a system which observes humans participating in various playground games and infers their goals and intentions through detecting and analyzing their spatiotemporal activity in relation to one another, and then builds a coherent narrative out of the succession of these intentional states. I show that these narratives capture a great deal of essential information about the observed social roles, types of activity and game rules by demonstrating the system's ability to correctly recognize and group together different runs of the same game, while differentiating them from other games. The system can build a coherent account of the actions it witnesses similar to the stories humans tell. Furthermore, the system can use the narratives it constructs to learn and theorize about novel observations, allowing it to guess at the rules governing the games it watches. Thus a rich and layered trove of social, intentional and cultural information can be drawn out of extremely impoverished and low-context trajectory data. I then develop this process into a novel, sophisticated intention-based control system for a mobile robot. The system observes humans participating in various playground games, infers their goals and intentions through analyzing their spatiotemporal activity in relation to itself and each other, and then builds a coherent narrative out of the succession of these intentional states. Starting from zero information about the room or the rules of the games, it learns rich relationships between players, their goals and intentions, probing uncertain situations with its own behavior. The robot is able to watch people playing various playground games, learn the roles and rules that apply to specific games, and participate in the play. The narratives it constructs capture essential information about the observed social roles and types of activity. After watching play for a short while, the system participates appropriately in the games. I demonstrate how the system copes with scenarios such as chasing, follow-the-leader and tag.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, of course, thanks for the success of this research go to Brian Scassellati, my advisor. Without his mentorship and guidance, the project would never even have been imagined, let alone pushed forward. I have learned a great deal, not only about robotics and computer and cognitive science, but about sustaining a research program, managing a laboratory, engaging in grantsmanship, and above all, teaching and learning from students in turn. Whatever future success I find will in large part be due to following Scaz's example.

I also owe gratitude to my dissertation committee, Drew McDermott, Brian Scholl and Benjamin Kuipers. Their comments during the process of preparing the dissertation were invaluable, and my admiration for their research certainly influenced the direction my own work has taken. In addition, I would like to single out a few other department faculty from whom I have learned a great deal about the academic enterprise: Dana Angluin, Paul Hudak and Steve Zucker.

A number of colleagues in the social robotics lab provided invaluable assistance in designing, testing and piloting the system and my experiments. I am greatly indebted to Kevin Gold, Fred Shic, Marek Doniec, Eli Kim, Justin Hart, Dan Leyzberg, Eleanor Avrunin, Henny Admoni and Emily Bernier.

Support for this work was provided by National Science Foundation awards #0534610 (Quantitative Measures of Social Response in Autism), #0835767 (Un-

derstanding Regulation of Visual Attention in Autism through Computational and Robotic Modeling) and CAREER award #0238334 (Social Robots and Human Social Development). I would also like to acknowledge the generous support of the DARPA Computer Science Futures II program, Microsoft, and the Sloan Foundation. This work was also made possible through a software grant from QNX Software Systems Ltd.

Finally, I would like to thank all of the people without whose love, friendship and support I would never have reached this point. I don't want to single anyone out; you know who you are. I will be ever grateful.

Chapter 1

Introduction

Every schoolchild knows that teachers have eyes in the backs of their heads. From across the playground at recess, out of the corner of an eye, they always notice when a friendly game of tag transforms into something superficially similar but much more likely to end in stained clothes, torn jeans and tears. We all have this capacity; we infer a great deal about the intentions, goals, social relationships and rules of interaction from watching people interact, even from such a distance that only gross body movement can really be seen. How else could we enjoy a football game from the upper bleachers, even though the players are nothing more than small colored blobs? We can navigate the house by a four-watt nightlight and (usually) pilot automobiles through traffic in the dark and the fog. What's more, we usually can make do with even less perceptual information – we are driven to recognize dramatic situations even when presented with extremely simple cues.

More than a half century ago, Heider and Simmel found that animated boxes on a flat white screen are enough to trigger this inference process [48]. Given such a display, we happily spin narratives of anthropomorphized black squares playing with, hiding from, bullying and swindling one another. Furthermore, a wide variety

of human observers will construct the same dramatic story out of the same ludicrously simple animation. We easily spin stories about sterile geometric shapes, assigning them intentions, personalities and goals.

This process depends on very little information from the world – so little, in fact, that we can have some hope at designing computational processes that can manipulate the manageable quantity of data to accomplish similar results. A computer or robot system able to watch groups of people perform tasks or play games and figure out their goals and the rules by which they act has a number of obvious applications: entertainment, assistive technology, monitoring and safety. In addition, I hope that the process of building systems that can perform this kind of inference with real people and in real-world situations may help illuminate and ground the psychological questions of how these capabilities develop and operate in humans.

Such capabilities are particularly enticing to study, because they represent a primitive cognitive process which can be studied using real-world data, without necessarily requiring a complete solution to the enormous and distinct problem of deriving data from the brain's sensory-processing apparatus. Because we can, with proper instrumentation, derive positional information directly from the environment without visual processing (or with only the very simple visual system described here), we can begin to construct *succinct* computational models of this early social behavior and investigate their ramifications.

Not only does the system represent a cognitive model built from very rudimentary and tractable percepts, but it also demonstrates a social aptitude that is dependent *only* upon these, not also upon years of development, experience and common sense. Of course that limits the expressive capacity of the system; even the simple playground games used in the experiments have dimensions and rules beyond what the robot can conceptualize. This is a model of how perceptual psychophysics can make

Figure 1.1: The robot-controlled toy truck

a powerful contribution to social reasoning, not a general account of social cognition itself. Still, starting from zero information about its environment, the rules of the games it encounters, or even its own physical nature, our robotic system is able to learn meaningful relationships between people, their goals and intentions, while probing uncertain situations with its own behavior.

This work represents the latest step in my efforts to model a computationally tractable piece of human social cognition and decisionmaking. Within the constraints of its conceptual framework, my robot comprises a complete functional entity, from perception to learning to social interaction to mobility. Interim results from various parts of the system described here have been reported in articles such as [15], [19] and [20].

The dissertation is organized as follows. In Chapter 2, I undertake a review of relevant literature in psychology and cognitive science. I scratch the surface of the rich trove of work being accomplished in investigating human responses to impoverished stimuli, and show how my system can help to illuminate important questions of early social development. Chapter 3 describes the means by which the system makes sense of the actions it perceives, by hypothesizing about vectors of attraction, attributing intentional states on that basis, and generating a narrative that helps it to to differentiate activities and learn about the rules of the games it watches.

Chapter 4 describes the final incarnation of the system, illustrated in Figure 1. Here, it has been given eyes and wheels in the form of a web camera and a radio-controlled toy car, as well as a sophisticated RF sensor network used in the final experiments. The chapter covers the robot's architecture and construction, and shows how the techniques developed in Chapter 3 are translated into action, rather than mere observation. Chapter 5 explains the results of a wide variety of experiments with the system, from a comparison of human and machine interpretations

of scenarios, to an analysis of the system's success in classifying and categorizing games, to an evaluation of its performance as a socially-aware robot control system. Finally, Chapter 6 contains a few parting thoughts.

The extensive appendices all represent published, peer-reviewed research which I have produced as part of my graduate work. They are included in the dissertation to demonstrate both a breadth of accomplishment and contributions which hew more closely to the traditional ambit of computer science.

Chapter 2

Intentionality, Causality and Force Dynamics: A Review

2.1 Intention recognition and social attribution

> We hope to deepen our insight into the processes of perception, whether of movement or of other human beings.
>
> *Fritz Heider and Marianne Simmel*

Heider and Simmel's original study consisted of showing a two-and-a-half-minute movie to 114 undergraduate Smith women, wherein two triangles and a circle move about in a stylized animation (a still from which is shown in Figure 2.1). The researchers asked their subjects to explain what occurred, and to describe the roles, character and emotional state of each of the shapes. They elicited markedly similar responses: for instance, 97% described the larger triangle in terms of aggression and pugnacity, while 92% explained one of the circle's movements as an attempt to hide from a fight.

Heider and Simmel explain this remarkable consistency by noting that the key to interpreting the gross physical motions of their animated shapes lies in our at-

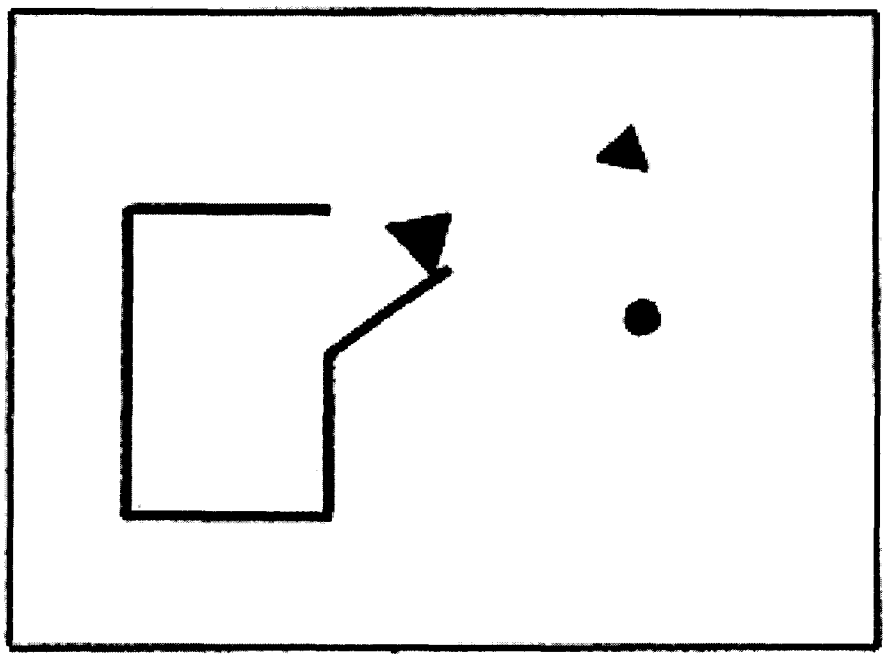

Figure 2.1: A Heider and Stimmel video still, from [48].

tribution of intentionality. "A description of movements in terms of motives again taps enviornmental layers of greater invariancy... Acts of persons have to be viewed in terms of motives in order that the succession of changes becomes a connected sequence" – in other words, a narrative.

Heider and Simmel established that low-context motion data provides sufficient information to enable people to infer complex social roles and intentional states. Thirty years later, Bassili attempted to quantify the results, applying specific variations of spatiotemporal contingencies to animations of featureless circles [5]. One of the circles in his animations moved in an algorithmically determined fashion with respect to the other: closely chasing it, changing direction in concert with it and moving in roughly the same direction, or ignoring it and moving randomly. His subjects reported far stronger impressions of intentionality and animacy in the cases where the two circles' motions were contingent upon each other.

Making sense of very low-context motion data is an important cognitive task

that we perform every day, an irrepressible instinct that develops quickly in children, around the age of 9 months [77]. Through habituation studies of infants watching animated circles moving about on blank screens, Rochat, Striano and Morgan found that "infants do appear to become increasingly sensitive to subtle changes potentially specifying intentional roles" in abstract, low-context displays. When shown a video in which one circle chased another, the older (8-10 month) children were more likely than the younger ones to notice when the circles' roles (chasing vs. fleeing) suddently switched. Much like these developmentally more-advanced children, our system detects role-switches such as these, and learns that they signal possibly meaningful events. They telegraph potential changes in intentional state, and both our system and very young children can detect and interpret them in low-context animation displays.

This low-level processing skill develops early in children, and is accompanied by the development of other social skills [21, 23, 38], such as the attribution of agency and intentionality. Csibra found that children (again, at nine months but not at six) were able to interpret goal-directed behavior appropriately, once again using abstract Heider-and-Simmel-style animations. An experimental group was shown a small circle proceeding across the visual field toward another circle, making a detour around an interposing rectangle. When the rectangle disappeared, the infants expected to see the circle move straight at the other, and were surprised when the circle moved in the same detour path that it took around the obstacle.

Csibra and his group conducted several other experiments to control for various confounds, such as the possibility that infants merely preferred the circle's straight-line approach, all else being equal. For example, by habituating the children to a circle that did not take a direct line toward a goal when one was available, they found that the differential dishabituation noted above disappeared. This, they reasoned,

was because the infants no longer believed the behavior of the circle to be rational and goal-oriented. The social cognition undertaken by these children depends entirely upon their interpretation of the relative motion of these simple figures.

Attempting to determine and to model when and how typical children develop the social skills that these animations demonstrate is one avenue of research; another emerges when one looks at what happens to people whose ability to do so is impaired. Heberlein and Adolphs [47] investigated the performance of one such person, a victim of the rare Urbach-Weithe disease, which causes calcifications in the anteromedial temporal lobes [88] and completely ravaged the subject's amygdala and adjacent anterior endorhinal cortex. Asked to describe Heider and Simmel's animation, she used language that almost completely ignored any of the social or intentional implications that control subjects invariably mentioned, treating it entirely in terms of the abstract motion of abstract shapes. However, her inability to anthropomorphize reflexively in this way was not due to a general social deficit – she was able to describe social scenes involving dogs and children perfectly well. The authors conjecture that her amygdala damage impaired a deep automatic social-attribution reflex, while leaving intact her ability to reason and deliberately retrieve declarative social knowledge.

Researchers working with individuals on the autism specturm have used Heider and Simmel animations to uncover similar social attribution deficits. Ami Klin [56] presented the animation to twenty each of autistic, Asperger's and normally developing adolescents and adults, and gave them each the opportunity to describe and answer questions about their interpretation of the shapes' activity. He found a marked difference between the tendency of the clinical groups and the controls to attribute agency and intentionality to the shapes, and to situate the scenario in a social milieu. For example, a typical narrative from a normally developing adolescent:

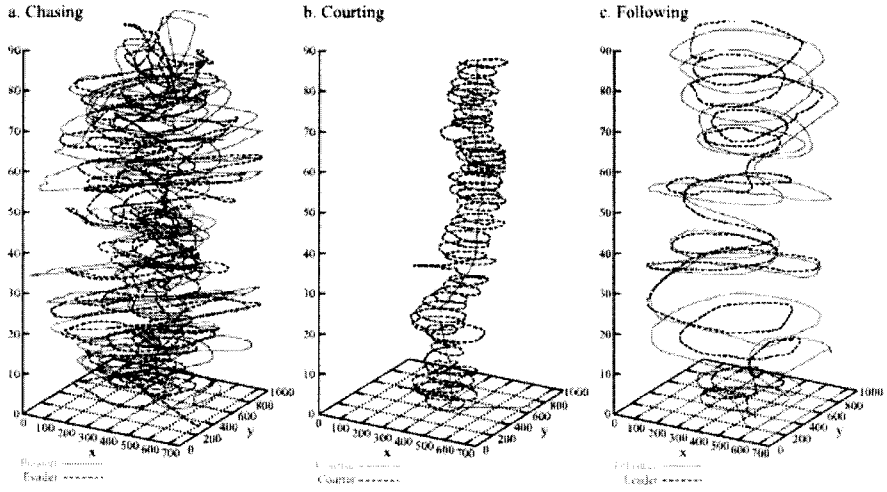

Figure 2.2: Trajectories from [4]. Compare with Figures 5.5 and 5.6.

"The smaller triangle more like stood up for himself and protected the little one. The big triangle got jealous of them, came out and started to pick on the smaller triangle." In contrast, an autistic adolescent with comparable verbal IQ: "The small triangle and the circle went around each other a few times. They were kind of oscillating around each other, maybe because of a magnetic field." In all, Klin found that the clinical groups noticed only about a quarter of the social elements usually identified by the controls.

Substantial evidence has been found for the notion that intention recognition skills develop at an early age, and is hampered by specific lesions and developmental disorders. Far less work has gone toward developing a useful computational model to account for this feat, a task that our work attempts to accomplish. However, other researchers have generated some results along somewhat similar lines.

Peter Todd and his colleagues have used simple animations of the Heider and Simmel variety for testing simple social classification heuristics [4, 39]. They asked subjects to generate animations via a game: two people sat at computers and controlled two insect-like icons using a mouse, producing simple socially significant scenarios

such as "pursuit and evasion", "play" or "courting and being courted". Using these animations, they tested the ability of adults and children to categorize the interactions properly. They discovered that three-year-old children are able to distinguish, say, chasing from fighting, at rates above chance, and four- and five-year-olds perform better still. A sample of German adults was able to distinguish the six different tested intention regimes around 75% of the time, significantly better than children. Furthermore, the authors performed a cross-cultural study, looking at the categorization performance among Amazonian hunter-gatherer tribespeople. Their scores were nearly identical to those of the German adults, suggesting that the ability to intuit intention from this low-context data is not merely a cultural artifact.

In addition to studying human performance, Todd and his colleagues attempted to develop a computational mechanism to perform the same kinds of distinctions. They generated features such as relative heading and absolute speed and used very simple decision mechanisms to identify the social context behind the animation. Their approach showed that machines could recover this sort of social information from simple animations as well. However, they largely intended their experiments to demonstrate, successfully as it happens, the efficacy of their simple heuristics in relation to more computationally complex classification mechanisms. Their scenarios were very stylized, with a very narrow repertoire of possibilities, and while humans created the animations, they did so in an artificial machine-mediated fashion. Our approach, on the other hand, provides us with a record of real human activity in the world. Our scenarios are richer, and the level of detail is much greater. Rather than simple classification among a half-dozen intentions, our system generates a genuine narrative, and is not beholden to a pre-existing behavioral schema.

What's more, this can be accomplished quickly enough to serve as a control system for a robot, enabling us to explore the relationship between *watching* a game

and *participating*. When taking an active part, the system can probe uncertainties in its learning, collapsing ambiguity by performing experiments, and explore how motor control relates to social interaction [102].

2.2 Causality and animacy

> Certain physical events give an immediate causal impression, and ... one can "see" an object *act* on another object, *produce* in it certain changes, and *modify* it in one way or another.
>
> *Albert Michotte*

Our system also draws from and contributes to investigations of the fundamental cognitive processing modules underpinning perception and interpretation of motion. Once again, the pioneering work in the field was undertaken more than sixty years ago. The Belgian psychologist Albert Michotte performed hundreds of experiments investigating our tendency to attribute causality to low-context motion [66,67]. Using a primitive animation apparatus consisting of a rotating disc, some number of colored lines painted in various swirls, and a slit which caused the lines to appear as points to the observer (Figure 2.3), he found and characterized a number of irresistable perceptual effects. For example, when one animated circle approaches another, stationary, circle, and upon reaching it stops its motion while the previously fixed one begins to move, observers invariably perceive the second circle's motion to caused by the impact of the first, an effect Michotte named "launching".

We appear to possess special-purpose perceptual modules responsible for our rapid and irresistable computation of physics-based causality. Gelman [37] and Tremoulet [95] showed that trajectories alone are enough to stimulate the perception of animacy or inanimacy. When shown small particles moving about, subjects reported greater animacy from objects that made large speed or direction changes,

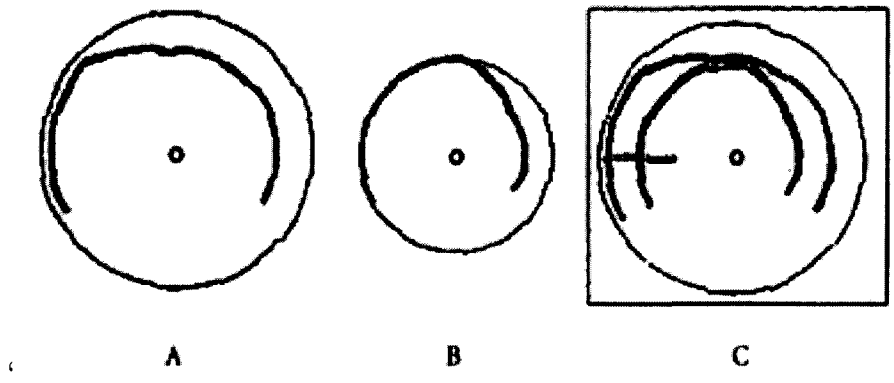

Figure 2.3: A schematic of Michotte's animation apparatus, from [66].

especially if such changes violated our physical intuitions about Newtonian mechanics. True, context has an effect – when subjects had access to orientation information as well as trajectories, they made somewhat different determinations. In later work, Tremoulet attempted to investigate the role of context more thoroughly [96]. Subjects watched a particle move along an angled trajectory, while another object was also present. That object's location affected the perception of the particle's animacy: if the particle's trajectory appeared to turn either toward or away from the object, the event looked much more animate and intentional.

We are far more likely to attribute intentionality to motion that looks like chasing or fleeing, and it is this reflex that our system both exploits and investigates. Tremoulet calls this a context-driven perception, in that perceptions are guided not purely by intrinsic motion but by motion relative to other objects. However, I would note that the interaction is still marked by a paucity of contextual data. Social perception requires two or more agents, naturally, but nothing beyond their position and trajectory is required to make sense of the interaction.

Cognitive neuroscientists have looked for evidence of these animacy processing modules in the brain. Blakemore [6] conducted an fMRI experiment where subjects watched an animation where two objects interacted. One object colliding with an-

other, causing a Michottean launching effect, activated the middle temporal gyrus. When the objects changed direction of their own accord, consistent with animacy perception, subjects' right lingual gyrus lit up. And one object orienting itself towards the other's motion, interpreted as attention, involved the superior parietal lobe. These results lend additional credence to the idea that these sorts of perceptions are the result of localized, bottom-up neural processing. My model demonstrates how such processing might occur.

Another demonstration of animacy processing in the brain comes from an fMRI experiment by Wheatley [98]. Her animations were not quite context-free in the sense that I've been using in this discussion – her subjects watched an object move on a cartoon background with recognizable features. A simple triangular object moved in a figure-8 pattern, in front of a background that looked either like a frozen lake or a tabletop strewn with toys. The idea was for the same motion to be taken as animate in one context (an ice skater on a pond) and inanimate in another (a spinning top). And indeed, nearly all of the subjects reported the expected bias, given the background. Strikingly, the whole suite of brain regions that have been identified as contributing to social cognition, from the amygdala to the medial frontal gyrus, was much more active when the stimulus was taken to be animate. The authors speculate that animacy perception primes the whole apparatus of social cognition.

If this is true, a deficit in perceiving animacy may disrupt all manner of social development. Rutherford [79] reports on experiments with autistic children that suggest that animacy perception in individuals with social deficits is less reflexive and automatic than for typically developing children and those with non-social developmental disorders. Rutherford presented these three categories of children (typically-developing, socially-unimpaired but developmentally disabled, and autism spectrum) with animations of moving circles. In the control condition, the children

were asked to determine which of two balls was heavier, using Michottean stimuli such as one ball rolling towards another at high speed and hitting another, which rolls away slowly. All three groups of children learned to distinguish the heavier from the lighter ball at about the same rate. In the experimental condition, where, for example, one circle moves out of the way of another, the children were rewarded for picking the circle that moved in animate fashion. The autistic children took, on average, twice as many trials as the typically developing children to learn to distinguish animate from inanimate. Interestingly, though autistic children had a harder time with the animacy task than the weight task, typical ones picked up on animacy much faster. However, once the children had learned to distinguish animate from inanimate motion appropriately, there was no difference in the performance of the various groups when faced with previously-unseen test data. Thus, these animacy cues are available to perception, and can be learned, even by autistic children. Modeling the interpretation of these perceptual cues, as our system does, may help illuminate both the initial difficulties and the learned compensatory mechanisms observed in autistic individuals.

These modules appear responsible for our rapid and irresistable computation of physics-based causality [11], as well as facile, subconscious individuation of objects in motion independently of any association with specific contextual features [60] [80] [68]. Furthermore, different processing modules appear to attend to different levels of detail in a scene, including global, low-context motion such as used by our system [61].

This is notable because such experiences seem to reflect automatic (and even irresistible) visual processing and are insulated to some degree from other aspects of cognition. Such percepts seem relatively unaffected by perceivers' intentions and beliefs, but are tightly controlled by subtle aspects of the displays themselves (for a

review, see Scholl and Tremoulet [81].

One characteristic of many of these studies in the psychology of perception, as well as the intention-attribution work described in the previous section, is a reliance on subjects' narrative responses. Narrative is important, and indeed the system described here uses narrative generation as a tool to make sense of the events it witnesses and participates in. However, disentangling people's actual perceptual ability from their intuition and higher-level cognition can be difficult. Experiments that rely on directly measurable phenomena, such as fMRI [63], have a place. In addition, experiments that measure *performance*, rather than judgment, on tasks requiring these perceptual skills can eliminate some of these confounds. In [35], Tao Gao and his colleagues attempt to do just that.

Gao used an animation of moving circles, one of which may or may not be chasing another, and asked subjects to determine whether chasing occured in a particular video, and to identify the hunter and target if so. The hunter varied in its *subtlety*, the directness with which it approached the target. The subjects' chasing perceptions were very reliable, but only if the chasing circle's heading never varied beyond about a 60° arc toward the target. If the chaser sidled up to the target by moving more laterally more often, subjects' abilities both to detect chasing occurring at all, and to identify participants in the behavior, quickly collapsed to near-chance.

This is true even though such subtle chasing remained remarkably effective. When given the opportunity to control the target circle, rather than merely observe, subjects were quite adept at identifying and avoiding the chasers that proceeded directly toward them (the "obvious wolf" in the authors' parlance), and able to avoid (though not necessarily identify) chasers whose choice of direction were nearly unconstrained by the position of the target (the "incompetent wolf"). However, chasers which often moved sideways but tended toward the target were far more difficult to avoid.

These psychophysical results can be immediately and directly translated to the domain of the system presented here, and the model tested for consistency. Indeed, the beliefs formed with respect to the observed agents' intentions are directly affected by how often and how reliably two agents move relative to one another (see Section 3.5). In future work I intend to test my system's performance on these explicit stimuli, and hypothesize that it should be comparable. It would also be interesting to move the other direction, and test whether the results I have obtained with respect to participation (Section 4.6) hold in human subjects. Are humans better at identifying a chasing activity if they are controlling one of the participants?

2.3 Force dynamics

> An object in a given situation is conceptualized as having an intrinsic force tendency, either toward action or toward rest.
>
> *Leonard Talmy*

The specific analysis undertaken by my system, hypothesizing vectors of attraction and repulsion between agents and objects in the world in order to explain the causal relationships we note in an interaction, comes from a notion of rudimentary folk physics. This model was first elucidated in 1988 by Leonard Talmy [91], who investigated how "force dynamics" form a natural semantic category that underpins an important part of lingusitic structure. He noticed that many of the things we wish to talk about – causes and effects, physical motion, and ultimately social interaction, can be viewed through the lens of physical or metaphorical forces at work in the world.

In its basic form, Talmy envisioned interactions between two agents, which he labeled "Agonist" and "Antagonist" (Figure 2.4). The semantic categorization we

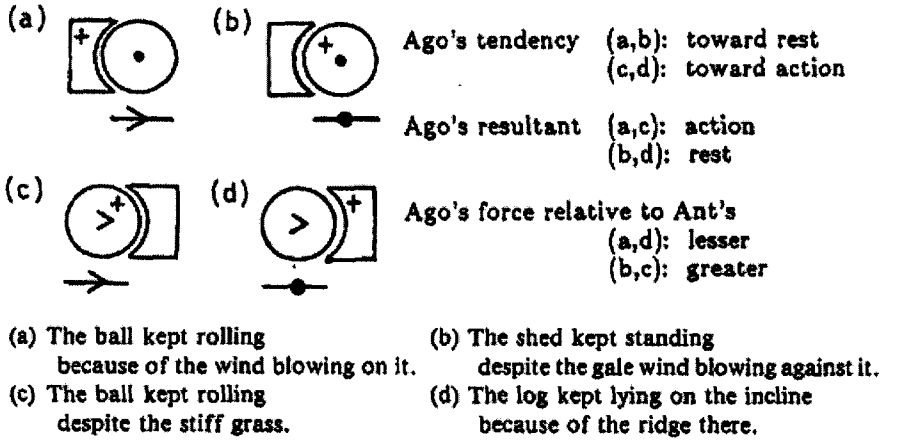

Figure 2.4: The basic framework of Talmy's force-dynamic patterns, from [91].

undertake when describing these interactions rests upon an analysis of the forces at work on the two agents. If the agonist's tendency toward motion exceeds the antagonist's ability to prevent that motion, the interaction is one of hindrance. With different magnitudes of force vectors at work, the interaction could instead be construed as assistance, causation or prevention, in this simple case.

Talmy then extends this framework to cover multiple agents, varying durations and directions of force application, and a number of similar characteristics that allow us to fine-tune our semantic categorization of situations and events. Most importantly for our purposes, he demonstrates how our notions of *physical* force become metaphors for a similar semantics dealing with social reference, modal verbs, expectation and argument. As an example, consider the following exchange (of Talmy's): "You know, I think Eric should sing at a recital – he has a beautiful singing voice." "Yes, but he can't stay on key." From a force-dynamical viewpoint, a pleasant voice is a force pushing toward a public performance, while an inability to hold key is another force represented as a vector in the opposite direction, and in this case, stronger than the first. Talmy's point is that this is exactly the same, in terms of our conceptualization of the situation and the cause and effect involved, as

an object sitting on a table that would fall to the ground due to gravity except for the stronger force exerted by the table surface.

Phillip Wolff extended this notion of folk-physical force vectors into experimental psychology [101]. The application of force has a great impact (no pun intended) on our understanding of the semantics of interaction, and on our ideas about causality, intention and influence. Humans can explain many events and interactions by invoking a folk-physics notion of force vectors acting upon objects and agents. This holds not only for obviously physical systems (we talk easily of how wind direction affects the motion of a sailboat), but for social interactions as well (the presence of a policeman can be interpreted – and in fact is described by the same vocabulary – as a force opposing our desire to jaywalk).

Wolff takes Talmy's notion of folk-physical forces and develops a more explicitly vector-based causal theory, which he then proceeds to verify experimentally against subjects' actual perceptions of causality. According to Wolff's model, people make sense of the causal structure of an event by taking the end state of the action, imagining the force vector that must have caused that end state, and explaining that vector as the resultant of physical or intentional forces. To wit, the observed behavior of a patient (Talmy's "agonist") can be explained by the vector sum of the forces intrinsically associated with the patient, those associated with an affector (or "antagonist"), and any other forces that may also be involved.

This notion was tested by showing animations to subjects. Unlike most of the other animations reviewed in this chapter, and the ones used in the research reported in the following chapters, these were *not* simply featureless shapes on a blank background, but rather computer-generated full-color full-context cartoons of boats and fans and blimps and women floating on rafts or trying to cross streets. Wolff's reported experiments were all variations on a similar theme. He presented his subjects

with videos of vehicles or people attempting to move in a particular direction, and encountering an external force – a physical one such as blowing fan, or a social one like a policeman's gestures. He then asked them to choose from a list of sentences to describe what had happened: "The fans caused the boat to reach the cone" vs. "The fans enabled the boat to reach the cone", for example. He found that the orientation of force vectors in each scenario was an extremely good (usually better than 90%) predictor of the descriptive langauge and causal interpretation produced by the test subjects.

Working with Anton Barbey, Wolff pushed his force-based causality framework to account for multiple agents [3] and transitive causal relationships. They found that, as entities proliferate and the complexity of situations increase, they also become more ambiguous, since the folk-physical representations of force vectors are "underspecified with respect to magnitude." That is, it is much easier to ascertain an operative vector's direction than its intensity, and when a number of vectors are at work, their relative contributions to the observed resultant are not obvious. Thus this transitive dynamics model explains why causal interpretations might be probabilistic in these cases – it requires evidence-based reasoning about unobserved phenomena.

Force dynamics theory is by no means uncontroversial in its account of how humans arrive at determinations of causality and agency. A number of other mechanisms have been proposed, such as mental model theory [42] and intervention on Bayes networks [75]. A pattern has emerged in the literature whereby a researcher will develop a model that predicts a pattern of responses with respect to people's causal intuition. The author will then demonstrate that the model under consideration achieves similar results as the tests run with other models, while developing an experimental twist that delivers the expected result only under the new paradigm.

For example, Sloman, in [87], develops a *causal model theory*, distinct from the mental or transitive dynamics models. In an experiment where subjects' causal intuitions are probed with respect to magnetism, conductivity and ionization, he reports that his model predicts the answers that his subjects actually provide, whereas Wolff's produces no clear determination.

This type of argument seems like it could be carried forward *ad infinitum*. The question I would pose, instead, is "Can a particular causal model sufficiently inform decisions about observed events, to yield appropriate interpretations and responses?" In other words, my work attempts to demonstrate the *utility* of force dynamics theory. In the domain of low-context trajectory data, hypothesizing about such forces is natural and computationally tractable. My system explicitly generates these systems of forces in order to make sense of the events it witnesses.

My work is by no means the first to use notions of force dynamics in creating computational models. Jeffrey Siskind [85,86] invokes Talmy's work as well, to inform his LEONARD system, which uses folk-physics notions to form judgments and assign labels to events. Using a camera and a motion-and-color tracker, it identifies polygons and theorizes about the physical relationships necessary to account for what it sees. For example, if a block is resting on a table, and a hand enters the frame and lifts it into the air, the system searches for the most physically parsimonious explanation: the block, originally supported passively by the table, becomes attached to a hand, which is an object that can move itself. It hypothesizes about the physical forces at work, and uses them to identify actions, in this case, the verb PICK-UP.

Siskind's work illustrates another very different application of the tool of force dynamics. Far from the notions of social motivation that I use, his hypothesized forces come from the physics-textbook world of friction and attachment. And while his system does not attempt to learn relationships and activities as mine does, it

demonstrates a subtle and robust classification system, quite powerful in its own domain of classifying a particular kind of verbs.

Chapter 3

Intention-based Reasoning

Our system begins with positional data taken from real humans playing games with one another, derived from an installed laboratory sensor network. True, this represents a bit of a shortcut – humans have no sensory apparatus that provides us with instantaneous physical location in a 3-D coordinate space. But we do have a huge visual computational apparatus that manages to translate retinal impulses into coherent, persistent, reliable object concepts. At the current state of the art in computational perception, this is still not possible – though work like ours may help to make it so.

Does this mean, then, that we cannot investigate the development of embodied, real-world cognition about complex social behaviors until we have completely solved the problems of perception which stand in the way? No, but it does suggest that we must endow our robots with other means of acquiring the information they need if we are to make any progress. Today's most advanced robotic navigation systems still require laser rangefinders and satellite triangulation to accomplish what human drivers can do with their eyes. Likewise, to bypass some of the complexities of visual perception, we use a sensor network (depicted in Figure 3.1) to supplement the

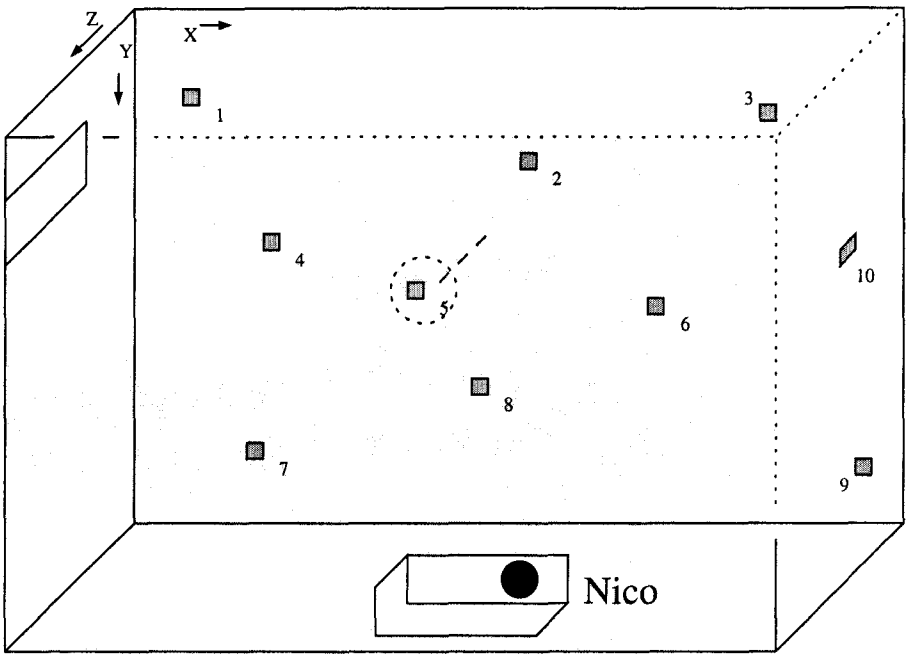

Figure 3.1: Schematic representation of lab layout, from above. Sensors 1-4 and 6-9 are affixed to the ceiling ($z = 0$). Sensor 5 hangs from an overhead light, 95 cm below the ceiling, and sensor 10 is attached to a wall at $z = 128$ cm.

robot's cameras with regular position updates for the people and objects it watches. As described in Chapter 4, we eventually do implement the architecture using a vision-based system.

The platform upon which we first implemented our developmental model is the robot Nico (see Figure 3.2), prior to endowing the system with a fully mobile RC car platform as described in Chapter 4. It is an upper-torso humanoid modelled after the body dimensions of a one-year-old child, possessing seven degrees of freedom in its head and six in in each arm. Nico has been used for a wide variety of research into childhood development of kinematics [90], language acquisition [41] and theory of mind [40], as well as investigating social interaction among humans such as the development of gaze following [27].

Nico's cameras give it stereoscopic vision, both a wide-angled peripheral and a highly-detailed foveal image. From the perspective of modelling the development of

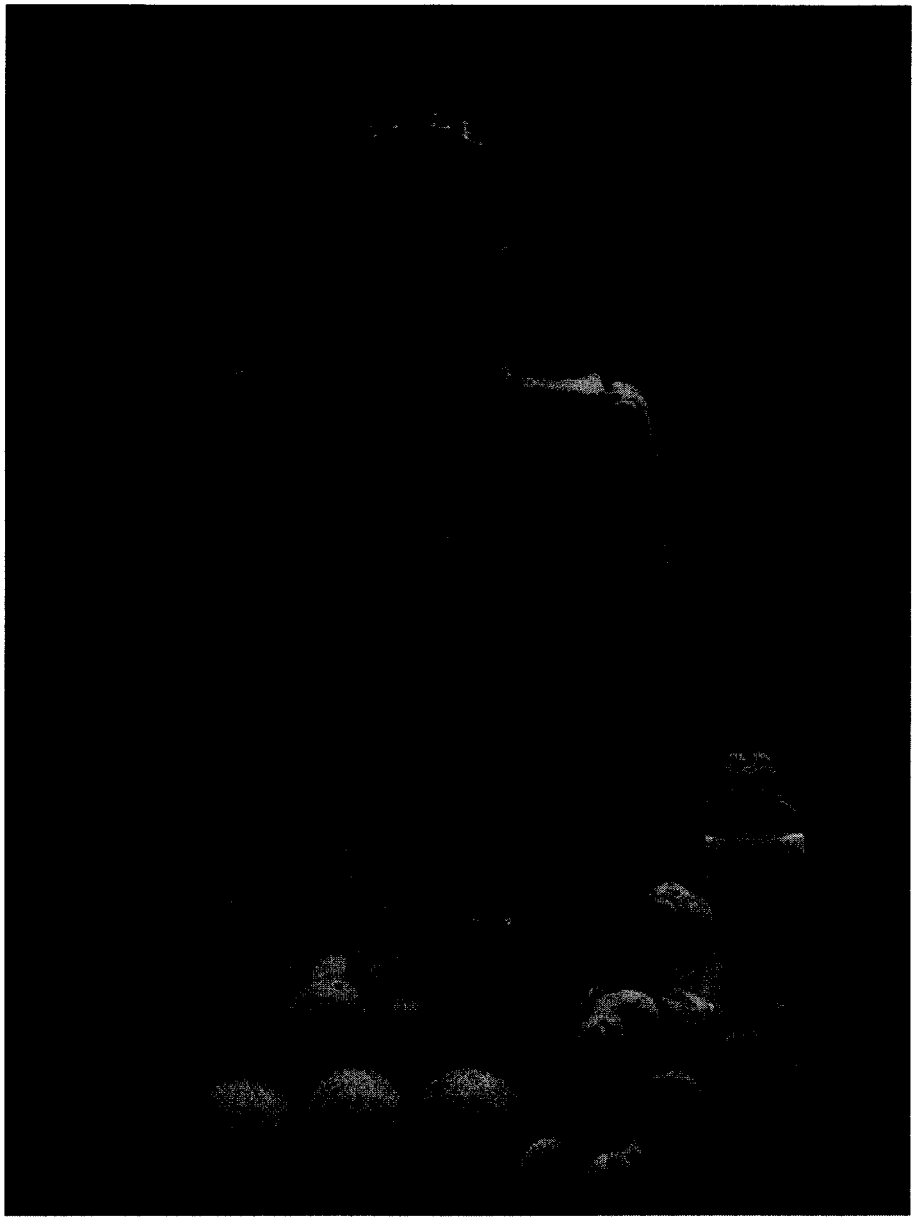

Figure 3.2: The robot Nico, which participated in the experiments as described in Section 3.4.

human social cognition, of course it would be best to sense the whereabouts of the people it observes by using this visual input exclusively, and having demonstrated the concepts explained in this paper, our current system attempts to do just that. The precise positioning data provided by our sensor network serves as a baseline for bootstrapping the more complex visual perceptual problem.

The system uses moment-by-moment trajectory data to hypothesize about the current intentional states of the people involved, in terms of their tendency to move toward and away from one another. It then creates a narrative by stringing together sequences of these hypothesized intentional states and noticing relevant facts about the moments where the players' intentions shift. Armed with this information, the system can report the narratives that it constructs, produce evaluations of events that agree with human observers, and reliably classify and differentiate a number of playground games: Tag, Smear, Keepaway and Catch. Furthermore, from comparing the raw data to the narrative sequences it constructs, the system can infer some of the hidden rules of the games it sees. For instance, it deduces the fact that in Tag, a person can become the new "it" only when the old "it" approaches sufficiently close to tag.

3.1 Detecting motion

Most of the literature examining how intentions and goals can be inferred from motion cues relies on canned or simulated worlds. From Heider and Simmel's manually-constructed animations to Peter Todd's computer bug game, researchers have studied humans' and computers' abilities to classify and interpret simple, abstract animations by generating such animations in simple, abstract ways. In contrast, our data derive from real-world human interaction, obtained with a localizing sensor network

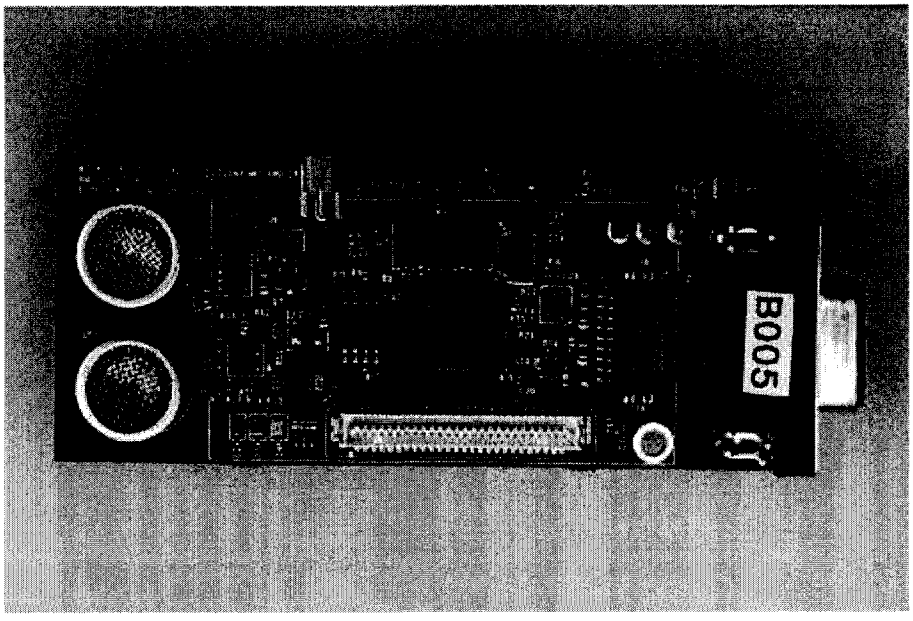

Figure 3.3: The Cricket ultrasound/radio sensor used in localization and tracking. [76]

of radio- and acoustic-enabled Cricket nodes.[1] These devices send messages to one another using a simultaneous radio broadcast and ultrasound chirp, and the receiving unit can calculate distance by comparing the difference in arrival times between the two signals. With a beacon node corresponding to each agent involved in a particular game or dramatic scenario, and ten listener nodes in fixed positions throughout the room, we can determine a person or object's location within a few centimeters through triangulation. The sensor suite generates five position reports per sensor per second – fast enough to create fairly smooth motion trajectories at normal human running speeds, and to notice important events in significantly less than a second. Each participant carried or wore a uniquely-identified sensor throughout a particular scenario, and the toys or objects they manipulated were likewise tagged (see Figure 3.3).

This sensor system produces a five-dimensional vector of data: the identity of an

[1]See [76] for hardware details. We entirely redesigned the embedded control systems and data analysis software to meet the needs of our experiments.

object, its x, y and z coordinates in the room, and the time at which these coordinates were measured (accurate to the millisecond). Figure 3.1 shows the coordinate frame and the robot's position with respect to the rest of the room. The z coordinate, measuring vertical distance from the lab's ceiling, is only used for appropriate distance measuring and robot-human interaction; a two-dimensional top-down representation is perfectly adequate to interpret the activity. Our robot uses this data to perform its analysis of the interactions it sees, and to respond appropriately.

3.2 Generating human-readable data

In addition, we use the data to construct simple Heider-and-Simmel-style animations representing the scenario. Our system generates a colored square for each agent and interpolates between detected positions to create an animation for human evaluation. From a record of the raw position reports coming from sensors, we generate a Scalable Vector Graphics (SVG) file specifying an animation which can then be viewed in a web browser. Each test subject viewed the videos and was asked to click with the mouse whenever a significant event occurred, with the system capturing the time of each click. After the viewing, the subject was asked to specify her interpretation of the events she saw. In this way, we can directly compare the human and robot ability to generate hypotheses concerning the witnessed activity, using exactly the same perceptual data. In most cases, the interpolation of the motion between the detected positions is simply linear, but for some of our test runs, we used cubic splines to generate a smoother cartoon. Our human test subjects had no trouble interpreting the simpler linear model, however, though most test subjects reported that the smoother animation looked subjectively better.

To be considered as an appropriate model of social cognition, our system should

produce the same analyses from the same data as the subjective impressions of the humans watching these cartoons. Provided that humans can readily interpret the intentions, goals and dramatic moments in animated representations of simple four-dimensional data, our machine-based algorithm should be able to do the same. To do this, we take our cue from psychology and folk physics and cause the robot to imagine "forces" acting on the people it sees, causing them to act the way they do.

3.3 Characterizing attraction and repulsion

Our basic approach involves generating a set of hypotheses for each agent's attractions and antipathies toward the other objects and agents in the room, and recognizing the points at which these hypotheses needed to change in order to continue to represent the data well. We assumed that each agent moved in accordance with these qualities – in other words, at each moment in time, an agent's velocity (or, alternately, acceleration) is the sum of the attractive and repulsive vectors associated with each of the other agents and objects in the scenario. The algorithm searches for the best fit in a least-squares sense among competing hypotheses over a short time frame – depending on the experiment, anywhere from 0.5 to 2 seconds. Additionally, the influences between agents could be inverse, inverse-square or constant.

To make these calculations, we proceed as follows: For each agent and object in the observed environment, the system calculates the "influence" of the other people and objects on its perceived two-dimensional motion, expressed as constants in a pair of differential equations:

$$V_{x_i^n} = \frac{c_{x_j}(x_j^n - x_i^n)}{d_{ij}^n} + \frac{c_{x_k}(x_k^n - x_i^n)}{d_{ik}^n} + \cdots \tag{3.1}$$

(and similarly for the y dimension). Here, $V_{x_i^n}$ represents the x component of agent i's velocity at time n. x_i^n, x_j^n and x_k^n are the x coordinates of agents i, j and k respectively, at time n. Likewise, d_{ij}^n and d_{ik}^n are the Euclidean distances between i and j or i and k at time n. We obtain the (noisy) velocities in the x and y direction, as well as the positions of the other agents and objects, directly from the sensor data:

$$V_{x_i^n} = \frac{x_i^{n+1} - x_i^n}{t_{n+1} - t_n} \tag{3.2}$$

(again, also for the y dimension).

This results in an underconstrained set of equations; thus to solve for the constants we collect all of the data points falling within a short window of time and find a least-squares best fit. We assume that the agent's velocity is determined by a combination of influences from the other participants in the scenario, represented by one of a small number of equations:

- Constant: $V_{x_i^n} = \frac{c_{x_j}(x_j^n - x_i^n)}{d_{ij}^n} + \frac{c_{x_k}(x_k^n - x_i^n)}{d_{ik}^n} + \cdots$ (and similarly along the y dimension)
- Inverse: $V_{x_i^n} = \frac{c_{x_j}(x_j^n - x_i^n)}{(d_{ij}^n)^2} + \frac{c_{x_k}(x_k^n - x_i^n)}{(d_{ik}^n)^2} + \cdots$ (and, again, similarly along y)
- Inverse squared: $V_{x_i^n} = \frac{c_{x_j}(x_j^n - x_i^n)}{(d_{ij}^n)^3} + \frac{c_{x_k}(x_k^n - x_i^n)}{(d_{ik}^n)^3} + \cdots$ (same with y)

We then collect all of the data points falling within a short time period into a pair of matrices. For three total agents involved, using the constant-influence hypothesis, and with a time window providing three position reports per sensor ($\delta n = 0.5$ seconds), these matrices would be:

$$A = \begin{vmatrix} \frac{x_j^n - x_i^n}{d_{ij}^n} & \frac{x_k^n - x_i^n}{d_{ik}^n} \\ \frac{y_j^n - y_i^n}{d_{ij}^n} & \frac{y_k^n - y_i^n}{d_{ik}^n} \\ \frac{x_j^{n+1} - x_i^{n+1}}{d_{ij}^{n+1}} & \frac{x_k^{n+1} - x_i^{n+1}}{d_{ik}^{n+1}} \\ \frac{y_j^{n+1} - y_i^{n+1}}{d_{ij}^{n+1}} & \frac{y_k^{n+1} - y_i^{n+1}}{d_{ik}^{n+1}} \\ \frac{x_j^{n+2} - x_i^{n+2}}{d_{ij}^{n+2}} & \frac{x_k^{n+2} - x_i^{n+1}}{d_{ik}^{n+1}} \\ \frac{y_j^{n+2} - y_i^{n+2}}{d_{ij}^{n+2}} & \frac{y_k^{n+2} - y_i^{n+2}}{d_{ik}^{n+2}} \end{vmatrix}, b = \begin{vmatrix} V_{x_i^n} \\ V_{y_i^n} \\ V_{x_i^{n+1}} \\ V_{y_i^{n+1}} \\ V_{x_i^{n+2}} \\ V_{y_i^{n+2}} \end{vmatrix}$$

The matrices representing the other two hypothesis domains are constructed similarly.

Finally, we construct a QR-decomposition of matrix A, and use it to find the least-squares solution of $A \times X = b$. The matrix X thus found corresponds to the best-fit constants for the hypothesized equations of motion.

The constants found by this process reflect the strengths of the relative force vectors in play at a particular point in time as they influence the motion of the agents in the scene – not physical forces, of course, but rather the conceptual forces arising from the social situation and the agents' goals. These constants form a set of features which the robot uses to segment and identify the events it watches. Moments when the constants change sign, or vary dramatically in magnitude, or where the set of equations that produce the best data fit changes, are particularly important, and often corresponding to significant events in the scenario. These are usually the same moments that human observers mark, as well, as the experiments described below bear out.

This sort of computation is not scalable to arbitrarily large numbers of participants, though the matrices grow in a fairly reasonable $O(n^2)$ fashion. On the other

hand, it does not seem plausible that humans are able to interpret the individual roles of every participant in a crowd scene, either. Our own attentional bound on objects we can visually track is quite modest, well within the computational limits of our approach. Rather, we tend to analyze large-scale interactions in terms of groups of people with similar roles and goals, and our approach can be extended to accommodate the same kind of evaluation.

3.4 Participation

Not only is the robot able to detect and interpret the events it sees, but it can respond and participate as well. We implemented a repertoire of gestures and behaviors that the robot can use in real time while observing a social interaction. The robot's motor system acts on information from the sensor tracking system and the intention recognition system. While the robot constantly receives sensor position data from the tracking system, the intention recognition system can induce Nico to perform the following behaviors:

- **Visual Tracking:** Nico can visually track a particular agent or object by turning its head to follow a sensor's movement about the room. At any time, the system can substitute one object of attention for another, or leave off visual tracking entirely.

- **Arm Tracking:** Nico stretches its right arm to point at a particular object or agent. This behavior is controlled flexibly and dynamically, according to the robot's current evaluation of the events and intentions it perceives. The left arm was not functional at the time of the experiment.

- **Avoidance Behavior:** Nico can decide to show avoidance behavior towards

the sensor attached to one of the objects or people in the room. It will look away from whatever it is avoiding and shield its face with an arm. This behavior supersedes the previous behaviors.

The first two behaviors can be combined – Nico can watch and point at different things simultaneously – but the avoidance behavior can only be executed exclusively.

We measured the absolute positions of the head (H) and the right shoulder (S) in the coordinate system that the Cricket sensors use. To track an object with position X either visually or with the arm we geometrically compute the difference vectors $D_H = X - H$ or $D_S = X - S$ respectively. Using the difference vectors and a geometrical model of the robot the system computes the desired motor positions. The head always uses two degrees of freedom (pitch and yaw). The arm usually uses only two degrees of freedom in the shoulder (move arm up/down, left/right). However, if the robot needs to point far to the right, the shoulder joints reach the limit of their range of motion and the two elbow joints must be used in addition to complete the pointing gesture with the lower arm.

This, of course, is a very rudimentary form of participation in the games. Later, in Chapter 4, I describe transferring the system to a mobile robotic system that can participate fully alongside human players.

3.5 Identifying intentions over time

Each constant determined by the process described above represents the influence of one particular agent or object on the motion of another at a particular point in time. Some of these may be spurious relationships, while others capture something essential about the motivations and intentions of the agents involved.

To determine the long-term relationships that do represent essential motivational

information, we next assemble these basic building blocks – the time-stamped pairwise constants that describe instantaneous attraction and repulsion between each agent and object in the room – into a probabilistic finite state automaton, each state representing a set of intentions that extend over time. At any particular point in time, any particular agent may be attracted or repelled or remain neutral with respect to each other object and agent in the room; this is characterized by the pairwise constants found in the previous step. The system assumes that the actors in the room remain in a particular *intentional state* as long as the pattern of hypothesized attractions, repulsions and neutralities remains constant, discounting noise. Such a state consists of the collection of intentions of each individual with respect to every other (pairwise) during a particular phase of an interaction. A particular state, then, might be that A is attracted by B and neutral toward C, B is repelled by A and neutral toward C, and C is repelled by A and neutral toward B. This state might occur, for instance, in the game of tag when A is it and has decided to chase B.

The system maintains an evolving set of beliefs about the intentions of the people it observes, modeled as a probability distribution over all of these possible states. As new data comes in, the current belief distribution is adjusted, and the system assumes that the most likely alternative reflects the current state of the game.

$$Bel_n(S) = \frac{Bel_{n-1}(S)(1 + \lambda \sum_{c \in S} \sigma(S, c_n))}{Z} \tag{3.3}$$

Here, the belief in any particular state S at time n (the state being a particular collection of mutual pairwise intentions between all of the players of a game) is the belief in that state at time $n-1$, modified by the current observation. c_n is the value at time n of one of the pairwise relationship constants derived from the data in the previous step; the function σ is a sign function that returns 1 if the constant's sign

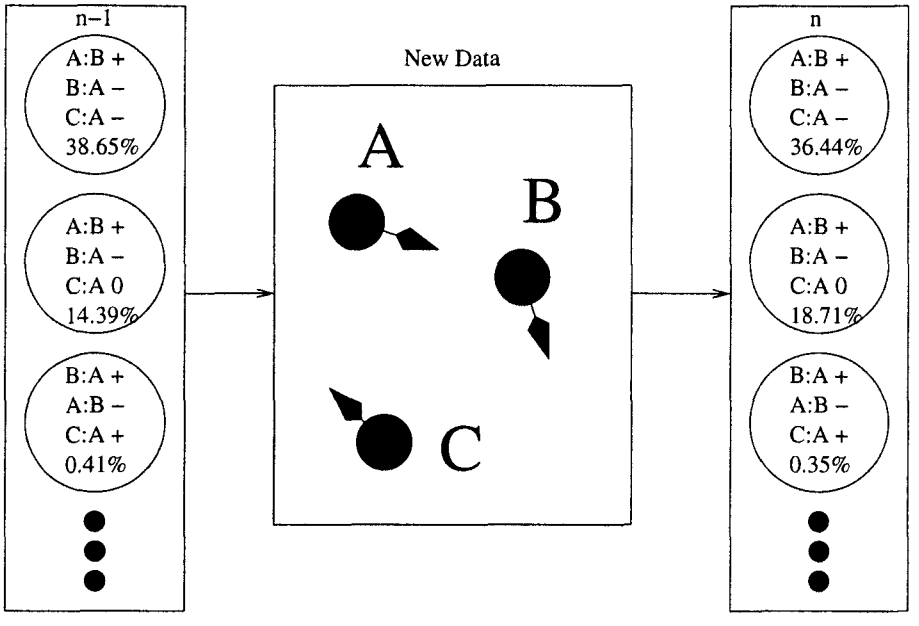

Figure 3.4: The system's level of belief in a few intentional states, evolving as new information arrives. The letter pairs in each state circle represent the first object's intentions toward the second: + denotes attraction, – repulsion, and 0 neutrality. At time $n-1$, the system believes that A is intending to chase B, while B and C are fleeing from A (this state, the top circle, enjoys the highest degree of belief). At time n, new data arrives that shows A continuing to chase B, but C moving sideways. Accordingly, the system's belief that B and C both want to escape from A declines (top circles), while the belief that C is neutral (middle circles) increases. More of the same kind of data would have to come in before the system would switch its belief to the middle-circle state, at which point it would review its observational history to determine the point where C actually stopped running from A. The bottom circles represent B and C chasing A, while A wants to evade B – a situation that the system currently views as very unlikely.

and the intention represented by the current state agree, -1 if they disagree, and 0 if the state is neutral toward the pairwise relationship represented by the constant. λ is a "learning rate" constant which affects the tradeoff between the system's sensitivity to error and its decision-making speed. Through trial and error, we found a value of $\lambda = 0.08$ to yield the best results. Finally, Z is a normalizing constant obtained by summing the updated belief values across all of the states in the system. For an example of how the beliefs of a few of these states evolve, see Figure 3.4.

The function σ depends on the signs, not the values, of the constants obtained in the previous step. It is straightforward to adjust the function to take account of the magnitudes of the social forces involved, but we discovered that doing so increased the effect of sensor noise unacceptably, without causing a noticeable improvement in performance. Rather, it is the evolution of these belief states over time which mitigates and smooths the effects of noise. While the data is rife with random hiccups in the sensed positions, the beliefs in a particular intentional state change only relatively slowly. As long as the preponderance of evidence supports a particular set of intentional hypotheses, the system's judgement resists being affected by momentary aberrant noise.

Notice that the system never explicitly handles ambivalence in its computations – the states that accommodate neutral attitudes never update their beliefs based on the values of any constant that may exist between the neutral pair. Instead, the degree of belief in these states changes owing to normalization with other states that are changing based on attraction and repulsion, which amounts to a decision to "default" to a neutral state – if our degree of belief in both pairwise attraction *and* repulsion falls, then naturally our belief in a state of ambivalence between the pair should increase.

This is not an approach which can deal easily with large numbers. The number

of states in the system depends exponentially on the number of objects and agents being tracked. Each state represents a particular configuration of attractions, repulsions and neutralities between each agent pair. Although the number of states is manageable with four participants, it will eventually become too computationally complex to represent this way. An upper bound on the state space is 9^4, or 6561, but a large majority of those putative states are never observed. How quickly the belief computation would become unwieldy, especially if one were to apply techniques such as zero-compression (discussed at greater length in Appendix B), is an open question. However, as already discussed in section 3.3, there is no reason to believe that humans are any more capable of keeping track of many people's simultaneous motivations and goals, either.

3.6 Constructing narrative

The process described in the preceding section converts instantaneous, noisy velocity vectors into sustained beliefs about the intentional situation that pertains during a particular phase of a witnessed interaction. As the action progresses, so too do the system's beliefs evolve, and as those beliefs change, the sequence of states becomes a narrative describing the scenario in progress. This narrative can be analyzed statistically to identify the action in progress, differentiate it from other possible activities, and also provide the system with clues to use in unsupervised feature detection.

When the system finds that it now believes the action it is watching has changed state, it retroactively updates its beliefs – the point at which the belief changed is *not* the actual time point at which the witnessed behavior changed. Rather, several rounds of data collection caused the change in belief, so if the system is to reason about an actual event that changed the participants' intentional state, it must

backtrack to the point where the new data began to arrive. This it accomplishes by looking at the derivative of the beliefs in the two states. The system assumes that the actual event occurred at

$$\mathbf{max}(t)(\frac{d}{dt}Bel_t(S_{\mathbf{old}}) \geq 0 \vee \frac{d}{dt}Bel_t(S_{\mathbf{new}}) \leq 0) \tag{3.4}$$

That is, the most recent time point when either the system's belief in the old state was not decreasing or the system's belief in its new state was not increasing.

The system now strings these states and durations into an actual (admittedly dull) narrative: "State 13 for 4.6 seconds, then state 28 for 3.9 seconds, then state 4 for 7.3 seconds..." It translates the states into actual English: "A was chasing B for 4.6 seconds while C was running from A, then B was chasing C for 3.9 seconds while A stood by..." It collects statistics about which states commonly follow which others (a prerequisite for developing the ability to recognize distinct activities). Furthermore, it has identified points in time where important events take place, which will allow the system to notice information about the events themselves.

3.7 Differentiating activity

The system has now constructed a narrative, an ordered sequence of states representing the hypothesized intentions of the people participating. These intentions ebb and flow according to the progress of the game and the shifting roles of the actors as they follow the rules of their particular game.

How can one evaluate this narrative to determine whether it amounts to anything we would recognize as understanding the game? One way, already discussed in Section 3.2, is to compare the narrative with ones constructed by human observers

of the same events. This is an attractive approach as it directly compares human and machine performance, but usually in qualitative fashion. Hence, we additionally decided to use a statistical approach and determine whether the generated narrative suffices for the system to distinguish between different games. At the same time, it should be able to recognize that the narratives it constructs regarding different instances of the *same* game are sufficiently similar.

To do this, we treat the narratives (perhaps naturally) as linguistic constructions, and apply similarity metrics which are most often used in computational linguistics and automated language translation. In this work, we use Dice's coefficient [26], which compares how often any particular word (or in our case, intentional state) follows any other in two separate texts (or narratives).

$$s = \frac{2n_t}{n_x + n_y} \tag{3.5}$$

Here, n_t is the total number of bigrams (one particular state followed by another) used in both narratives, while n_x and n_y are the number in the first and second narratives, respectively.

As an example, take comparison between two games of Tag. One has 21 unique state transitions of the kind suggested in Figure 3.4, such as a transition from believing that C is fleeing from A to one where C is neutral toward A. Another game has 26 transitions, and the two games share 20 state transitions in common. In this case, the coefficient would be $\frac{2(20)}{21+26} = 0.86$.

This is obviously an extremely simple heuristic, and indeed Dice's coefficient does not perform as well as state-of-the-art statistical comparisons for linguistic analysis [72], but it has the advantage of simplicity and does not rely on a large narrative corpus to function.

3.8 Unsupervised rule recognition

Building step-by-step from applying force dynamics to real-word data to create hypotheses about individual intentional states, through establishing a probabilistic belief about the sequence of intentional states in a particular narrative, to comparing and contrasting narratives to find similarites and differences, our system has by now developed a fairly sophisticated and rich understanding of the social framework of the activities that it has watched. Furthermore, it possesses the tools to make its own educated guesses at contextual elements separate from the trajectory information which it used to build up its narratives. We demonstrate this by letting the system look at relative position information (*not* relative motion, which is what it has been using up to this point) and allowing it to form its own hypotheses about whether position is significant to the rules of the games it observes.

The system has already identified the moments where the intentional states of the actors change during gameplay. Why do these changes happen? Something in the rules of the game, most likely. And something to do with relative or absolute position is a good guess. Associated with each particular change in intentional state (the state bigrams discussed in section 3.7), the system collects statistics on the absolute position of each actor and the relative distances between them all. If the mean of the relative distances is small, say, $\mu < 30$ cm, or, likewise, if the standard deviation of the absolute positions is small ($\sigma < 30$ cm), then it reports that it has found a potential rule, such as: in this game, and similar ones, the role of chaser and chased switch only when the two come into close contact. In other words, Tag!

Chapter 4

An Intention-based Robotic Control System

The system involves a number of interconnected pieces, depicted in Figure 4.1. Each component is described below in turn.

4.1 Vision

The system employs a simple but robust method to tracking the players as they move through the play space. Using an inexpensive USB webcam mounted on a stand in such a way as to provide a complete image of the floor of the room, the system uses naive background subtraction and color matching to track the brightly-colored vehicles. Before play begins, the camera captures a 640x480 pixel array of the unoccupied room for reference. During a game, 15 times a second, the system examines the raster of RGB values from the webcam and looks for the maximum red, green and blue values that differ substantially from the background and from the other two color channels of the same pixel. These maximum color values are

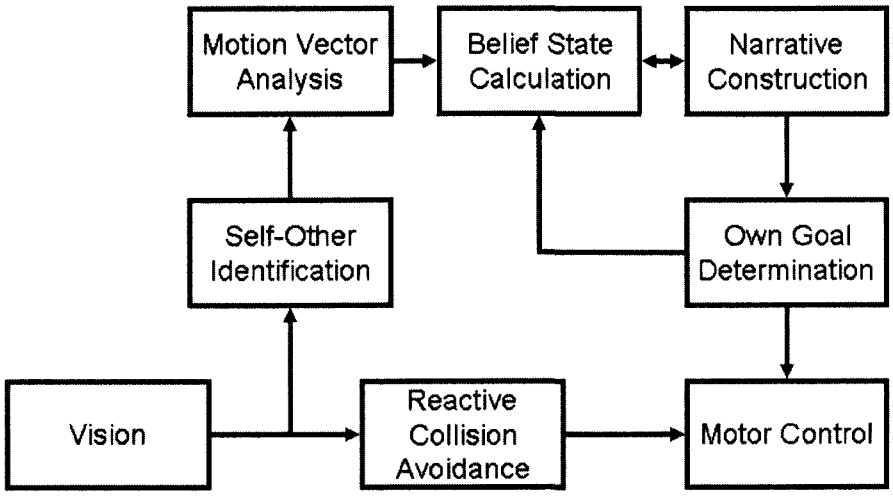

Figure 4.1: System components and information flow

taken to be the positions within the visual field of the three vehicles – one painted red, one blue, and one green (by design). Obviously, this is not a general-purpose or sophisticated visual tracking algorithm, but it is sufficient to generate the low-context percepts that are all our cognitive model requires.

Note that the camera is *not* overhead. The information coming to the robot is a trapezoid with perspective foreshortening. It would be possible to perform a matrix transformation to convert pixel positions to Cartesian geospatial ones, but our system does not go to the computational expense of doing so. The image may be distorted, but only in a linear way, and the vector calculations described below work the same, whether in a perspective frame or not.

4.2 Motor control

In order not only to observe but to participate in activities, we provided our system with a robotic avatar in the form of a $20 toy remote-controlled car. By opening up the plastic radio controller and wiring in transistors to replace the physical rocker

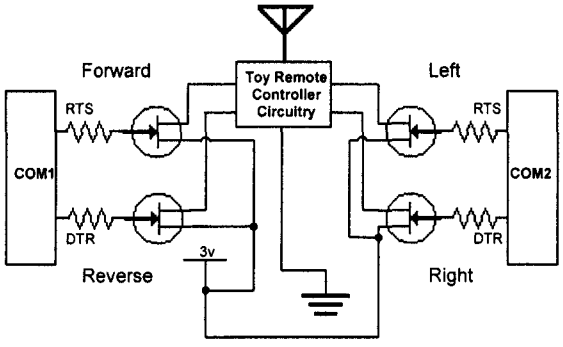

Figure 4.2: Circuit diagram for computer-controlled toy RC car

switches that control the car's driving and steering, and connecting these wires to controllable voltage pins on a computer's serial ports, we turned the system into a high-speed (7 m/s) robot. See Figure 4.2 for wiring details.

The controller is quick and reactive. The system maintains the position history over the previous $\frac{1}{5}$ second – three position reports, including the current one. With this information, it computes an average velocity vector and compares it with the intended vector given by the own-goal system described further below. Depending on the current direction of drive and the angle of difference between the actual and intended vectors, a set of commands is sent to the robot as shown in Figure 4.3.

4.3 Reactive collision avoidance

The room's walls obviously have an effect on the motions of the players, since their actions are constrained by the physical dimensions of the space. We chose to deal with wall avoidance in simple fashion. If the robot approaches too near the edge of the play area, a reactive behavior emerges that is independent of the goal state: if the robot is located within a certain number of pixels of the edge of the play area, an emergency goal vector pointing straight out from the wall or corner supersedes whatever the robot had been trying to do beforehand. This danger area ranged from

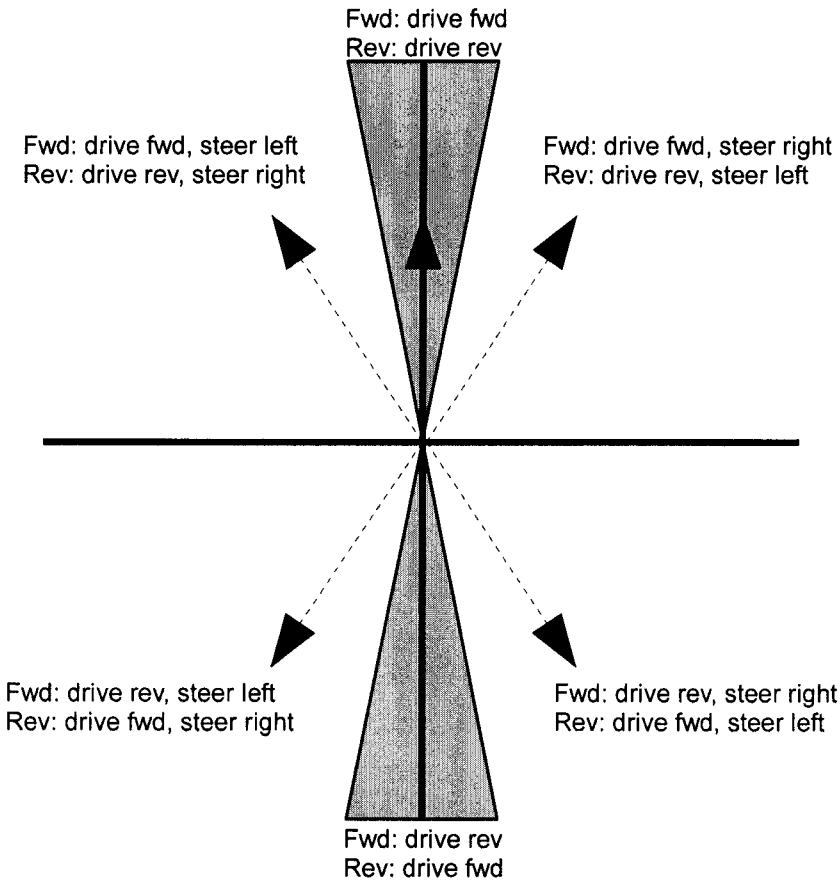

Figure 4.3: Robot directional control. The goal vector is computed relative to the robot's current motion, which is always taken to be straight along the y-axis. Motor commands are chosen depending upon the quadrant in which the goal vector appears and the current state of the robot's drive motor.

30 pixels wide at the bottom of the image (closest to the camera) to 18 near the top. Interestingly, several study participants noted the robot's ability to avoid running into walls, claiming that the robot was a much better driver than they were! Poor driving performance on the part of the human subjects was one motivator for the RF sensor network system described in Section 4.8.

4.4 Self-other identification

The system does not immediately know what salient object in its visual field "belongs" to itself. The playing area contains three different-colored toy cars, but it controls only one. Using a technique described in [40] for robotic self-recognition, the system sends out a few random motor commands and detects which of the perceived objects responds in a correlated fashion. The system sends a brief pulse (200 ms) of the command for "forward", followed by a similar command for "back", repeating as necessary. At the same time, the system inspects the visual field for the positions of the three salient colorful objects, looking for one moving predictably forward and back in time with the commands (finding and computing the necessary motion vectors are a byproduct of the analysis described in the next section). In this way, the system identifies itself for the duration of the exercise. Although the process would theoretically continue for as long as necessary, we found that throughout our experiments it never took more than one forward and reverse command for reliable identification.

Notably, this is precisely the same procedure invariably used by the human participants, who were each handed a remote controller without being told which of the three cars they were to drive. Each time, without exception, the participant worked the controls forward and backward, watching the playing area to note which car

acted as directed. The system has access to no privileged information about what it sees, no more than an undergraduate test subject walking into the lab space for the first time.

4.5 Motion vectors and belief states

Having determined which vehicle it is driving, the system begins to observe the behavior of the others to begin working out the rules of the game, as described in Sections 3.3 and 3.5.

Because the robotic system employs visual tracking, the temporal resolution is improved over the previous sensor network-based incarnation – although of course, it only works because color segmentation on solid-color toy cars turns out to be quite easy. The visual system runs at 15 Hz; we found that a window of 220 milliseconds (about three position reports) worked best. Three position reports were also ideal for the ultrasound location system, but the slower refresh rate meant a longer and less accurate recognition window.

Another difference between the robotic system and the passive observer comes from an adjustable learning rate (λ in the equations shown in Section 3.5). Previously, we used a constant $\lambda = 0.08$, but the robotic system adjusts the learning rate on the fly, between $\lambda = 0.04$ and $\lambda = 0.12$. This depends upon whether it is merely observing or is actively participating and trying out hypotheses (see the following section).

4.6 Own goal determination

As the system begins to observe the activity of its human partners, it develops a belief distribution over their possible intentional states. Because it controls a robot of its own, the system is then able to *probe* the likeliest candidate states. It chooses the belief state it has rated most likely, and acts in such a way to confirm or reject the hypothesis. It adjusts its beliefs accordingly, and more decisively than if it was not participating.

For example, say that the system had the highest degree of belief in the following state: Green was chasing Red and ignoring Blue, while Red was fleeing from both Green and Blue. To probe this state of affairs, the system would drive Blue toward Red. If Red continued to move away from Blue and Green did not react, the system's degree of belief in this state would further increase; if the other players reacted in some other way, the belief would subside, eventually to be replaced by another belief state judged more likely. In our experiments, human players were instructed to involve the robot in their gameplay only insofar as the robot appeared to act appropriately, so that the robot's probing actions would be less likely to spoil the game.

The ability to participate in and change the course of the game is a powerful tool for efficient learning. Machine learning theory is full of algorithms which perform much better when they are allowed to pose queries, rather than simply passively receiving examples [2]. Our system possess an analogous ability, able to query its environment and settling ambiguities in its beliefs by manipulating its own intentions and behaviors. At the same time, it watches for the effects on others' behaviors of the social forces brought into play by its actions. We show the effectiveness of such participation below.

4.7 Narrative construction

The process described in the preceding sections converts instaneous velocity vectors derived from somewhat noisy video into sustained beliefs about the intentional situation that pertains during a particular phase of an interaction. As the action progresses, so too do the system's beliefs evolve, and as those beliefs change, the sequence of states becomes a narrative describing the scenario in progress. This narrative can be analyzed statistically to identify the action in progress, differentiate it from other possible activities, and also provide the system with clues to use in unsupervised feature detection. It can collect statistics about which states commonly follow which others (a prerequisite for developing the ability to recognize distinct activities). And it identifies points in time where important events take place, which will allow the system to notice information about the events themselves.

For this particular set of scenarios involving playground-like games, we set the system to look for game rules by observing the relative positions of the participants during the crucial moments of a belief state change, and to search for correlations between the observed distances and the particular state change. Distance is only one feature that could be considered, of course, but it is a common-enough criterion in the world of playground games to be a reasonable choice for the system to focus on. If the correlations it observes between a particular state transition and a set of relative distances are strong enough, it will preemptively adjust its own behavior according to the transition it has learned, thus playing the game and not only learning it.

4.8 RF tracking

While the visual tracking system employed by the robot works reasonably well, it has a number of shortcomings. First of all, the background subtraction and color

segmentation are very brittle, and require a careful preparation of the room, its furniture, organization and lighting, before experiments could take place, and this played a part in limiting the opportunities available to run experiments. Secondly, the number of human participants in the games were limited to two – the number of brightly-colored remote-control vehicles available to play with. This obviously restricts the kinds of games the robot is able to watch. To learn tag with only two people to demonstrate the rules of the game is difficult, since it should be a fluid game with several people running from "it". And finally, driving remote-controlled vehicles well and accurately is a skill that takes time and practice to develop. The cars are fast, tricky to handle and not terribly maneuverable. Our experimental subjects were not always able to operate their vehicles as well as they wanted, and this limited the range of actitivies they were able to engage in reliably.

Because of these shortcomings, we employed a third sensor system in order to finally reach a system that could actually learn and play complex and dynamic games such as tag. The Ubisense system [89] employs a system of ultra-wideband phased-array receivers deployed in the corners of a room, connected by timing cables of precise length. Small RFID-like tags, shown in Figure 4.4, broadcast periodic radio pulses (at 5 Hz) which are detected by the sensors. Through a combination of angle-of-attack and time-difference-of-arrival calculations, the system arrives at each tag's location and reports it over the network to the robot controller.

Experiments carried out with the RF sensor network involved three people wearing baseball caps to which the radio beacons had been attached, as well as two affixed to the radio-controlled toy car to be manipulated by the computer system. Of these two, one was used as a position-reporting sensor just like the ones worn by human participants, while the other was placed along the car's longitudinal axis and allowed the system to distinguish forward from reverse movement. This simplified the control

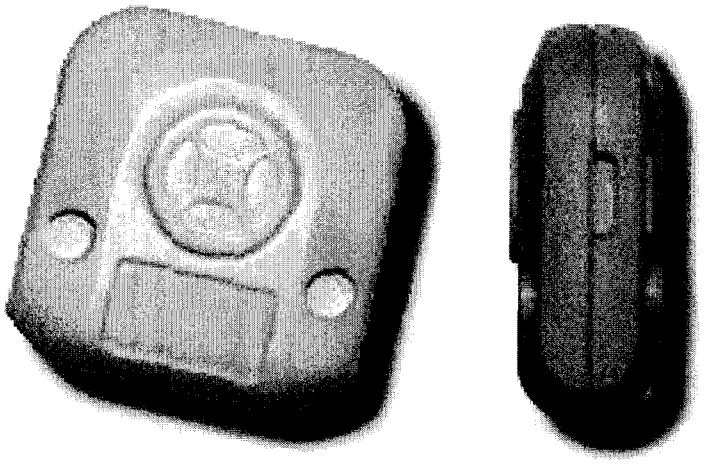

Figure 4.4: The RF tags carried by human participants and affixed to the robot truck.

logic and removed a source of error encountered in the visual tracking experiment.

In this way, the human participants could play tag in the usual fashion, and the system received regular position reports almost exactly as it had using the visual system, but with substantially increased flexibility. A comparison of the three sensor networks' performance is described in Section 5.9.1.

Chapter 5

Experimental Validation

5.1 Canned scenarios

Our first experiment explored the feasibility of the force-dynamic approach in qualitative fashion. We devised simple scenarios such as "Person A wants the toy, but Person B tries to prevent him from getting it," and asked subjects to act out the events described. We collected the motion trajectory data and generated animations and a descriptive narrative of the interactions it detected. This experiment and the following one (described in section 5.2) relied only on the earlier processing steps described above, through section 3.4.

We first asked nine human observers, undergraduate and graduate students between the ages of 18 and 29, to watch a series of five cartoons and describe what happens in their own words. Although the descriptions were somewhat idiosyncratic, most identified the same relationships between the pictured entities and noticed the same dramatic moments where those relationships change. For instance, for the interaction illustrated in Figure 5.3, one observer commented "The red guy wants to get the blue thing. First he just walks up, but then the green guy tries to get in the

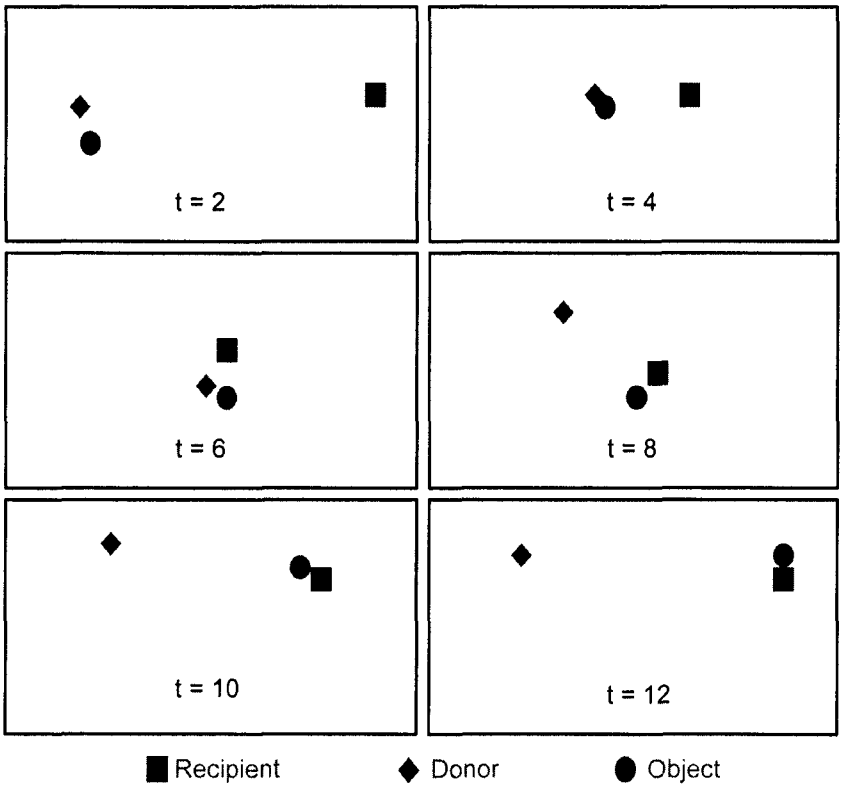

Figure 5.1: A scenario where the donor hands an object to the recipient. The object accompanies the donor as the two agents approach each other, and then leaves with the recipient as they walk away.

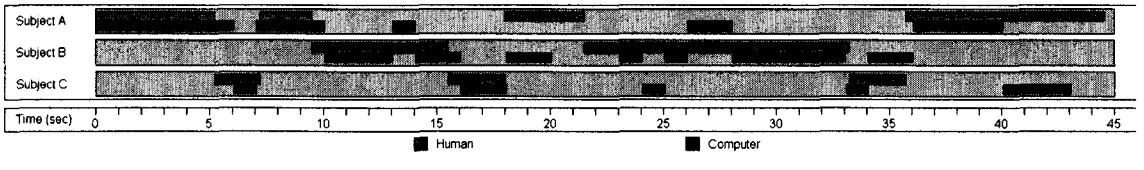

Figure 5.2: Robot's opinion of who is "IT" in a game of tag, compared to a single human's analysis of identical data. The agreement between human and robot is closest in this sample to the statistical average, out of twelve total (three human coders $\times$ four games).

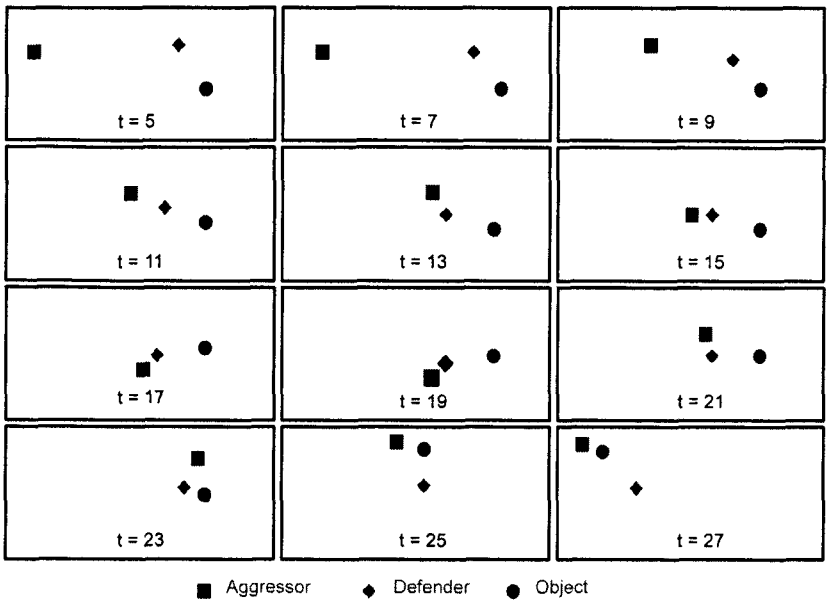

Figure 5.3: A scenario where the defender tries to prevent the aggressor from snatching the object, but ultimately fails. The object remains still while the aggressor approaches (from $t = 7$ to $t = 11$ seconds) and the defender moves to interpose itself (from $t = 11$ to $t = 21$ seconds). The aggressor obtains the object at $t = 23$ seconds and moves rapidly away, followed by the defender.

way. They dance around a bit, and then the red guy gets past, grabs the blue square and takes off. The green one chases him." Different subjects used slightly different vocabularies, but told the same story and noticed the same events. The green square "got between" or "fought off" or "tried to stop" the red one, while the red square, 23 seconds into the animation, always succeeded in "taking" or "stealing" the blue object.

We allowed the system to analyze the same data and produce its estimates of the time-varying constants and equations of motion which describe the agents' relationships with each other. We provided basic English phrases for the system to use to describe its conclusions, selected according to the values of the mathematically-determined constants. At every significant event (whenever the hypothesized vectors changed sign or a different set of equations produced a better fit), the robot pro-

duced a new phrase. For the scenario just described, Nico was able to produce the following: "Red approaches blue. Green approaches red and approaches blue. Red avoids green and approaches blue. Green approaches red and avoids blue. Red and blue avoid green. Green approaches red and approaches blue." After the human subjects produced their own scenario descriptions, they were asked to answer the question "Was Nico's summary accurate (yes or no)?" They responded positively 34 out of 45 times.

In order to obtain a more objective measure of how clear and interpretable our animations are, we provided a set of descriptions of the scenarios depicted in animations, as well as a few descriptions that did not correspond to any animation. For example, we described the scenario shown in Figure 5.1 as "Red gives green the blue toy." We asked our subjects to match descriptions and animations. All six subjects (three of the nine mentioned before did not participate in this portion of the experiment) chose exactly the same description for each of ten animated scenarios (five which they had already seen as described above and five which they had not), validating our assumption that humans are quite adept at understanding complex social interactions with very impoverished context and sensory data, and moreover different people's interpretations of the same data are largely consistent with one another.

5.2 Human and machine interpretation of tag

We devised our second experiment to take a closer, more quantitative look at the robot's performance in a longer, more complex motion scenario, and to give the robot the opportunity to participate meaningfully in the action. To that end, we chose to focus on watching and learning about the game of Tag. Since Nico is not ambulatory,

its participation in the game was of necessity somewhat stylized. However, it can still take appropriate actions that are readily interpreted as relevant to the game by human observers. While a game is being played, Nico looks at the person it currently judges to be "IT", while pointing to another player, the one farthest away from the robot itself – as if to say, "Hey, you, go over there. Get that guy, not me!" If the person Nico perceives as "IT" comes within two meters of itself, it assumes a defensive posture, throwing its arm over its face and looking away.

Figure 5.2 shows the robot's performance identifying intentions and game state over the course of a 45-second game. We also converted this game and others to an animation and showed them to three human subjects, who were instructed to indicate the identity of "IT" as participants tagged each other back and forth over the course of the game. One of these sequences is also shown in the figure for comparison. Compared across four games and three subjects, the robot and humans agreed 70.8% of the time. This compares with an agreement between the human coders alone of 78.5% – that is, when watching a cartoon of a typical 45-second game of tag, two humans will disagree with each other about the identity of "IT" for a half a second here, a couple seconds there, for a total of about 9.8 seconds of incongruity. Both the human and robot performances are much greater than random (33%).

The small disparity in performance likely arises from the fact that the robot does not yet have preconceived notions about how the game of Tag operates. Specifically, it cannot deduce that "IT" can only pass between players when they are next to each other, unless such a pattern emerges from the games it sees. Human observers know this *a priori*. Such a discrepancy is illustrated in Figure 5.4. At $t = 9$, blue (which had been chased by red) now moves toward green, while green runs away. Both human and robot judge that red has tagged blue and blue is now "IT". At $t = 13$,

however, blue is still chasing green a little, but red begins moving quickly toward the other two. The computer assumes this means red is now "IT", but the human observer knows that red was too far away to have been tagged. Shortly afterwards, red and blue start running away from green, and it is obvious to both human and robot that a new player has become "IT'. This shortcoming we address in the further experiments reported below.

5.3 A growing collection of games

For the next experiment, we instructed test subjects to play several more common playground games for periods of three minutes at a time. The sensor network tracked four objects during the period of play, games played either with four people or with three people and a ball. The system had no independent knowledge of which objects were human and which were inanimate. Six trials of each game were played, three each by two independent groups of people (undergraduate and graduate students). The games were:

- Tag: The person designated as "it" chases and tags one of the other players, at which point that player becomes "it" and the roles switch (Figure 5.5).

- Smear: One of the players carries a ball while the others attempt to catch and steal the ball for themselves. Whoever carries the ball is the target of everyone else in the game.

- Keepaway: Two players throw the ball back and forth, while a third tries to intercept the passes. If successful, the interceptor becomes a thrower and the person responsible for throwing the bad pass must now attempt to intercept (Figure 5.6).

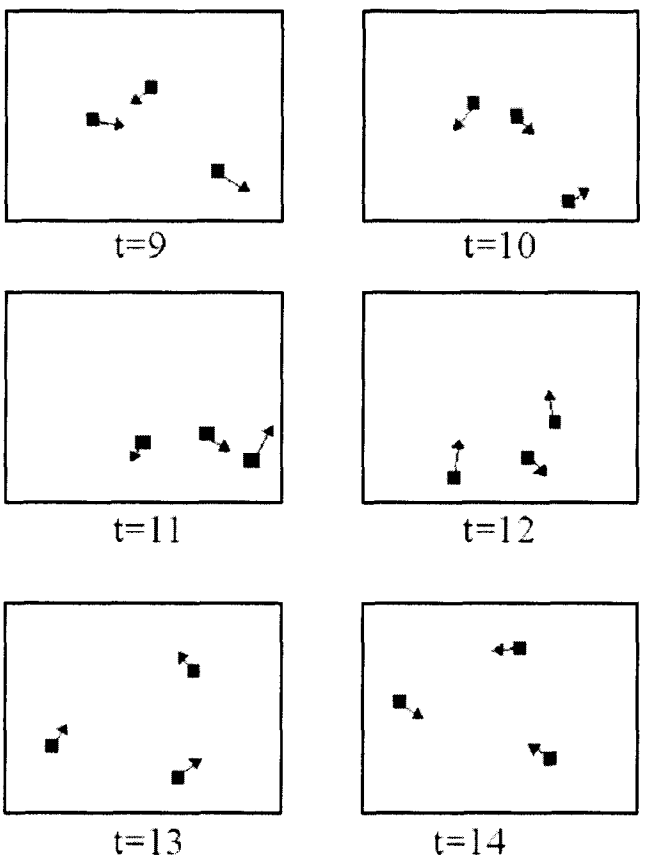

Figure 5.4: Stills excerpted from game represented in Figure 5.2, representing elapsed time between 9 and 14 seconds. The colored squares are exactly what the human coders see in the animations. The arrows have been added for clarity on the page. See text for details.

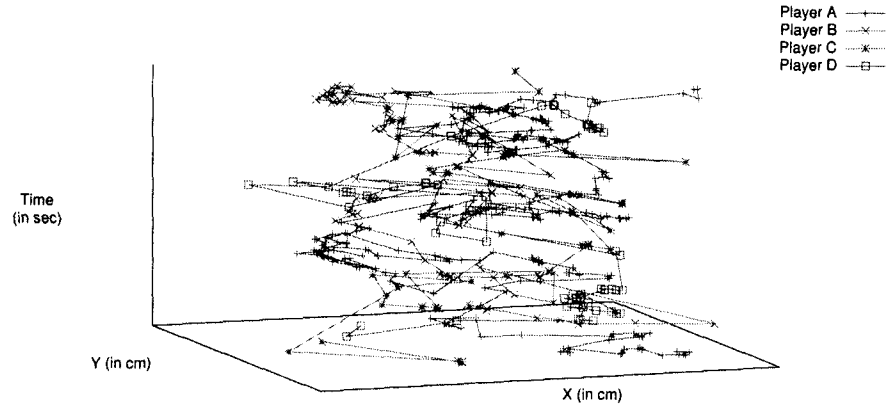

Figure 5.5: A four-person game of Tag. Each player's position is plotted in the x and y dimensions, with time increasing vertically. All four people are constantly in motion, looping about the room as they chase and are chased in turn.

Table 5.1: Similarity metrics (Dice's coefficient)

	Tag	Smear	Keepaway	Catch
Tag	0.80	0.62	0.38	0.28
Smear		0.72	0.54	0.30
Keepaway			0.66	0.66
Catch				0.86

- Catch: Each player throws the ball to one of the other two players in turn.

In comparison to the simulated games of Todd and his colleagues (Figure 2.2), the real-world data plots in Figures 5.5 and 5.6 are noisy and harder to interpret or classify. Nevertheless, the narrative-based classification scheme described here performs well.

5.4 Activity differentiation

For each game, we calculated the similarity metric, as described in Section 3.7, between it and each of the other games – five other instances of the same game

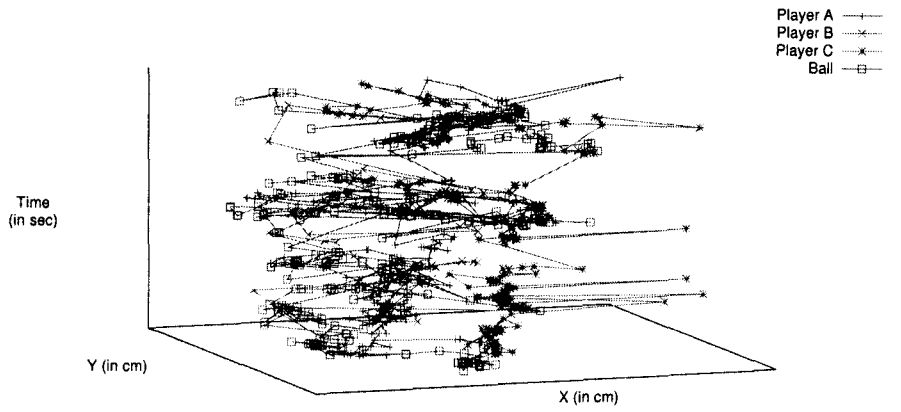

Figure 5.6: A three-person game of Keepaway. The ball is constantly in motion, as is *one* of the players, while the other two players are relatively stationary. The identity of the moving player changes several times during the game.

Table 5.2: Classification rates

Tag	85%
Smear	77%
Keepaway	53%
Catch	83%

and six of each of the others. Table 5.1 shows the average of these intra-and inter-game similarity metrics. Tag, Smear and Catch all were significantly more similar to themselves than to any other game. The exception, Keepaway, was indistinguishable from Catch. Interestingly, the converse was not true – although an average game of Keepaway is as different from other Keepaway games as from Catch, games of Catch are much more similar to each other than to Keepaway.

Table 5.2 depicts a performance measure based on classification rates. For each game, we looked at the five games that the system judged as most similar and scored how many of those were actually the same game. If performance was perfect, then for each of the six trials of each game, the system would pick out all five of the other games of the same type, leading to 30 correct answers per game. In none of

Table 5.3: Sample location statistics for Tag

	$\mu(d_{AB})$	$\mu(d_{AC})$	$\mu(d_{BC})$
A chasing → B chasing	21.6	104.8	184.3
A chasing → C chasing	58.3	18.0	112.5
B chasing → A chasing	28.4	90.5	83.8
B chasing → C chasing	107.9	141.6	21.4
C chasing → A chasing	217.9	29.3	191.4
C chasing → B chasing	106.7	137.1	29.3

the games was the performance this good, but all are well above random (22%). Keepaway once again underperforms.

These results show that the system was able to perceive the intentions of the game participants and construct narratives that captured essential similarities and differences between and among games. Given any two games, by hypothesizing about the intentions of the players and stringing together intentional states into a narrative, the system could usually tell whether the two scenarios told similar stories, and thus whether they were likely different versions of the same game or different games altogether. Interestingly, it even was able to work out that certain games, though different, were more closely related than others – Tag and Smear, which both involve chasing and tagging other people, looked more like each other than Keepaway and Catch, which both involve tossing a ball around, and vice versa.

5.5 Rule recognition

The average number of narrative state transitions detected by the system across the 24 3-minute games was 28 – not a huge corpus to analyze for location-based rules. However, the system already knows which scenarios are similar enough to be considered the same game, enlarging its pool of data significantly. For each game, the system averaged together the absolute and relative location statistics for the five

most similar games, found as described in the previous section. The system identified all of the following (by displaying noteworthy state transitions and associated measurements, which I have translated into text for illustration):

- Tag and Smear: when the role of chaser switches between players, those two people are close together.
- Smear: the two chased objects are always next to each other (presumably the ball and the ball carrier)
- Catch: three people are always standing in the same place whenever the ball changes direction.

Table 5.3 shows an example of the relative location data used to make these determinations. In the six games most similar to a particular exemplar of Tag, for every type of intentional state transition, the table depicts the average distance (in cm) between the three players. In every case, the two players who are swapping roles are located close to one another.

A fairly straightforward extension to the system at this point would be to take these discovered rulesets and reapply them to the classification problem. At present, the system leverages classification on the basis of intentional state transitions to extract a dataset of similar games, which it then uses to search for rules. Having found such rules, it could go back into the original data and attempt to classify on the basis of the rules as well as the intentions. In this way, for example, having discovered that catch involves a great deal more standing still than does keepaway, it should be able distinguish the two better than it currently does. As currently constituted, however, the system does not do this.

5.6 Robot participation

We tested the system in a 20x20-foot lab space with an open floor. We ran trials on three separate occasions, with two human subjects driving the red and green remote-controlled cars and the system controlling the blue one. We also ran one additional control trial with three human drivers and no robot-controlled car. The subjects themselves were in the room with the vehicles, but seated against the wall behind the camera's field of view. Each set of trials involved different people as drivers. The drivers were instructed to play the game as well as they could, and that the robot participant did not begin the game knowing how to play. Accordingly, they should engage the robot only once it appeared to be acting appropriately, ignoring it at first. Data from the first experiment were collected during each trial; the final experiment involving modified tag was conducted only during the last trial.

5.7 Chasing and following

The first game we tested was simple. Each player had only one unchanging goal. The driver of the red car was asked to stay as far away from the others as possible, while the green car gave chase. In each trial, the behavior of the system was consistent. Within less than a second, the system determined the intentional states of Green and Red with respect to each other. It then proceeded to generate and test hypotheses regarding their intentions toward itself, by approaching each of the two cars. Within a few seconds more, it was able to determine that Red was fleeing from both, and Green was indifferent to Blue. Since the intentional state never changed, no positional information was ever recorded or analyzed.

The fact that the robot can *participate* in the game provides it with significant added power to probe the players' intentional states. For comparison, we also ran

Table 5.4: Results from Chase and Follow

	No. of Games	Average Time	σ
Chase	6	7.5 sec	1.41
Follow	4	33.5 sec	4.66
Chase (observe only)	3	29.3 sec	10.69

versions of the game that involved three human drivers, relegating the robot system to the role of passive observer. Still, the system applied the same algorithms to hypothesize the intentions of the players, and eventually converged on a stable, correct belief state. But it took nearly four times as long, on average: 29.3 seconds as opposed to 7.5.

The second game, Follow the Leader, increased the complexity of the game. The driver of the red car was instructed to drive wherever he or she wished, and the green car was to follow, but remain a foot or two away – stopping when the red car stopped, reversing if it got too close. Success in this game came when the system understood this: it should approach the red car from across the room, but avoid it close in. In this game, the system was only successful in four runs of the game, out of six. In both of the other two trials, it formed the belief that the game was Chase, just as in the previous experiment, and never noticed the change from following to ignoring or avoiding.

5.8 Modified tag

Having confirmed that the system was able to understand and participate in simple games, we asked our subjects to play the somewhat more sophisticated game of tag. In previous research that involved the system merely watching people play, rather than attempting to participate, we enjoyed a great deal of success [15, 19]. However, several factors conspired against us. The RC cars are not nearly as agile

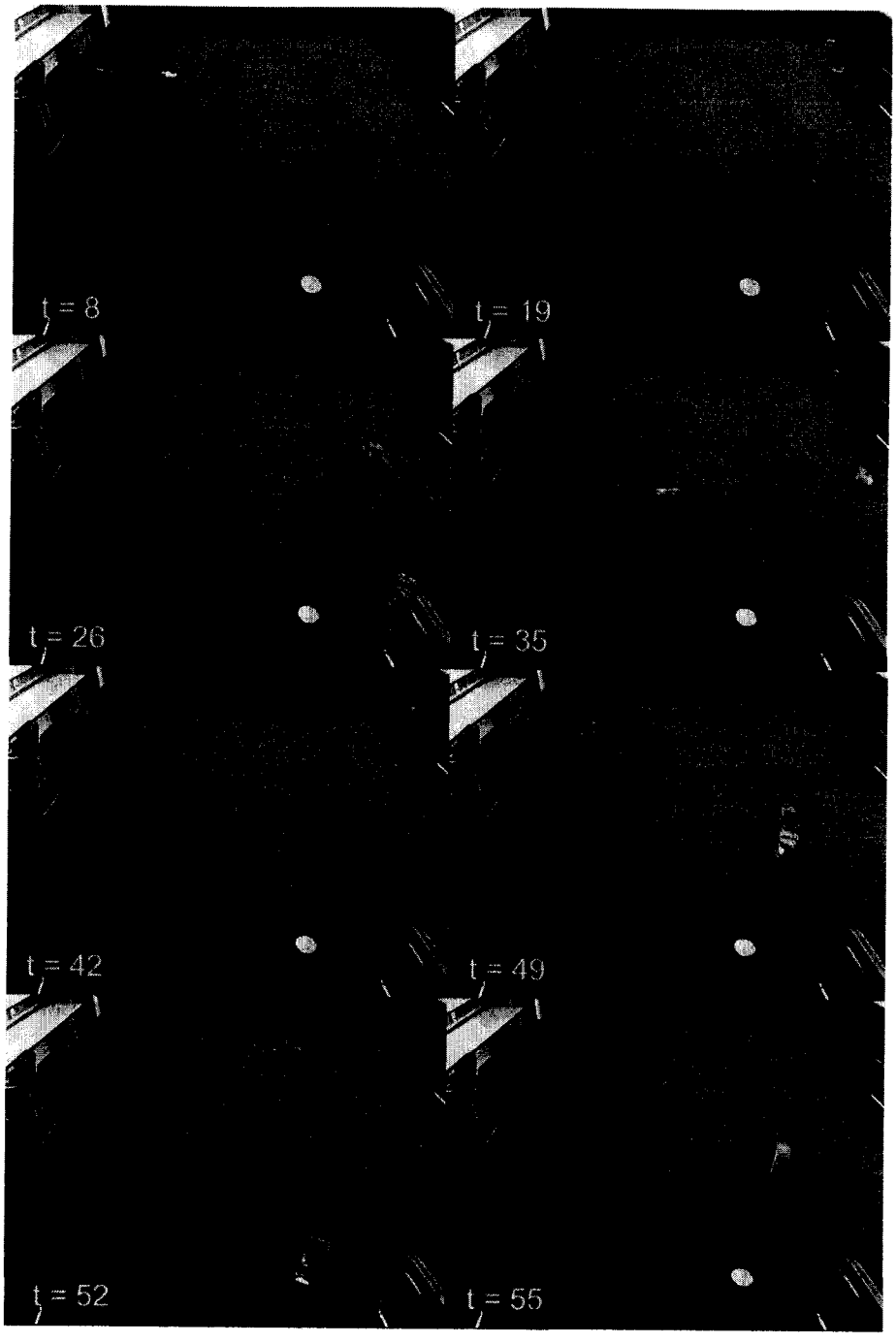

Figure 5.7: Succession of images from modified tag. See text and Table 5.5 for details.

as actual humans, and our subject drivers had significant difficulties controlling the vehicles well enough to conduct the game. In addition, one of the three participants in the game – the robot – had no idea how it should be played (which, after all, was something of the point). The two human players were unable to demonstrate gameplay adequately by themselves. We asked a pair of students unconnected to our tests to watch the videos of the tag attempts, and neither of them were able to identify the game being played, either.

Since freeform tag was too difficult for all involved, we developed rules for a tag-like game in order to test the system's ability to understand turn-taking and role shifts within the context of a game. In the modified game, only one person was supposed to move at a time. The player designated as "it" picked a victim, moved toward it, tagged and retreated. Then the new "it" repeated the process. In other words, we maintained the rules of tag, but slowed down the gameplay and simplified the task of determining what each participant was attempting to do at any particular time, by introducing turn-taking. Figure 5.7 depicts a set of stills from one of these modified tag games. A frame-by-frame description of the progress of the game is depicted in Table 5.5.

At each time point, the table includes a human-constructed verbal description of the action of the game, as well as the textual description produced by the system itself. This comes from the robot's own actions (which it knows absolutely and need form no beliefs about), and its belief in the intentional states of the players during a particular narrative episode. At the start, the robot watches the other two players each tag one another, without participating. Then, not knowing what its own role in the game is, it begins to move toward and away from the other players, observing their reactions. Because both of the human players are currently ignoring the robot, these actions are inconclusive. However, by second 42, the system has accumulated

Table 5.5: Action and narrative during modified tag. Each time point corresponds to a still from Figure 5.5.

t (sec)	Action Description	System narrative
8	R approaches and tags G	R chases G.
19	R withdraws	G chases R, and
	G approaches and tags R	R runs from G.
26	G withdraws	R chases G, and
	R approaches and tags G	G runs from R and
	B approaches G and R	me, and I chase G.
35	R withdraws	G chases R, and
	G approaches R	R runs from G, and
	B approaches G and R	I chase R.
42	G tags R	G chases R, and
	B approaches G and R	I chase R.
49	G and B run from R	R chases G, and
	R tags G	G runs from R, and
		I run from R.
52	R withdraws	R chases G, and
	B approaches and tags R	G runs from R, and
		I run from R.
55	G approaches B and R	R runs from me, and
	B withdraws	G runs from R, and
		I chase R.

enough data to know that intentional shifts are signalled by close proximity. In the fifth frame, it recognizes the tag and reverses its own direction at the same time. By the seventh frame, it is testing to see whether approaching the red car will cause it to reverse course. By the end of this sequence, the system still has not determined that there is only one player ("it") with the chasing role, but it is well along the way – it understands tagging and the ebb and flow of pursuit and evasion.

5.9 Real tag through denser sensing

Experiments up to this point have shown that the system has a substantial ability to develop hypotheses with respect to the intentions and goals of the people it watches,

Table 5.6: Accuracy and precision of various tracking systems

Sensor type	Static CEP-90 (cm)	Update average (Hz)	Spurious rate	Jerk ($\lvert\frac{cm}{sec^3}\rvert$)
Ultrasound	2.2	1.61	4.15%	4040
Visual	0.9	15.00	1.80%	0.634
Radio	2.6	4.84	0.19%	0.00451

to consider how those evolving intentions coalesce into a narrative, and how the rules to a game can be reverse-engineered from the story that emerges. Adding the ability to participate increases the speed at which information is learned, and demonstrates that the system is able to act appropriately in complex social contexts. But it still hasn't really been able to play tag, due to the shortcomings described in the previous section.

5.9.1 Tracking system comparison

Our final experiment foregoes the vision system in favor of a radio sensor network described in Section 4.8. This raises the total number of position tracking systems used in the work to three, and a brief digression to account for the differences of each is warranted. A summary of the perfomance characteristics of the various systems is summarized in Table 5.6.

The static CEP-90 statistic is the radius of the circle, centered on an object sitting still in a known location, that contains 90% of the position reports emanating from the object. All three sensor systems do well in this respect. When in a dynamic environment, however, the story changes drastically. The ultrasound signal is by far the least robust of the three; as it moves about the room, the ultrasonic chirps have a tendency to be blocked, to arrive after a multipath propagation, or to be misidentified. Thus, although each system nominally produces at least 5 position reports a second, the Cricket system misses many. In contrast, the radio sensor

network is far more reliable – missed reports only happen due to rare radio collisions. And the video system processes each frame to find the color maxima, never missing a one.

We obtained the rate of spurious data from a random sampling of data from the three sources. A human observer watched each trajectory in slow motion, and flagged position reports that made no sense – sudden flashes across the room, say. The ultrasonic signals were far more susceptible to this kind of error. Finally, as an objective measure of the stability and accuracy of position reports, we calculated the third derivative of the position reports taken from all of the trials – the "jerk" – and averaged their absolute values. This is a measure of how shaky or twitchy the trajectories are, and varies across the sensor systems by seven orders of magnitude.

5.9.2 A robot learns to play tag

With the RF tracking system in place, we collected data in fifteen separate trials on three different occasions. On each occasion, three people drawn from a population of undergraduates, graduate students and postdocs were tasked with playing ordinary games of playground tag in a large open laboratory while carrying a uniquely identified radio tag. Each time, the participants played two three-minute games by themselves, with the system merely observing, and three games that included the robot-controlled vehicle as a fourth participant.

To evaluate the success of the robot's observation, intention recognition, rule interpretation amd ultimately gameplay, we observed the sequence of narrative states produced by the system during the scenario. At first, of course, the robot knows nothing about the interactions it observes, and the set of intentional states it attributes to the participants are completely arbitrary. But eventually, as it observes behavior and begins to support its hypotheses with evidence, the reported states

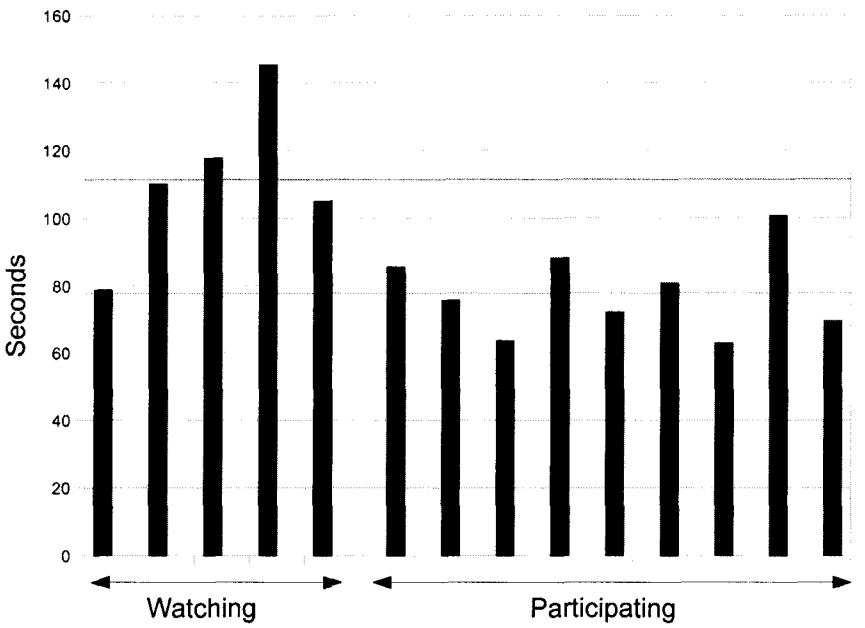

Figure 5.8: Point (in seconds) at which the robot figured out the rules to tag, in fifteen independent trials. In blue, the trials where the robot observed without participating; in orange, the trials in which the robot played along. One blue trial is missing because the system entirely failed to discover the rules to the game. The horizontal lines denote the mean.

begin to be consistent with the rules of Tag.

Figure 5.8 reports the points at which the system reaches a point where the intentional states it believes to hold among the players are entirely consistent with the actual game of Tag – in other words, the time when the ensuing narrative consists only of events of the form "A is chasing someone, and everyone else is avoiding A." Prior to the reported time of success, the system periodically indulges in belief states which do not reflect the rules of Tag, such as imagining two simultaneous chasers.

In fourteen of the fifteen trials, the system successfully learned the rules to tag, though in one case (the worst performance illustrated), it held its correct beliefs for only 34.5 seconds before the game ended, and may well have come to other wrong conclusions had the game lasted longer. In any case, in this experiment, as in previous ones, participation led to substantially better performance, coming to the correct conclusion in all cases and in an average of 33.75 seconds faster.

Chapter 6

Conclusion

Humankind, it has been suggested [71], should more appropriately be called *Homo narrans*: storytelling man. We are driven to tell stories to make sense of the world, even or perhaps especially when we have only sparse information to go on. In this work, I present a computational system that begins to demonstrate how a set of inferences and conclusions can be drawn from rudimentary motion data, leading to largely correct judgments about activity classification and rule inference. To do this, it hypothesizes about human intentions and constructs narratives around real-world social interaction. The better the story it can tell, the more powerful the conclusions it can draw about the activities it sees.

I have also shown that, given this impoverished data full of motion cues, machines are able to draw many of the same conclusions and tell the same kinds of stories that humans do. Furthermore, the narrative hypotheses are constructed in real time from real-world motion data. The verisimilitude of the data thus collected enables us to draw stronger conclusions with respect to real human interaction and interpretation, in contrast to data derived from simulation or computer-mediated play.

Understanding and making sense of gross motion comes early in childhood de-

velopment. I have demonstrated that it is possible to begin to draw sophisticated conclusions about actions, events and social dynamics, without relying on a rich experiential background or even detailed information coming in from the visual system – a sense of relative location between distinct blobs is enough. Thus we are modeling a primitive and early-emerging cognitive social skill at a very basic level, needing to make few assumptions that stretch either biological plausibility or real-world accuracy.

Biological beings excel at making snap decisions and acting in a complex world using noisy sensors providing information both incomplete and incorrect. In order to survive, humans must engage and profit from not only their physical environment, but a yet-more-complex social mileu erected on top. One of our most powerful and flexible cognitive tools for managing this is our irrepressible drive to tell stories to ourselves and to each other. This is true even or perhaps especially when we have only sparse information to go on. And beyond the telling, we take great delight in participating. We play games, we act, and the stories we love most are the ones in which we are the central characters.

I have developed a system that takes advantage of the very fact that it receives only rudimentary sensory impressions, and uses them to weave a story in which it can take part. The relative positions of moving objects are more than enough data for a human observer to begin making sense of the interaction by imagining their intentions and goals. By applying force dynamics to hypothesize about such human intentions, and by acting on those hypotheses to explore and verify its beliefs about the world, our system attempts to do the same.

The system has to figure out for itself how its motor controls correspond to action in the world. It theorizes about and tries to learn the intricate rules to games it knows nothing about. The verisimilitude of the data thus collected enables us to

draw stronger conclusions with respect to real human interaction and interpretation, in contrast to data derived from simulation or computer-mediated play. And it does it in the real world, in real time, at human speed, using few shortcuts.

The cognitive model I have built reflects very low-level computational processing of the same kinds of nearly context-free percepts that have been used to probe basic intentionality and animacy reactions in humans. Throughout the dissertation, I have relied on terms such those, because it is these phenomena that the model is intended to elucidate. How useful a role can this kind of low-level information processing play in interpreting and responding appropriately to complex social behavior? My system demonstrates that a certain success is possible without relying on a great deal of experience and context. I have shown that psychophysics itself, or rather, a computational model thereof, is sufficient to form the basis for a control system that can act appropriately in a social context.

Bibliography

[1] L. Aakerlund and R. Hemmingsen. Neural networks as models of psychopathology. *Biological Psychiatry*, 43:471–482, 1998.

[2] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[3] Aron K. Barbey and Phillip Wolff. Learning causal structure from reasoning. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society*, 2007.

[4] H. Clark Barrett, Peter M. Todd, Geoffrey F. Miller, and Philip W. Blythe. Accurate judgments of intention from motion cues alone: A cross-cultural study. *Evolution and Human Behavior*, 26:313–331, 2005.

[5] J. Bassili. Temporal and spacial contingencies in the perception of social events. *Journal of Personality and Social Psychology*, 33:680–685, 1976.

[6] S.-J. Blakemore, P. Boyer, M. Pachot-Clouard, A. Meltzoff, C. Segebarth, and J. Decety. The detection of contingency and animacy from simple animations in the human brain. *Cerebral Cortex*, 13:837–844, 2003.

[7] Maurizio Borsotto, Weihong Zhang, Emir Kapanci, Avi Pfeffer, and Christopher Crick. A junction tree propagation algorithm for bayesian networks with

second-order uncertainties. In *18th IEEE International Conference on Tools with Artificial Intelligence*, 2006.

[8] J. Breese and K. Fertig. Decision making with interval influence diagrams. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*, 1991.

[9] Andrew G. Brooks, Jesse Gray, Guy Hoffman, Andrea Lockerd, Hans Lee, and Cynthia Breazeal. Robot's play: interactive games with sociable machines. *Comput. Entertain.*, 2(3):10–10, 2004.

[10] R. A. Chambers, M. N. Potenz, R. E. Hoffman, and W. L. Miranker. Simulated apoptosis/neurogenesis regulates learning and memory capabilities of adaptive neural networks. *Neuropsychopharmacology*, 29:747–758, 2004.

[11] Hoon Choi and Brian J. Scholl. Perceiving causality after the fact: Postdiction in the temporal dynamics of causal perception. *Perception*, 35:385–399, 2006.

[12] Committee on Networked Systems of Embedded Computers. Embedded everywhere: A research agenda for networked systems of embedded computers. Computer Science and Telecommunications Board, National Research Council, 2001.

[13] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief network. *Artificial Intelligence*, 42:393–405, 1990.

[14] F. G. Cozman. Credal networks. *Artificial Intelligence*, 120:199–233, 2000.

[15] Christopher Crick, Marek Doniec, and Brian Scassellati. Who is it? inferring role and intent from agent motion. In *Proceedings of the 11th IEEE Confer-*

ence on Development and Learning, London UK, 2007. IEEE Computational Intelligence Society.

[16] Christopher Crick and Willard Miranker. Apoptosis, neurogenesis and information content in hebbian networks. *Biological Cybernetics*, 94(1):9–19, 2006.

[17] Christopher Crick, Matthew Munz, and Brian Scassellati. Robotic drumming: synchronization in social tasks. In *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication*, 2006.

[18] Christopher Crick and Avi Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003.

[19] Christopher Crick and Brian Scassellati. Inferring narrative and intention from playground games. In *Proceedings of the 12th IEEE Conference on Development and Learning*, Monterey CA, 2008. IEEE Computational Intelligence Society.

[20] Christopher Crick and Brian Scassellati. Intention-based robot control in social games. In *Proceedings of the Cognitive Science Society Annual Meeting*, 2009.

[21] Gergely Csibra, Gyorgy Gergely, Szilvia Biro, Orsolya Koos, and Margaret Brockbank. Goal attribution without agency cues: the perception of 'pure reason' in infancy. *Cognition*, 72:237–267, 1999.

[22] A. Darwiche and G. Provan. Query dags: a practical paradigm for implementing belief-network inference. *Journal of Artificial Intelligence Research*, 6:147–176, 1997.

[23] Verena Dasser, Ib Ulbaek, and David Premack. The perception of intention. *Science*, 243:365–367, 1989.

[24] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In *Proceedings of the 16th European Conference on Artificial Intelligence*, 2004.

[25] R. Dechter, R. Mateescu, and K. Kask. Tree approximation for belief updating. In *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002.

[26] Lee R. Dice. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302, 1945.

[27] Marek W. Doniec, Ganghua Sun, and Brian Scassellati. Active learning of joint attention. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids 2006)*, 2006.

[28] C Drake, M R Jones, and C Baruch. The development of rhythmic attending in auditory sequences: attunement, referent period, focal attending. *Cognition*, 77(3):251–88, 2000.

[29] H. F. Durrant-Whyte and M. Stevens. Data fusion in decentralized sensing networks. In *Proceedings of the 4th International Conference on Information Fusion*, 2001.

[30] G. Elidan. Bayesian network repository. http://www.cs.huji.ac.il/labs/compbio/Repository.

[31] P. S. Erikson, E. Perfilieva, T. Bjork-Eriksson, A. M. Alborn, C. Nordburg, D. A. Peterson, and F. H. Gage. Neurogenesis in the adult human hippocampus. *Nature Medicine*, 4:1313–1317, 1998.

[32] E. Fagiuoli and M. Zaffalon. 2u: an exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, 1:77–107, 1998.

[33] W. T. Freeman, J. S. Yedida, and Y. Weiss. Generalized belief propagation. In *Proceedings of the Neural Information Processing Systems Conference*, 2000.

[34] R. Fung and K. Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. In *Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence*, 1989.

[35] Tao Gao, George E. Newman, and Brian J Scholl. The psychophysics of chasing: a case study in the perception of animacy. *Cognitive Psychology*, 59:154–179, 2009.

[36] M. S. Gazzaniga, R. B. Ivry, and G. R. Mangun. *Cognitive neuroscience.* Norton, 2002.

[37] R. Gelman, F. Durgin, and L. Kaufman. *Causal cognition: A multidisciplinary debate*, chapter Distinguishing between animates and inanimates: Not by motion alone, pages 150–184. Clarendon Press, Oxford, 1995.

[38] Gyorgy Gergely, Zoltan Nadasdy, Gergely Csibra, and Szilvia Biro. Taking the intentional stance at 12 months of age. *Cognition*, 56:165–193, 1995.

[39] Gerd Gigerenzer and Peter M. Todd. *Simple heuristics that make us smart.* Oxford University Press, New York NY, 1999.

[40] K. Gold and B. Scassellati. Learning about the self and others through contingency. In *AAAI Spring Symposium on Developmental Robotics*, Palo Alto, CA, 2005. AAAI.

[41] Kevin Gold and Brian Scassellati. Grounded pronoun learning and pronoun reversal. In *5th International Conference on Development and Learning (ICDL-06)*, 2006.

[42] Eugenia Goldvarg and P. N. Johnson-Laird. Naive causality: a mental model theory of causal meaning and reasoning. *Cognitive Science*, 25:565–610, 2001.

[43] E. Gould, A. Beylin, P. Tanapat, A. Reeves, and T. Shors. Learning enhances adult neurogenesis in the hippocampal formation. *Nature Neuroscience*, 2:260–265, 1999.

[44] A. Z. Hajian, D. S. Sanchez, and R. D. Howe. Drum roll: Increasing bandwidth through passive impedance modulation. In *Proceedings of the 1997 International Conference on Robotics and Automation*, 1997.

[45] Erin E Hannon and Scott P Johnson. Infants use meter to categorize rhythms and melodies: Implications for musical structure learning. *Cognitive Psychology*, 50(50):354–377, 2005.

[46] Simon Haykin. *Neural Networks. A Comprehensive Foundation.* Prentice Hall, New Jersey, USA, 1999.

[47] Andrea S. Heberlein and Ralph Adolphs. Impaired spontaneous anthropomorphizing despite intact perception and social knowledge. *Proceedings of the National Academy of Sciences*, 101:7487–7491, 2004.

[48] F. Heider and M. Simmel. An experimental study of apparent behavior. *American Journal of Psychology*, 57:243–259, 1944.

[49] J. Hill, R. Szewczyk, A. Woo, W. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *Proceedings of the 9th Inter-*

national Conference on Architectural Support for Programming Languages and Operating Systems, 2000.

[50] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[51] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *Proceedings of the 9th Annual Symposium on Network and Distributed System Security*, 2002.

[52] F. V. Jensen. *Bayesian networks and decision graphs.* Springer-Verlag, 2001.

[53] F. V. Jensen and S. K. Andersen. Approximations in bayesian belief universes for knowledge-based systems. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, 1990.

[54] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

[55] I Kato, S Ohteru, K Shirai, S Narita, S Sugano, T Matsushima, T Kobayashi, and E Fujisawa. The robot musician wabot-2 (waseda robot-2). *Robotics*, 3(2):143–155, 1987.

[56] Ami Klin. Attributing social meaning to ambiguous visual stimuli in higher functioning autism and asperger syndrome: The social attribution task. *Journal of Child Psychology and Psychiatry*, 41:831–846, 2000.

[57] David R. Kornack and Pasko Rakic. The generation, migration, and differentiation of olfactory neurons in the adult primate brain. *Proceedings of the National Academy of Sciences*, 98:4751–4757, 2001.

[58] Jure Kovac, Peter Peer, and Franc Solina. Human skin colour clustering for face detection. In *Proceedings of the 2003 International Conference on the Computer as a Tool*, 2003.

[59] K. E. Kyburg. Bayesian and non-bayesian evidential updating. *Artificial Intelligence*, 31(3):271–293, 1987.

[60] Alan M. Leslie, Fei Xu, Patrice D. Tremoulet, and Brian J. Scholl. Indexing and the object concept: developing 'what' and 'where' systems. *Trends in Cognitive Sciences*, 2(1):10–18, 1998.

[61] Jeff Loucks and Dare Baldwin. Sources of information in human action. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 121–126, Austin TX, 2008. Cognitive Science Society.

[62] E. A. Makakis and F. H. Gage. Adult-generated neurons in the dentate gyrus send axonal projections to field ca3 and are surrounded by synaptic vesicles. *Journal of Comparative Neurology*, 406:449–460, 1999.

[63] Raymond A. Mar and C. Neil Macrae. Triggering the intentional stance. In Greg Block and Jamie Goode, editors, *Empathy and Fairness*, pages 110–132. John Wiley & Sons, Chichester, UK, 2006.

[64] R. J. McEliece, D. J. C. Mackay, and J. F. Cheng. Turbo decoding as an instance of pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communication*, 16(2):140–152, 1998.

[65] P Michel, K Gold, and B Scassellati. Motion-based robotic self-recognition. In *IEEE-RAS/RSJ International Conference on Humanoid Robots*, Sendai, Japan, 2004. IEEE.

[66] A. Michotte. *La perception de la causalite.* Institut Superior de Philosophie, Louvain, 1946.

[67] A. Michotte. *Feelings and emotions: The Mooseheart symposium*, chapter The emotions regarded as functional connections, pages 114–125. McGraw-Hill, New York, 1950.

[68] Stephen R. Mitroff and Brian J. Scholl. Forming and updating object representations without awareness: evidence from motion-induced blindness. *Vision Research*, 45:961–967, 2004.

[69] Ranjan Mukherjee. Feedback control strategies for a nonholonomic mobile robot using a nonlinear oscillator. *Journal of Robotic Systems*, 16(4):237–48, April 1999.

[70] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, 1999.

[71] John D. Niles. *Homo narrans: the poetics and anthropology of oral literature.* University of Pennsylvania Press, Philadelphia, 1999.

[72] Franz Josef Ochs and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 2000.

[73] Arthur W.J.G. Ord-Hume. *Pianola : the history of the self-playing piano.* George Allen and Unwin, 1984.

[74] Judea Pearl. *Probabilistic reasoning in intelligent systems.* Morgan Kaufmann, 1988.

[75] Judea Pearl. *Causality: models, reasoning, and inference.* Cambridge University Press, 2000.

[76] Nissanka Bodhi Priyantha. *The cricket indoor location system.* PhD thesis, Massachusetts Institute of Technology, 2005.

[77] Philippe Rochat, Tricia Striano, and Rachel Morgan. Who is doing what to whom? young infants' developing sense of social causality in animated displays. *Perception*, 33:355–369, 2004.

[78] Michael Rosenblum and Arkady Pikovsky. Synchronization: from pendulum clocks to chaotic lasers and chemical oscillators. *Contemporary Physics*, 44(5):401–416, 2003.

[79] M. D. Rutherford, Bruce F. Pennington, and Sally J. Rogers. The perception of animacy in young children with autism. *Journal of Autism and Developmental Disorders*, 36:983–992, 2006.

[80] Brian J. Scholl. Can infants' object concepts be trained? *Trends in Cognitive Sciences*, 8(2):49–51, 2004.

[81] Brian J. Scholl and Patrice D. Tremoulet. Perceptual causality and animacy. *Trends in Cognitive Sciences*, 4(8):299–309, 2000.

[82] R. D. Shachter and M. A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence*, 1989.

[83] S. E. Shimoni. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68:399–410, 1994.

[84] J. Shneidman, B. Choi, and M. Seltzer. Collecting data for one hundred years. Technical report, Division of Engineering and Applied Science, Harvard University, 2002.

[85] Jeffrey Mark Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90, 2001.

[86] Jeffrey Mark Siskind. Reconstructing force-dynamic models from video sequences. *Artificial Intelligence*, 151:91–154, 2003.

[87] Steven Sloman, Aron K. Barbey, and Jared M. Hotaling. A causal model theory of the meaning of cause, enable, and prevent. *Cognitive Science*, 33:21–50, 2009.

[88] Claudio C V Staut and Thomas P Naidich. Urbach-wiethe disease (lipoid proteinosis). *Pediatric Neurosurgery*, 28:212–214, 1998.

[89] Pete Steggles and Stephan Gschwind. The ubisense smart space platform. In *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, volume 191, pages 73–76, 2005.

[90] Ganghua Sun and Brian Scassellati. Reaching through learned forward model. In *IEEE-RAS/RSJ International Conference on Humanoid Robots*, Santa Monica, CA, 2004. IEEE.

[91] Leonard Talmy. Force dynamics in language and cognition. *Cognitive Science*, 12:49–100, 1988.

[92] B. Tessem. Interval probability propagation. *International Journal of Approximate Reasoning*, 7:95–120, 1992.

[93] Toyota. Toyota partner robot. http://www.toyota.co.jp/en/special/robot/, 2006.

[94] Sandra E Trehub. The developmental origins of musicality. *Nature Neuroscience*, 6(7):669–73, 2003.

[95] Patrice D. Tremoulet and Jacob Feldman. Perception of animacy from the motion of a single object. *Perception*, 29:943–951, 2000.

[96] Patrice D. Tremoulet and Jacob Feldman. The influence of spatial context and the role of intentionality in the interpretation of animacy from motion. *Perception & Psychophysics*, 68(6):1047–1058, 2006.

[97] P. Walley. *Statistical reasoning with imprecise probabilities.* Chapman and Hall, 1991.

[98] Thalia Wheatley, Shawn C. Milleville, and Alex Martin. Understanding animate agents: Distinct roles for the social network and mirror system. *Psychological Science*, 18:469–474, 2007.

[99] Matthew Williamson. *Robot Arm Control Exploiting Natural Dynamics.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1999.

[100] S. Wirth, M. Yanike, L. M. Frank, A. C. Smith, E. N. Brown, and W. A. Suzuki. Single neurons in the monkey hippocampus and learning of new associations. *Science*, 300:1578–1581, 2003.

[101] P Wolff. Representing causation. *Journal of Experimental Psychology*, 136:82–111, 2007.

[102] Daniel M Wolpert, Kenji Doya, and Mitsuo Kawato. A unifying computational framework for motor control and social interaction. *Philisophical Transactions of the Royal Society B*, 358(1431):593–602, March 2003.

Appendix A

Loopy Belief Propagation as a Basis for Communication in Sensor Networks [18]

A.1 Introduction

Sensor networks are an exciting new kind of computer system. They consist of a large number of tiny, cheap computational devices distributed in an environment. The devices gather data from the environment in real time. Some data processing occurs in real time within the network itself; other data is shipped to a server for offline processing. In some cases the devices react online to the state of the environment.

One of the central questions in sensor networks is what to do with the data. When the data is to be processed online within the network what form should the information take, how should it be computed, and how should it be communicated? When nodes need to react to the information online, how can we ensure that each node receives the information it needs? In addition, how should the overall flow of

information be organized? All this needs to be accomplished at minimal cost and in a distributed fashion.

Consider, for example, the task of monitoring a building for outbreak of fire. A set of temperature sensors will be deployed throughout a building. Accurately detecting fire requires combining information from multiple sensors. For example, if a fire breaks out midway between two sensors, combining slightly elevated temperature readings at each of the sensors can provide a much quicker response than waiting until a single sensor has a very high reading. In addition, sensors that are physically deployed for a long time in an environment are subject to multiple kinds of failure. They may provide noisy readings, or they may break down completely. Combining information from multiple sensors can overcome these types of failures. In this application, we would like an immediate online response to occur as soon as a fire is strongly believed to be happening. Ideally, the sensor information would be combined online, to produce a quick and accurate response. How is this to be done?

This chapter argues that the Uncertainty in Artificial Intelligence (UAI) community can provide good answers to these questions. In many applications, like fire monitoring, the key task is to form beliefs about the state of the system based on the collected sensor readings. Since the environments in which sensor networks are dployed typically have a great deal of uncertainty, this is a core UAI task.

In particular, we argue that loopy belief propagation (LBP) is an ideal computational and communication framework for sensor networks. LBP has emerged as one of the leading methods for approximate inference in graphical models. It has properties that make it naturally suited for the sensor network domain. It can easily be implemented as a distributed algorithm, and the processing performed at each node is very simple and can be implemented cheaply on a tiny device.

The sensor network application presents many challenges to the LBP framework

that have not been encountered in previous applications. We investigate experimentally whether LBP is able to withstand some of these challenges. First, algorithms for sensor networks should be asynchronous. Attempting to enforce synchrony and a particular order of processing would be costly, and could lead to a loss of robustness if one step of processing fails. The first step of our investigation is to confirm that LBP does not rely on a synchronized order of message passing, but works just as well in an asynchronous environment in which each node communicates intermittently. Second, sensor networks often consist of devices of vastly different size and computational capability, and in addition the devices deployed in a physical system may be at very different levels of functionality. As a result, we would expect that in a deployed system, there will be nodes that compute and communicate far more frequently than others. We show that LBP continues to peform well even in highly asynchronous systems with vastly different communication rates. Third, nodes in a sensor network are subject to failure. Good sensor networks are designed with redundancy to allow for such failure. We show that LBP can exploit such redundancy to perform well even as nodes fail, and that it enjoys a graceful degradation property. Fourth, in LBP, beliefs gradually converge to the correct beliefs after a change in sensor readings. In a dynamic setting, it is possible that the environment might change again before the beliefs have had a chance to converge. One might suspect that this would lead to an unstable system, where the beliefs never track the truth. We show that this is not the case, and that LBP continues to perform well even when we expect many environmental changes to occur in the time it takes beliefs to converge.

As a result of our experiments, we assert that LBP is a strong candidate to be a basis for computation and communication in sensor networks. It is semantically well-founded, computing correct beliefs from sensor readings, relative to a probabilistic model. It is simple enough to be deployed into a wide variety of domains. Most

important, it enjoys a number of properties that allow it to cope with the stresses of deployment in a dynamic system subject to various kinds of failure.

A.2 Sensor networks

The push toward sensor networks has been driven by advances in hardware [12]. Silicon devices can be made smaller and cheaper than ever before. As a result, one can now envision systems that rely on hundreds, thousands or even more of these devices. These systems require a totally new approach to large-scale computing. On the one hand, they are much more tightly integrated with the environment than previous systems. Instead of relying on a small number of interfaces, every tiny piece of the system is embedded in and in contact with the environment. On the other hand, they rely fundamentally on computation being done by a large number of distributed devices that individually have limited capability. The level of devices varies from low-power, low-functionality devices to those that use a small operating system such as TinyOS [49] to achieve a reasonable level of computational capability. Algorithms for these devices must be implemented very cheaply, perhaps directly in hardware.

Sensor networks have a wide variety of potential applications. One type of application is simple data collection for the purpose of offline study. For example, one proposed network will collect data for studying Sudden Infant Death Syndrome (SIDS) by placing sensors in diapers. In such applications, where the main purpose is collecting data for offline data mining, there is no need for data processing to form beliefs online. In other applications, however, the purpose is real-time response. In addition to the fire-monitoring application, one can consider a disaster recovery application in which the goal is to help hospital services cope with a large-scale

disaster. More mundanely, one might imagine a widespread, fine-grained inventory control system. In such applications, the system would be greatly enhanced by the ability to form beliefs online.

Another application of sensor networks has been in the field of robotics, including the research undertaken in the first several chapters of this dissertation. Decentralized sensor systems have been used in automated navigation and tracking in a variety of environments [29]. Such decentralized data fusion increases the scalability, survivability and modularity of a robot by eliminating critical points of failure. Current systems rely on a distributed Kalman filter algorithm for computing the local information at each node. However, such an approach requires that the network be tree-connected, which rules out many useful applications.

The term "sensor networks" is a misnomer, since there are many other kinds of devices in the systems. In adition to sensors, the devices may contain actuators that exert control on the environment. One proposed design for sensor networks, the Hourglass architecture [84], envisions three additional kinds of sensors: data nodes are equipped with a small amount of stable storage and are responsible for collecting and storing data from sensor nodes; communication nodes are responsible for communicating with the world beyond the network, which is a relatively expensive operation; and processing nodes perform some computation on the data within the network itself. The key point is that labor is divided between different kinds of devices. The sensor nodes themselves are not expected to do a lot of processing. In this chapter, we focus on the sensor nodes that directly collect information and the processing nodes that use the information to compute beliefs online. We would expect each processing node to examine a set of sensor nodes, and communicate with a small number of other processing nodes.

If construed in the widest sense, we can also consider the term sensor network to

include virtual networks of sensors distributed across the internet. Such a network could be useful for internet security. For example, one of the major current types of security problems are distributed denial-of-service (DDoS) attacks, where an attacker floods a site with so much traffic that it effectively cuts the site off the internet. One of the best approaches developed thus far for combatting DDoS is pushback [51]. When a router notices traffic passing through it above a certain threshold, it holds up the traffic, and also notifies the upstream source. This is only a local response. Using a sensor network approach, one could potentially develop a global response. Consider a system in which a small percentage of the routers in the internet try to monitor the level of traffic to target sites, and periodically send messages to each other. The number of such messages would be very small relative to the overall amount of internet traffic, so would require very little overhead. However, using such messages, each of the routers could form beliefs about the existence of a DDoS attack against a site. This would have multiple advantages. First of all, as so as the DDoS attack is detected, all the participating routers could engage in pushback, greatly increasing the effect of the response. Equally important, the DDoS attack could potentially be detected much sooner. Instead of having to wait for a single node to see traffic above a high threshold, the attack could be detected as soon as a large number of nodes see traffic that is only somewhat above threshold. The ideas of this chapter, about using loopy belief propagation as a basis for forming and communicating beliefs, hold equally well for such a virtual sensor network as for a physical network.

A.3 Modeling a sensor network as a graphical model

There is a great deal of uncertainty in any sensor network system. For one thing, the underlying domain exhibits uncertainty; without it, we would not need to deploy sensor networks throughout the system. The sensors typically give us only partial information about the state of the system; otherwise, we would not need to compute beliefs, we would know the answers simply by looking at the sensors. Furthermore, the sensors are noisy, they might be biased, and they might be broken. In short, reasoning under uncertainty to form coherent beliefs is a major task in sensor networks.

Each sensor individually provides a reading for a particular state variable at a particular point. That reading may depend not only on the system state but also on the sensor properties. Any interaction between sensors is assumed to be the result of high-level variables. It is the job of the processing nodes to form beliefs about these high-level variables from the sensor readings, taking into account the possibility of sensor error. In a complex, widely distributed system, there will be many interacting high-level variables, and their interactions might form a complex, loopy network. Each processing node will be responsible for a set of sensors and a small number of high-level variables. We can draw a network of processing nodes, in which two nodes are neighbors if their high-level variables interact. We now make a key assumption: that the joint probability distribution over the states of all processing nodes, can be decomposed into the product of *pairwise interactions* between adjacent processing nodes. This might be an approximation, but we believe that for physical systems that operate through local interactions it will tend to be a good one.

For example, suppose we have a temperature sensor S at some location. We model READING(S) as depending on TEMP(S). In addition, S will have a certain bias BIAS(S) which is added to the temperature to produce NOISY(S). However, there is

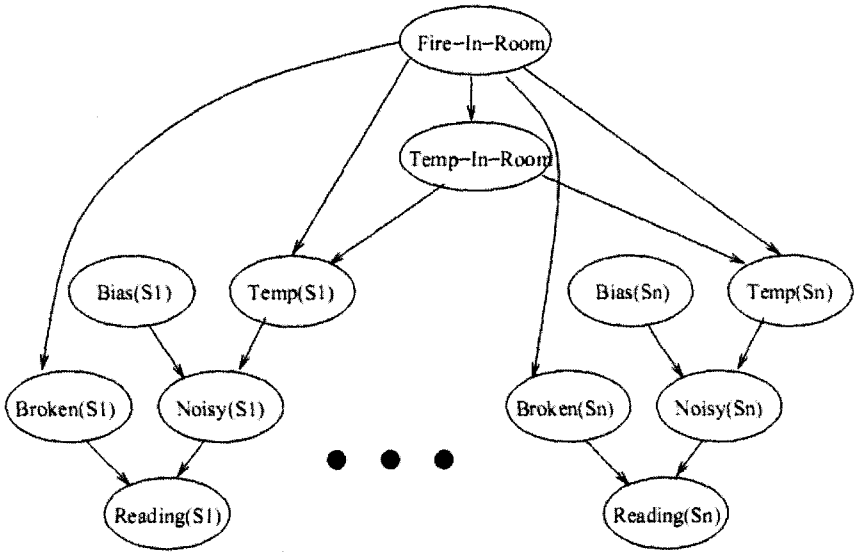

Figure A.1: Local BN for processing node

also a BROKEN(S) variable; if S is completely broken, the reading will be random. Each processing node will be responsible for a set of sensors. The processing node will have high-level variables TEMP-IN-ROOM representing the ambient temperature and FIRE-IN-ROOM, a Boolean representing whether or not there is a fire in the room. FIRE-IN-ROOM naturally influences TEMP-IN-ROOM, and TEMP-IN-ROOM influences TEMP(S) at each of the sensor locations. In addition, FIRE-IN-ROOM influences TEMP(S) because the temperature at a point is likely to deviate more from the ambient temperature if there is a fire. FIRE-IN-ROOM also makes it more likely that a sensor is broken. A schematic of the Bayesian network for a single processing node is shown in Figure A.1.

In addition, the temperature in adjacent rooms is highly correlated, as is the existence of fire. Therefore the TEMP-IN-ROOM and FIRE-IN-ROOM variables associated with one processing node are connected to those of neighboring processing nodes. Since the relationship between adjacent processing nodes is symmetric, a

Markov network is more appropriate than a Bayesian network for capturing the connectivity at this level. The relationship between two adjacent processing nodes is modeled with a compatibility function, which will be higher the closer the values of TEMP-IN-ROOM and FIRE-IN-ROOM between the two nodes. The graph of processing nodes, corresponding the the adjacency graph of locations, will be quite loopy.

The inference task in a processing node is to compute the distribution over high-level variables given sensor readings. No information need be passed back to the sensors. The sensors do not need to be told whether they are broken; that possibility is taken care of at the processing node. Since the network is used for a specific query, a technique such as Query DAGs [22] can be used to produce a computational framework in which the local beliefs can be computed very quickly. Query DAGs were designed for implementation in software or hardware for on-line, real-world applications, and so are ideal for computing local beliefs within a single processing node. However, they cannot be used for the overall process of computing beliefs in sensor networks, since they rely on exact inference algorithms. Since the inter-processing-node network can be quite loopy, exact algorithms are infeasible.

A.4 Loopy belief propagation

We therefore need to use an approximate inference algorithm. Furthermore, we need one that can easily be implemented in a distributed form, and that can be implemented efficiently in software or hardware. Loopy belief propagation (LBP) fits both of these criteria.

LBP is an extension of the belief propagation framework developed by Pearl for the polytree algorithm [74]. Pearl in fact emphasized the distributed potential of the algorithm as one of its attractive properties. While the algorithm produces exact

beliefs in singly-connected networks, it does not do so in networks with loops. Pearl discussed the idea of running belief propagation in loopy networks, but expressed concern that the beliefs would not converge.

In the coding community, the hugely successful Turbo coding scheme was developed, and it was shown that its decoding algorithm is equivalent to running belief propagation in a loopy network [64]. As a result of this success, there has been a resurgence of interest in the use of LBP as a general approximate inference algorithm for Bayesian networks. Empirical studies [25, 70] have shown that LBP is a highly competitive approximate inference algorithm. It works very quickly, and generally produces accurate approximations to the correct beliefs. While the algorithm does not always converge, cases of non-convergence are relatively rare and can easily be detected. Meanwhile, recent work [33] has laid the theoretical foundation for understanding LBP as well as pointing to generalizations. Due to its ease of implementation and strong empirical performance, LBP is emerging as a leading algorithm for approximate inference.

We follow [33] in our presentation of LBP. The nodes in the algorithm are the processing nodes. The value x_i of node i is the state of the high-level variables of processing node i. Let y_i denote the sensor readings at i, and $\mathbf{y}$ the complete set of sensor readings. The complete joint distribution over the state of the system, given the sensor readings, can be expressed as

$$P(x_1, \ldots, x_N|\mathbf{y}) = \frac{1}{Z} \prod_{ij} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i|y_i)$$

where Z is a normalization constant, the first product is taken over adjacent nodes, ψ_{ij} is the compatibility function between nodes i and j, while ϕ_i represents the effect of the local sensors on the belief in node i, as computed by the BN in node i. In

LBP, each node i sends a message m_{ij} to each of its neighbors j, and updates its beliefs b_i based on the messages it receives from its neighbors. The update rules are:

$$m_{ij}(x_j) \leftarrow \alpha \sum_{x_i} \psi_{ij}(x_i, x_j)\phi_i(x_i|y_i) \prod_{k \in N(i)\backslash j} m_{ki}(x_i)$$

$$b_i(x_i) \leftarrow \alpha \phi_i(x_i|y_i) \prod_{k \in N(i)} m_{ki}(x_i)$$

where α is a normalization constant, $N(i)$ denotes the neighbors of i, and $N(i)\backslash j$ denotes the neighbors of i except for j. The belief at a node takes into account the local evidence at the node, and the messages sent to it by all its neighbors. The message a node i sends to its neighbor j tells j which values it thinks are likely for x_j, based on what i thinks is likely for x_i as a result of its local evidence and messages from other neighbors, and the compatibility between x_j and x_i.

A.5 Asynchronous behavior

Since LBP is defined in terms of local update rules, it can easily be implemented in a distributed fashion. Furthermore, the algorithm requires only a simple set of multiplications and additions which can easily be implemented on a tiny device. In addition, the algorithm as formulated does not rely on any coordination of the messages. Each node can update its own beliefs and the messages it sends to its neighbors at any time, using the most recently sent messages from its neighbors. In practice, however, implementations of LBP on sequential computers have been synchronous. The simplest way to run LBP on a sequential machine is for nodes to take turns updating and sending messages.

While there was no reason in principle to believe that LBP would not work equally well in an asynchronous environment, the possibility existed that the sur-

prising convergence of LBP relied on an organized propagation schedule. If that was the case, any attempt to apply LBP to sensor networks would be doomed to failure. We therefore began our experimental investigations by determining whether the convergence properties found in previous experiments held up for an asynchronous implementation.

We performed experiments using two real-world Bayes networks, ALARM and HAILFINDER [30], and a synthetic sensor network called FIRESENSOR based on the fire monitoring model described in Section A.3. The two real-world nets are fairly small – 37 and 56 nodes, respectively, while FIRESENSOR consists of 680 nodes modelling one hundred identical sensor clusters connected in a 10x10 lattice. In each experiment, between 0 and 20% of the nodes were randomly assigned an observed value. We found that in all three networks, LBP continues to perform well under asynchronous conditions. In particular, asynchronous LBP converged whenever the synchronous version did, and to the same beliefs.

Next we investigated whether LBP was robust to wide variations in the rate at which nodes communicated. A typical sensor network will consist of devices of very different levels of capability, that compute and communicate at very different rates. Furthermore, devices in a deployed system will tend to adjust their computational performance to circumstances. For example, as a device loses power it will tend to communicate less frequently. In addition, it may be more difficult for one device to communicate to another for environmental reasons. For instance, if there is a lot of interference the signal from a device may only be picked up intermittently. We can model this situation as one in which the device communicates less frequently. For a variety of reasons, then, we can expect communication in a sensor network to happen at very different rates. LBP relies on all the nodes updating their beliefs and communicating messages to their neighbors. Is it able to cope with different rates of

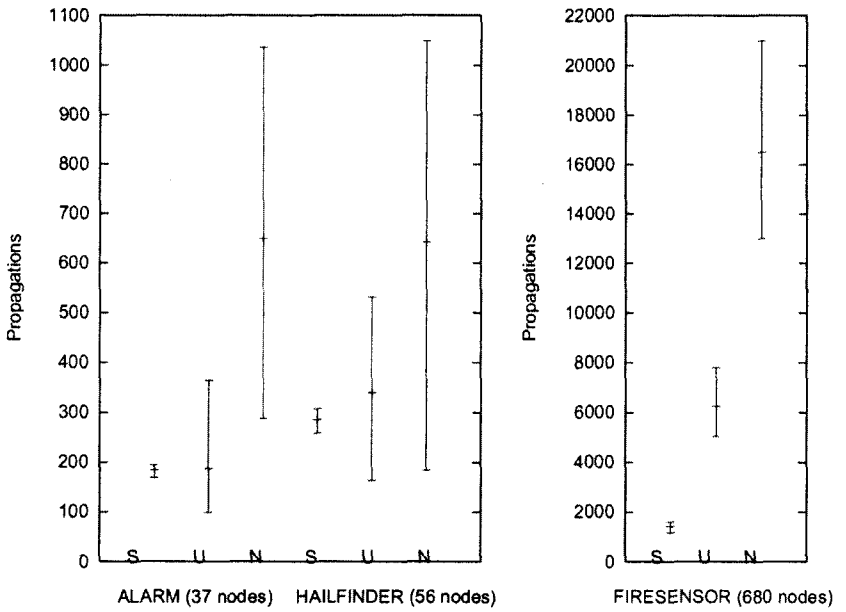

Figure A.2: Convergence performance in synchronous, uniform asynchronous, and non-uniform asynchronous networks

communication?

To answer this question, we ran experiments in which half the nodes in the network are much more likely to propagate than the other half. Each node used an exponential random process to determine when to pass messages to its neighbors. In a typical experiment, half the nodes were 10 times more likely to propagate than the others. Other ratios were also tested with similar results. We found that in all cases, asynchronous belief propagation with different propagation rates converged to the correct beliefs whenever ordinary LBP did.

In addition, the asynchronous cases perform much better than expected in terms of the number of propagations required for beliefs to converge. For instance, it might take cn propagations for the network to converge in the nondistributed algorithm, whter the constant c depends on the topological characteristics of the network – for instance, how many times messages must pass around a loop before its constituent nodes' beliefs converge. We might expect, based on this, that all nodes need to

propagate about c times to achieve convergence. If this is the case, basic probability tells us that for an asynchronous network with uniform propagation times, it would take an expected $cn \ln n$ propagations to converge. Figure A.2 shows a comparison of the number of propagation times required until convergence for synchronous LBP, asynchronous LBP with uniform propagation times, and asynchronous LBP with different propagation times. For each test network and each algorithm, the graph shows the minimum, maximum and median number of propagations required until convergence. In general, we see that the uniform asynchronous LBP uses only $\frac{2}{3}$ as many propagations as we might expect. This indicates that some nodes can underreport and the correct beliefs are still reached.

As for LBP with different propagation times, while the total number of propagations required is significantly more than for synchronous LBP, it is far less than one would expect if the slow-propagating nodes had to propagate c times in order for beliefs to converge. In a network in which half the nodes propagate 10 times more often than others, we would expect the total number of propagations required for the slow-propagating nodes to propagate c times to be about 5 times as high as for a uniform network. In fact, we find performance to be much better. For example, in the FIRESENSOR network the median number of propagations in the non-uniform case is 16,000, compared to 6,000 for the uniform case. It must be that by continuing to propagate, the fast-propagating nodes are "working overtime" and making up for the lack of propagation of the slow-propagating nodes. This is a nice property: as some of the nodes in the network slow down, they only partially slow down the network as a whole. We also discovered that the speed of convergence varies widely based on which nodes propagate often. Nodes that are centrally located and have large impact on the network also have a profound effect on inference speed. For example, if the 10 most highly-connected nodes (meaning the ones with the most

parents and children) of the ALARM network are set to propagate three times as often as the rest of the nodes in the network, this distributed, asynchronous process converges in just about the same number of steps as the sequential IBP algorithm. These results suggest that, when building sensor networks, identifying such nodes and applying resources to increase their speed would have a disproportionate positive effect on overall system performance.

A.6 Robustness to failure

The fact that networks continue to converge even when certain nodes participate markedly less often than others leads to the natural question of how such networks perform when some nodes fail to participate at all. In a distributed system of simple sensors, some will fail, and one would hope that such failures are not fatal. One can distinguish between different kinds of failures, corresponding to different kinds of nodes. Failure of sensor nodes are handled naturally in the probabilistic model by the BROKEN variables. This is a familiar kind of failure in probabilistic reasoning. Less familiar, however, is a failure of propagation nodes, which results in nodes ceasing to participate in the belief propagation process. Not only do they not form beliefs about their own state variables, they fail to send messages to other nodes. This could potentially ruin the LBP process. In fact, however, our experiments show that LBP continues to function in the face of "dead" nodes and degrades gracefully as their numbers increase.

Network topology has a profound effect on the performance of loopy propagation under degraded conditions. Redundancy is crucial. Without it, a single point of failure will cause the whole system to break down. In the limiting case, a node that bisects a network plays a crucial role in establishing accurate system-wide beliefs.

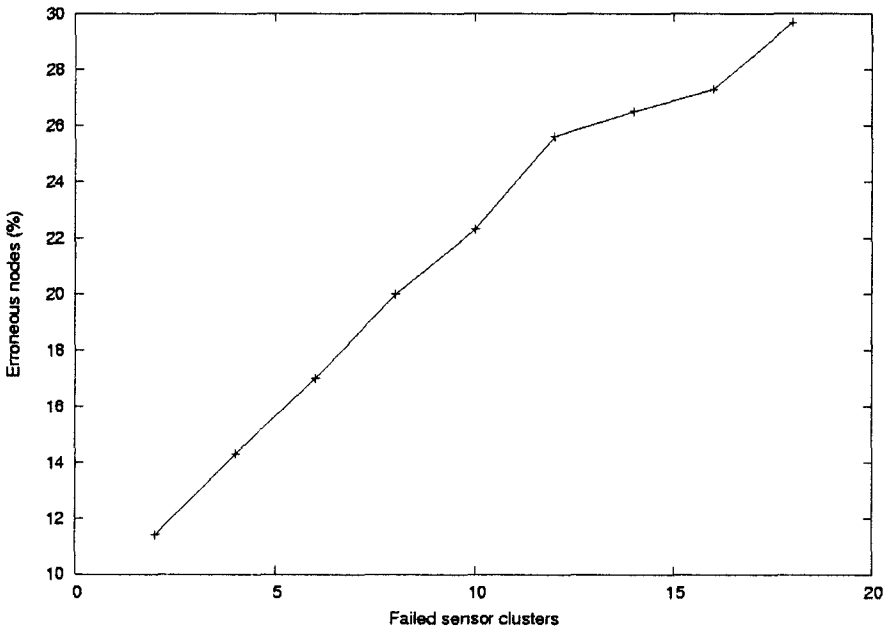

Figure A.3: Network degradation resulting from random sensor failures

If such a node ceases to function, then evidence on one side of the network cannot affect beliefs on the other side, and neither subnetwork can arrive at accurate beliefs. However, as long as at least one alternate path for information flow exists, inference is remarkably resilient.

We performed two sets of experiments with our synthetic sensor network to study the effects of node degradation. In the first, we randomly selected sensors, from 2 to 20 out of the 100 in the total network. We rendered them inoperable, then compared the beliefs of working nodes to their counterparts in a fully functional converged network. We randomly assigned observations to 10% of the nodes, choosing from among the working ones. We identified the number of nodes in the degraded network that differed from the values produced by the fully operational one, and determined the magnitude of the belief difference.

Figure A.3 shows the performance degradation as the number of failed nodes increases. With two sensors nonfunctional, only nodes directly connected to the

problem sensors show any errors at all, about 12% of the total network. As more nodes go offline, the number of affected nodes increases, but even with a fifth of the network dead, nearly two thirds of the network remains untouched by the problems.

Most of the nodes in a degraded sensor network remain completely reliable. The affected nodes, on the other hand, can be somewhat off the mark, but their beliefs still tend in the right direction. Somewhat unexpectedly (but presumably coincidentally), the average absolute belief error among the affected nodes remains almost perfectly constant as the number of dead sensors increases – right around 13%. The largest errors are found in nodes close to dead sensors and far from any observed evidence; the smallest are in those nodes strongly influenced by observations.

It is not surprising that nodes neighboring broken ones should have somewhat degraded performance, since they lose crucial information in forming their beliefs. It is also not surprising that nodes far away from observations should be more seriously affected than those close to them, since those nodes depend more heavily on their broken neighbor. What is perhaps more surprising is that the degradation does not spread in a significant way to neighboring nodes beyond the affected nodes. The beliefs at the erroneous nodes are enough in the ballpark that their neighbors are able to obtain most of the information they need. Thus loopy belief propagation hias a highly appealing graceful degradation property. Not only is the degradation local as nodes in the network fail, but it dissipates very quickly as one moves away from the failed nodes.

We ran a similar experiment on a variation of FIRESENSOR in which the processing nodes were connected in the pattern shown in Figure A.4. The network was designed to look like a plausible floor-plan for a building, which might not have as much redundancy as the complete lattice. Results were similar in nature to FIRESENSOR, and only slightly worse, due to the increased probability that knocking

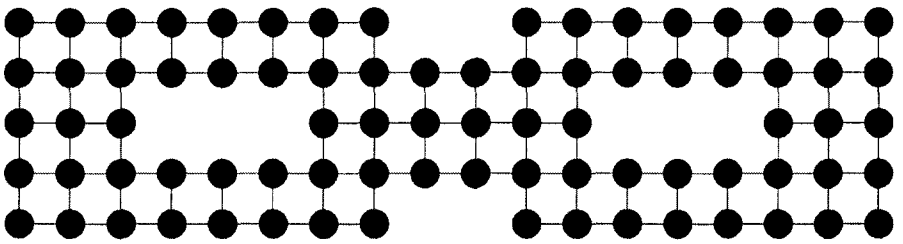

Figure A.4: FIRESENSOR building layout variant

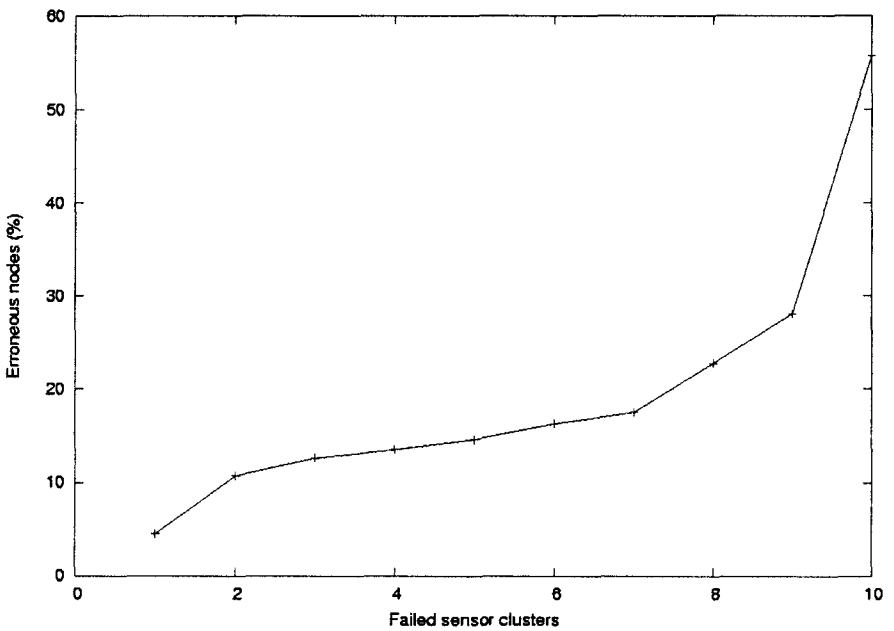

Figure A.5: Network degradation resulting from pathblocking failures

out a few nodes will block all paths from one side of the network to the other.

We designed our second set of experiments to explore the importance of redundant propagation paths in maintaining accurate beliefs, and the results are summarized in Figure A.5. Instead of choosing nodes randomly, we removed one sensor at a time from the fifth row of the 10x10 network, so that the number of communication paths between the bottom and the top of the network steadily decreased. Just as in the last experiment, we measured the total number of nodes with incorrect beliefs at convergence. Until we killed the eighth node, leaving only two paths, the system's performance did not differ significantly from the random case. Even then, the number

of affected nodes was only marginally worse, by about 10%. Of course, once all ten nodes in a row fail, error becomes extreme – only nodes whose prior probabilities are entirely determined by local observations reach correct beliefs. Thus individual messages between nodes in loopy propagation seem to encode an enormous amount of information about the state of large swaths of network – as long as a path exists, beliefs will flow.

Sensor networks are not static entities; their whole function is to change and adapt to fluctuations in the environment they are measuring. Although asynchronous networks propagate beliefs quickly and efficiently, they cannot do so instantaneously. Happily, even when nodes make and change observations in the midst of loopy propagation's flurry of messages, a system's overall beliefs continue to converge to accurate values.

In our experiments, we varied the rate of environmental change as a function of propagation time. We define a *time step* to be the mean propagation interval for each node, using an exponential random propagation model with a uniform mean across all of the nodes. At each time step, every node has a small chance of making a fresh observation. We simulated runs of the system, and measured performance as follows. Every $\frac{1}{10}$ of a time step, we determined the number of nodes whose beliefs differ from what they would be if the network had enough time to converge fully with a given set of observations. We compare the beliefs to those that would be obtained with an "instantaneous" LBP, rather than the true correct beliefs. This provides a measure of the error in the system due to slow convergence, as opposed to error due to the LBP approximation. To obtain an overall measure of performance we averaged this error over different time points.

Suppose that at each time step, each node makes a fresh observation with probability of p. Then there will be np environmental changes in each time step. For

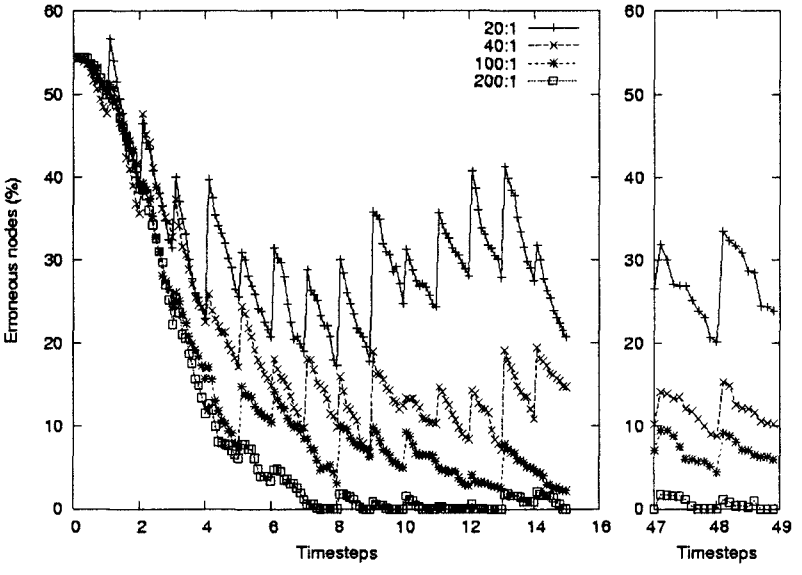

Figure A.6: Convergence performance at propagation speeds relative to environmental change

example, we found that for FIRESENSOR the network converges fully at nearly every time step if $p = 0.005$. Since the network has 680 nodes, over 3 environmental changes happen at every time step. Since this network ordinarily converges in 9 time steps with no environmental changes, about 30 such changes happen in the convergence time of the network. Nevertheless, the network has very low error in steady state. Even when environmental changes occur 10 times more rapidly, only about 20% of the network holds incorrect beliefs at any particular time step.

Figure A.6 shows the percentage of incorrect nodes in the sensor network over time at various rates of environmental change. We see that after a short burn-in period, in which the beliefs converge from their initial random settings, the percentage of incorrect nodes remains fairly stable in steady state. We conclude that LBP performs well even as the environment changes rapidly. Furthermore, it remains stable as the speed of environmental change increases, with graceful degradation of performance.

A.7 Conclusion

We view the contribution of this chapter as threefold. The first is to identify sensor networks as a fitting application area for probabilistic reasoning technology. Sensor networks are an exciting and growing field, and questions naturally arise there that have been studied by UAI researchers for years.

Secondly, we identified LBP as a particularly appropriate technology for sensor networks. It is fast and has been shown to produce generally accurate results. It is naturally distributed, and can easily be implemented in an asynchronous environment. It is also very simple, and the computations required at every node can be implemented in low-level software or hardware.

Thirdly, we have run a series of experiments to test whether LBP can withstand the variety of stresses that would be placed on it in a sensor network environment. LBP came through the experiments with flying colors, in fact surpassing our expectations. It continued to perform with highly non-uniform propagation, and in addition, required far fewer propagations than expected. Not only did it continue to work in the presence of node failures, but problems remained confined locally to the failure region. Stable in the face of environmental changes, it even continues to perform when many such changes occur during the time that it takes to converge. As a result of our experiments, we believe that LBP is up to the task of providing a foundation for reasoning and communication in sensor networks.

One issue not addressed in this chapter is modeling the system dynamics. Our graphical model is a snapshot of the state of the system at a particular point in time, and does not model changes in the state. While we have shown that beliefs in the static model are able to adapt to changes in the environment, we might do better by explicitly modeling and reasoning about such changes, using a representation such

as a dynamic Bayesian network. Extending our framework to such models is a topic for future work.

Appendix B

A Junction Tree Propagation Algorithm for Bayesian Networks with Second-Order Uncertainties [7]

B.1 Introduction

A Bayesian Network (BN) is a graphical representation of a joint probability distribution over a set of variables (also called *nodes*) [74]. A common criticism of the BN approach is that it requires a single probability representation for the conditional dependencies specifed by the network [59]. For instance, if the sky is cloudy, then the probability that it will rain may be (for example) 0.8. However, in knowledge representation, this requirement is inadequate to represent the complications in the real world. When an expert comes up with a single number 0.8, he might indicate that it is a simplification of a range [0.7, 0.9], or that it is a simplification of the fact

that it is a probability distribution with 0.8 as mean and 0.1 as standard deviation. In addition, oftentimes different experts have varying assessments of the probabilities for the same application scenario. One expert can conclude that the aforementioned probability is 0.8, while another can feel that it is too large and would suggest 0.7. Single probability representation is insufficient for reconciling discrepancies among a team of experts.

To extend the expressiveness of standard BNs, a concept of *higher-order uncertainty* has been introduced. Semantically, a higher-order uncertainty specifies additional levels of uncertainty beyond the probability distribution representation. For distinction, the uncertainties that can be represented by the conventional way is called *first-order uncertainty* (FOU). For example, given the "cloudy" status, the probabilities for "rain" and "not-rain" are single values and sum up to 1.0. One higher-order representation is to use *interval representation*. This approach has been studied by the imprecise-probability and credal-network communities [14, 97]. Under this representation, given the "cloudy" status, both probabilities for "rain" and "not-rain" are intervals. Following the interval concept, several interval propagation algorithms have been developed to propagate high-order uncertainties [8,14,24,32,92]. One limitation of interval propagation is that even in simple situations it is not highly informative. The next section will illustrate this with an example. In addition, even when an algorithm starts with very narrow intervals, those generated by the algorithm oftentimes become too broad to be useful.

In this chapter, we propose a further extension in representing conditional dependencies. Instead of an interval, we represent them with a probability distribution. In the cloudy-rain example, given the "cloudy" status, the probabilities for "rain" and "not-rain" are both represented as a distribution. For instance, given that the sky is cloudy, the probability that it rains is a distribution with a mean of 0.8 and standard

deviation 0.1, while the probability that it won't rain is a distribution with a mean of 0.2 and standard deviation 0.1. Such a probability distribution is an extension to an interval distribution because an interval can be seen as a uniform distribution bounded by the interval range. The uncertainties represented this way are called *Second-Order Uncertainties* (SOUs) or occasionally *two-layer* probabilities.

In spite of increasing expressiveness with model extensions, it becomes computationally more challenging to handle BNs with SOUs. We will use FOU-BNs to denote usual BNs, and SOU-BNs to denote BNs with SOUs. A FOU-BN represents a single joint probability distribution. In a FOU-BN, a probability inference $p(Q = q|E = e_0)$ is to calculate the posterior probability that the query variable Q is at one value (or state) q given the evidence $E = e_0$. In general, both Q and E can be nodes' sets. As shown later, a SOU-BN can be seen as a continuous set of probability distributions. Because of this, it is computationally demanding to carry out inferences over SOU-BNs. Indeed, in a SOU-BN, an inference $p(Q = q|E = e_0)$ refers to calculating the probability distribution of $Q = q$ given the evidence $E = e_0$. Therefore, an FOU inference calculates a probability value for a state of a variable, while an SOU inference calculates a probability distribution. Since conducting inferences in general is a NP-hard problem in FOU-BNs [13, 83], SOU-inferences are at least NP-hard.

Currently, no propagation algorithms for two-layer probabilities are available to conduct inferences over SOU-BNs. In this paper, we adopt the Junction Tree (JT) approach [52] and develop a SOU-JT algorithm to propagate the second- order uncertainties. The standard JT approach clusters nodes into cliques, connects cliques to form a junction tree, propagates the evidences and messages throughout the tree, and finally computes the query result. For SOU-BNs, we propose new procedures to conduct the message propagation and query answering steps. In addition, to bal-

ance the quality and the efficiency requirements in propagating two-layer probability distributions, we choose to propagate their means and covariances. Accordingly, the inference results will be represented by mean and variance of $p(Q = q|E = e_0)$. The propagation algorithm is approximate in the sense that it computes only the mean and variance of the event $Q = q$ rather than the exact probability distribution.

To validate the mean/covariance algorithm, we generalize the likelihood weighting algorithm [34, 82] to handle SOUs. The generalized algorithm exploits sampling techniques to generate "ground-truth" results for SOU inferences. We developed the sampling algorithm and used its solutions as our reference. We justified our propagation algorithm by demonstrating that it can efficiently generate high-quality results.

B.2 Related work

To increase the expressive power of BNs, researchers have proposed interval and two-layered probabilities for realistic BN applications. Probabilistic interval representation and relevant inferences in BNs have been widely studied in the literature [8, 14, 24, 32, 92]. The idea is to represent the sets of posterior probabilities as polytopes and represent a polytope using its vertices' set. At each propagation step, the algorithm calculates the vertices representing the posteriors of a node. Its major drawback is that the vertices' number grows very fast as the number of the parameters increases. Fagiuoli and Zaffalon proposed a *2U algorithm*, an exact interval propagation method for polytrees [32]. But the 2U algorithm applies only to single-connected BNs with binary variables. Tessem proposed one of the earliest approximate algorithms to propagate interval probabilities [92]. However, in this algorithm speed is achieved at the expenses of accuracy and the interval bounds tend

to diverge to [0, 1]. In addition to JT, a number of approaches exist for carrying out inferences using mathematic programming. One example is the MultiLinear Program (MLP) technique in credal networks [24]. It formulates an inference as a MLP and then solves the MLP. Its drawback is that the number of constraints grows explosively in the network size, and that a MLP is still a difficult non-linear optimization problem. As mentioned earlier, one limitation of interval propagation is its loose interval bounds. As interval propagation proceeds, the intervals generated in the algorithm quickly become too broad. In addition, a posterior distribution is poorly represented by an interval. Let us illustrate this using our cloudy-rain BN example that has a link from the node "cloudy" to "rain". It has been shown that, even if the parameters representing conditional dependencies of "cloudy" and "rain" are uniformly distributed in their ranges, the interval representing a posterior probability will not be uniformly distributed. The lack of informative power of interval-based approaches is even more evident when the parameters are not uniformly distributed, as shown in Figure B.1. The top two charts represent the distributions of the two received evidences for "cloudy". The posterior distributions for one state of "rain" node is shown in the bottom chart. We see that although the posterior distribution features a large interval (between 0.05 and 0.88), the majority of its distribution is confined in the [0.05, 0.40] interval. Incidentally, it is well approximated by a Beta distribution.

In spite of the numerous publications on interval probabilities and inferences, there is relatively little work on SOU-BNs that represent conditional dependencies as two-layered probabilities as found in this paper. This is possibly attributable to the computational challenges they pose. Realizing that it might be infeasible to come up with an exact method to do probabilistic inferences, we develop an algorithm to propagate the mean and covariances of nodes through a junction tree.

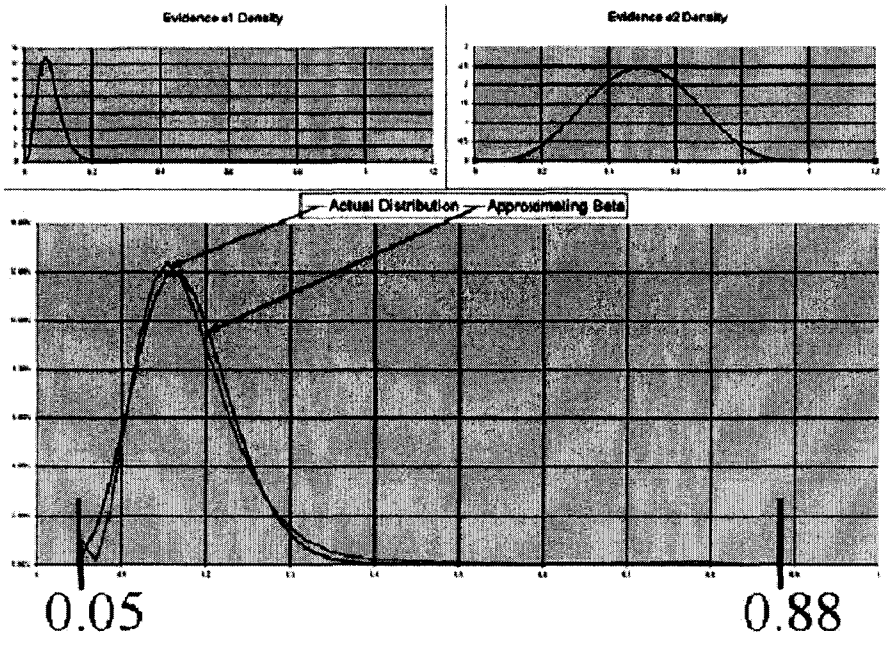

Figure B.1: An example for illustrating interval-based propagation and mean/covariance propagation

We specifically designed a set of operators for SOUs. These operators are essential for propagating means and covariances. For algorithm validation, we extended the likelihood weighting algorithm to handle SOUs and used it as a reference for our method.

B.3 BNs and probabilistic inferences

A BN is a Directed Acyclic Graph (DAG) representing a joint probability distribution over a set of variables [74]. In the graph, the nodes denote variables in an application, while the links denote the dependencies among them. Quantitatively, the Conditional Probability Table (CPT) of a node encodes the conditional distribution of the node upon the value assignments of its parent nodes. Let the variables' set be $X_1, \ldots, X_n$

and the parents of X_i be π_i. The joint distribution $p(X_1, \ldots, X_n)$ factorizes into:

$$p(X_1, \ldots, X_n) = \prod_i p(X_i|\pi_i). \tag{B.1}$$

We use the notation p_{ijk} to denote the conditional probability $p(X_i = j|\pi_i = k)$ where j is a value of variable X_i and k is a combination of the values of the parents of X_i. Therefore, a BN can be represented by $< \mathcal{N}, \{p_{ijk}\} >$ where $\mathcal{N}$ denotes the DAG qualitative part and $\{p_{ijk}\}$ denotes the quantitative part.

An *inference* or *query* $P(Q = q|E = e_0)$ is to calculate the posterior probability distribution of a query variable Q being at its specific value q, given evidence e_0 for node E. In general, Q and E can be sets of nodes.

JT is one of most popular inference algorithms [74]. JT consists of four steps: clustering nodes into cliques, connecting the cliques to form a junction tree, propagating information in the network, and answering a query. A *root* clique is the one with which inference starts. The core step is message propagation and consists of a message collection phase and a distribution phase. In message propagation, the minimal operational unit is a single message pass. When a message is passed from one clique $\mathbf{X}$ to another clique $\mathbf{Y}$, it is mediated by the sepset $\mathbf{S}$ between the two cliques. For a clique, a *potential* or a *message* is a mapping from value assignments of the nodes to the set [0,1.0]. Let potentials be $\phi_{\mathbf{X}}$, $\phi_{\mathbf{Y}}$ and $\phi_{\mathbf{R}}$. Using the HUGIN architecture [54], a message pass from $\mathbf{X}$ to $\mathbf{Y}$ occurs in two steps: projection and absorption. They are detailed in the following equations.

- **Projection.** Assign a new potential to $\mathbf{R}$, saving the old potential:

$$\phi_{\mathbf{R}}^{old} \leftarrow \phi_{\mathbf{R}}. \tag{B.2}$$

$$\phi_{\mathbf{R}} \leftarrow \sum_{\mathbf{X} \backslash \mathbf{R}} \phi_{\mathbf{X}}. \tag{B.3}$$

- **Absorption.** Assign a new potential to $\mathbf{Y}$, using both the old and the new tables of $\mathbf{R}$:

$$\phi_{\mathbf{Y}} \leftarrow \phi_{\mathbf{Y}} \frac{\phi_{\mathbf{R}}}{\phi_{\mathbf{R}}^{old}}. \tag{B.4}$$

B.4 SOU-BNs, SOU inferences and SOU potentials

We introduce SOU-BNs and probabilistic inferences and derive the mean/covariance representation for SOU-BNs. We will discuss probability potentials in the context of SOUs, a fundamental concept in the SOU-CTP algorithm.

B.4.1 SOU-BNs

An SOU-BN is a standard BN $< \mathcal{N}, p_{ijk} >$ where its CPT entries p_{ijk} are probability distributions. To denote p_{ijk} by a probability density, we introduce notation $p_{ijk\alpha}$. This is the probability of the event that $p(x_i = j|\pi_i = k) = \alpha$ for any $\alpha \in [0, 1]$, i.e., $p_{ijk\alpha} = p(p(x_i = j|\pi_i = k) = \alpha)$. With $p_{ijk\alpha}$, the probability distribution p_{ijk} is defined as

$$p_{ijk} = \{p_{ijk\alpha} | \forall \alpha \in [0, 1.0]\}. \tag{B.5}$$

An SOU-BN $< \mathcal{N}, p_{ijk} >$ can be viewed as a probability distribution over a set of FOU-BNs. For any FOU-BN in the set, its CPTs are chosen from the set $\{p_{ijk\alpha}\}$. Moreover, the parameters must meet the norm constraint, i.e., the probabilities of a

node conditioned on a parental assignment must sum up to 1.

This leads to slightly different query semantics: an FOU-BN query yields a single probability $P(Q = q|E = e_0)$, whereas inference on an SOU-BN yields a probability distribution of $p(Q = q|E = e_0)$ for a value of the query variable.

B.4.2 Mean/covariance representation of a BN

In the BN definition above, the CPT entries p_{ijk} are represented as distributions, usually as (continuous) probability densities. The representation needs a large amount of memory if the distribution is irregular. To provide a compact representation of a SOU-BN, we define the mean and covariance for CPTs. Although this representation is an approximation of an SOU-BN, it results in tremendous savings. As shown in Figure B.1, an approximate Beta distribution, determined using mean and variance estimates, can fit an actual posterior distribution.

Given the CPT entries p_{ijk} in a SOU-BN, we can define its mean and variance as follows:

$$\mu_{ijk} = \int_\alpha p_{ijk\alpha}\alpha d\alpha, \tag{B.6}$$

$$\sigma^2_{ijk} = \int_\alpha p_{ijk\alpha}(\alpha - \mu_{ijk})^2 d\alpha. \tag{B.7}$$

The covariances depict the relationship among different values given a node and its parents. With the mean and the variances, the covariance $\sigma^2_{i<j_1,j_2>k}$ is defined as follows:

$$\sigma^2_{i<j_1,j_2>k} = \int_\alpha \int_\beta p_{ij_1k\alpha}p_{ij_2k\beta}(\alpha - \mu_{ij_1k})(\beta - \mu_{ij_2k})d\alpha d\beta. \tag{B.8}$$

The goal of an inference $P(Q = q|E = e_0)$ is to compute a mean and a variance for

probability distribution $Q = q$ given the evidence e_0.

B.4.3 SOU potentials

A potential is defined over a set $\mathbf{X}$ of variables $\{X_1, \ldots, X_n\}$. We will use $X_{1:n}$ to represent $(X_1, \ldots, X_n)$ and $x_{1:n}$ to indicate an assignment to it. The potential defined over the set can be denoted by $\phi_{\mathbf{X}}$ or $\phi(X_{1:n})$. In SOU-BNs, a potential has to be interpreted differently from FOU-BNs.

For each value assignment of all variables, the notation $\phi(x_{1:n})$ can be understood as a set of real numbers. Intuitively, one can think of $\phi(x_{1:n})$ as a probability distribution that consists of probability masses $\phi(x_{1:n}; \alpha)$ over any $\alpha \in [0, 1]$. For each $\alpha \in [0, 1]$, $\phi(x_{1:n}; \alpha)$ specifies the probability that the entry $\phi(x_{1:n})$ is equal to α, i.e., $\phi(x_{1:n}; \alpha) = p(\phi(X_{1:n} = x_{1:n}) = \alpha)$.

With this notation, given a potential $\phi(X_{1:n})$, its mean value at $x_{1:n}$ is

$$\mu_\phi(x_{1:n}) = \int \phi(x_{1:n}; \alpha)\alpha d\alpha \tag{B.9}$$

and its variance at $x_{1:n}$ is

$$\sigma^2_\phi(x_{1:n}) = \int \phi(x_{1:n}; \alpha)(\alpha - \mu_\phi(x_{1:n}))^2 d\alpha. \tag{B.10}$$

The covariance for two different instantiations $x_{1:n}$ and $x'_{1:n}$ of the variables' set $\{X_{1:n}\}$ is the product

$$\sigma^2_\phi(x_{1:n}; x'_{1:n}) = \int_\alpha \int_\beta \begin{matrix} \phi(x_{1:n}; \alpha)\phi(x'_{1:n}; \beta) \\ (\alpha - \mu_\phi(x_{1:n}))(\beta - \mu_\phi(x'_{1:n}))d\alpha d\beta. \end{matrix} \tag{B.11}$$

where α and β can be any number in [0,1].

Note that if $x_{1:n} = x'_{1:n}$, the covariance $\sigma^2_\phi(x_{1:n}; x'_{1:n})$ is actually the variance

$\sigma^2_\phi(x_{1:n})$. Note also that the CPTs for a variable are examples of potentials. In fact, the CPT potential of a variable is defined over the set of the variable and its parents.

B.5 Junction tree propagation

The proposed SOU Junction Tree (SOU-JT) algorithm parallels the standard JT for FOU-BNs [50]. Just like FOU-JT, SOU-JT proceeds in four steps: construct a junction tree, initialize the tree, conduct message passing via global propagation, and generate the query results. The junction tree construction step is identical to the FOU case. In the rest of this section, we discuss the remaining three steps. For each step, we first discuss the operations and then the operators that implement them.

B.5.1 Junction tree initialization

The tree initialization sets up the initial values for the clique tree. In particular, the initial potentials and evidences are assigned to the cliques. Then, within each clique, these potentials are multiplied together to form one single potential at the end of the step. The initialization step proceeds as follows.

1. For each cluster $\mathbf{X}$ and sepset $\mathbf{S}$, set its mean matrix to be 1.0 and its covariance matrix to be 0.0. That is, $\mu_{\phi_\mathbf{X}} \leftarrow 1.0$, and $\sigma^2_{\phi_\mathbf{X}} \leftarrow 0.0$.

2. For each variable X, assign it to a cluster $\mathbf{X}$ that contains the node and all its parents. Such a cluster is called *the parental cluster of* X.

3. For each evidence variable E, identify a cluster that contains E and attach a potential λ_E to the clique.

4. For each clique, perform the multiplication operation over the current potential $\phi_\mathbf{X}$, the possible CPT potentials $p(X|\pi(X))$ for all assigned Xs and the

evidence potential λ_E in the clique:

$$\phi_{\mathbf{X}} \leftarrow \phi_{\mathbf{X}} p(X|\pi(X))\lambda_E. \tag{B.12}$$

where Π_X is over all the nodes whose CPTs are attached to the clique and Π_E is over all available evidences. The multiplication operator is defined in the next section.

Multiplication operator

The multiplication operator acts on a number of potentials $f_1, \ldots, f_n$. Each f_i is represented by a mean matrix and a covariance matrix. The operator returns a potential represented by a mean matrix and a covariance matrix. We assume that f_is have the same domain [1].

We first consider the two-potential multiplication. Let the two potentials be g and h. Let the variables' set be $\{X_{1:n}\}$, the mean matrix and the covariance matrix be $\mu_g(\mu_h)$ and $\sigma_g^2(\sigma_h^2)$, the resulting potential be gh. Then the mean matrix μ_{gh} is

$$\mu_{gh}(x_{1:n}) = \mu_g(x_{1:n})\mu_h(x_{1:n}) \tag{B.13}$$

The covariance matrix σ_{gh}^2 is

$$\begin{aligned}\sigma_{gh}^2(x_{1:n}; x'_{1:n}) = {} & \sigma_g^2(k; k')\sigma_h^2(x_{1:n}; x'_{1:n}) + \mu_h(x_{1:n})\mu_h(x'_{1:n}) \\ & \sigma_g^2(x_{1:n}; x'_{1:n}) + \mu_g(x_{1:n})\mu_g(x'_{1:n})\sigma_h^2(x_{1:n}; x'_{1:n}).\end{aligned} \tag{B.14}$$

Equation B.13 holds because the various CPTs in a BN are independent. Indeed, Equation B.13 can be verified using a Taylor expansion. Equation B.14 can be derived by expanding the covariance of gh at $(x_{1:n}; x'_{1:n})$ considering that g and h are

[1] In the event that f_i has a different domain from another f_j, we can apply the *extension operator*, introduced in the next section, to them and then carry out multiplications in a unified domain.

independent because each CPT appears exactly in one clique.

Additionally, we have proven the following results: If the matrices σ_g^2 and σ_h^2 are symmetric, so is the covariance matrix σ_{gh}^2. The resulting covariance matrix σ_{gh}^2 is independent of the multiplication order among the potentials. Finally, the multiplication operator is associative. The first result can be used to achieve computational savings, while the other two results can be exploited in multiplying more than two potentials.

Extension operator

When the multiplication operator (and the division operator discussed later) acts on two potentials that have different sets of variables, an extension operator is needed to unify their domains. Given a potential $\phi(X_{1:n})$ (with mean μ_ϕ and covariances σ_ϕ^2) and a set of variables $\{Y_{1:n}\}$, *the extension of the potential* $\phi(X_1, \ldots, X_n)$ to include $Y_{1:m}$ is the potential $\phi(X_{1:n}, Y_{1:m})$. For any assignment $(x_{1:n}, y_{1:m})$, the mean is defined as:

$$\mu_\phi(x_{1:n}, y_{1:m}) = \mu_\phi(x_{1:n}). \tag{B.15}$$

For any $(x_{1:n}, y_{1:m}; x'_{1:n}, y'_{1:m})$, its covariance is defined as:

$$\sigma_\phi^2(x_{1:n}, y_{1:m}; x'_{1:n}, y'_{1:m}) = \sigma_\phi^2(x_{1:n}; x'_{1:n}). \tag{B.16}$$

By definition, if the extended potential is projected back to the original domain space, the result is the same as the original potential.

B.5.2 Global message propagation

In this step, as in standard JT, we choose a *root clique* to direct the global message passing. The ordering among the cliques are the same as in standard JT. We will concetrate on a single message pass here.

There are two steps in computing the mean and covariance matrices for the sepset and the clique that receives a message. In projection, Equation (B.3) is achieved by a sum-out operator. In absorption, Equation (B.4) is achieved by a combination of a multiplication and division operator. Since the multiplication operator has been defined in the preceding subsection, we will define a sum-out and a division operator.

Sum-out operator

A sum-out operator acts on a potential and a set of variables. It sums out the variables in the given set and returns a potential defined over a smaller set of variables. Let the potential be ϕ and its set of domain variables be $\{X_{1:n}\}$. Without loss of generality, let the node to sum out be X_n. The mean matrix of $\phi(X_{1:n-1}$ is defined to be:

$$\mu_\phi(x_{1:n-1}) = \sum_{x_n} \mu_\phi(x_{1:n-1}, x_n) \tag{B.17}$$

and the covariance matrix of $\phi(X_{1:n-1})$ is defined to be:

$$\sigma^2_\phi(x_{1:n-1}; x'_{1:n-1)} = \sum_{x_n, x'_n} \sigma^2_\phi(x_{1:n-1}, x_n; x'_{1:n-1}, x'_n). \tag{B.18}$$

The above operator can be readily extended to sum out multiple nodes. In that case, the variables can be summed out one by one in any order.

Division operator

The division operator acts on two potentials and returns a quotient potential. Let the two potentials be g and h, and the result be g/h. Given g and h, the mean matrix of g/h is defined to be:

$$\mu_{g/h}(x_{1:n}) = \frac{\mu_g(x_{1:n})}{\mu_h(x_{1:n})} \tag{B.19}$$

and the covariance matrix $\sigma^2_{g/h}$ at $(x_{1_n}; x'_{1:n})$ of g/h is defined to be:

$$\frac{\sigma^2_g(x_{1:n}, x'_{1:n}) - \mu_{g/h}(x_{1:n})\mu_{g/h}(x'_{1:n})\sigma^2_h(x_{1:n}, x'_{1:n})}{\sigma^2_h(x_{1:n}, x'_{1:n}) + \mu_h(x_{1:n})\mu_h(x'_{1:n})}. \tag{B.20}$$

Equations B.19 and B.20 are respectively the inverse of B.17 and B.18.

B.5.3 Query answering

After the clique tree is made consistent through global propagation, we have $\phi_{\mathbf{X}} = p(\mathbf{X}, E = e_0)$ for each cluster (or sepset) $\mathbf{X}$. The query result is obtained in two steps – marginalization and normalization.

- The marginalization step can be accomplished using the sum-out operator, as discussed earlier.

- To normalize, we would use $Y_i = \frac{X_i}{\sum_{i=1}^{n} X_i}$ to replace the node X_i. This yields a set of "normalized" variables. The potential is then defined over the normalized variables and represented by its matrices. These matrices are then exploited to calculate the query result – the mean and the variance of $p(Q = q|E = e_0)$, an approximation to the distribution $p(Q = q|E = e_0)$. In the following, we define the normalization operator.

Normalization operator

Given a potential $f(X_{1:N})$, the operator returns the matrices representing a potential over a set of normalized variables $\{Y_n = \frac{X_n}{\sum_i X_i} | n = 1 : N\}$. Let the normalized potential be $\bar{f}$. For each node Y_n, its mean and variances are computed by the following two equations where $\beta = \sum_i \mu_i$ and $\sigma_{ij}^2 = \sigma_f^2(x_i, x_j)$.

$$\begin{aligned} \mu_{\bar{f}} &= \frac{\mu_n}{\beta} + \frac{1}{2\beta^3}[(2\mu_n - 2\beta)\sigma_n^2 + \sum_{i=j\neq n} 2\mu_n\sigma_i^2 \\ &+(2\mu_n - \beta)(\sum_{i\neq n}\sigma_{in}^2 + \sum_{j\neq n}\sigma_{nj}^2) + \sum_{i\neq j\neq n} 2\mu_n\sigma_{ij}^2]. \end{aligned} \tag{B.21}$$

$$\begin{aligned} \sigma_{\bar{f}}^2 &= \frac{1}{\beta^4}[(\beta - \mu_n)^2\sigma_n^2 + \mu_n^2\sum_{i\neq n}\sigma_i^2 - \mu_n(\beta - \mu_n) \\ &(\sum_{i\neq n}\sigma_{in}^2 + \sum_{j\neq n}\sigma_{nj}^2) + \mu_n^2\sum_{i\neq j\neq n}\sigma_{ij}^2]. \end{aligned} \tag{B.22}$$

B.6 Sampling approach for SOU-BNs

In this section we generalize the *likelihood weighting* algorithm for BNs to accommodate SOUs [34, 82]. The resulted sampling algorithm, run for a suitably long time, serves as a reference algorithm for validating results obtained with SOU-CTP.

The likelihood weighting algorithm relies on random sample generation. For simplicity, imagine a BN with no evidence entered. The algorithm produces a sample drawn from the prior distribution for each node without parents, then samples each child node based on the distribution given already-instantiated parents, until each node in the network has been assigned a state. It performs this process over and over again, remembering the final outcome for each state in each trial. It then averages over all of the trials to estimate the joint probability distribution for each variable in the network.

Our SOU sampling algorithm works similarly, except that for each node, we

sample twice - first, to determine the prior probability distribution given the mean and covariance matrices, then determining an actual outcome given this particular instantiation of the prior distribution. With this, we obtain a first-order estimate of the mean values of the distribution. To generating estimates we must first generate a sample of prior distributions over all of the nodes in the network, then for each such set we generate enough samples to characterize the probability distribution. Taking the set of all samples of such prior distributions, we can compute the variance on the probability of any particular node.

The likelihood weighting algorithm used in our referee system gets its name from the treatment of evidence within the network. Any particular trial receives a weight based on the likelihood of a particular sample given the available evidence. In this way, the sampling process iteratively approaches the true posterior distribution of the network. In particular, the weight for any given sample, composed of particularly instantiated evidence variable e and hidden variable z, is the product of the likelihoods for each evidence variable given its parents:

$$w(z,e) = \Pi_{i=1}^{m} p(e_i|\pi(E_i)) \tag{B.23}$$

In the SOU case, the likelihoods are part of the unknown - thus the weights become themselves randomly distributed, based on the mean and covariance values within the network. Once again, iterating over this process leads to the mean values; computing the variances requires that we sample a particular instantiation of priors, maintain constant weights while we generate a set of samples, and then perform the whole process repeatedly to obtain covariance estimations.

B.7 Experiments

B.7.1 Experimental setup

The validation process involved several different SOU-BNs, some of which were based on real-world applications and some were randomly generated. The real-world networks involved included HAILFINDER (a 57-node network for predicting weather in northern Colorado), ALARM (a 37-node network for monitoring patients in intensive care) and WIN95PTS (a 76-node network for diagnosing printer problems). All networks can be found at various depositories for BN research scattered across the web; see for example [30]. We made changes to these FOU BN models to include SOUs, manually augmenting the models with or generating random means and covariances of CPTs. To further evaluate the algorithmic performance, we created two random networks: a very loopy 36-node network arranged in a 6x6 lattice (most nodes therefore having eight neighbors), and a large but sparsely connected graph of 200 nodes. Each node represented between two and six states, and the mean and covariance values of the conditional probability tables were chosen at random.

In each experiment, we used our sampling algorithm to generate approximate referee mean likelihood and variances, then compared those results with the performance of our SOU-JT algorithm. Each node's deviation was calculated using $\frac{|x-\hat{x}|}{x}$, where x represents the value (either mean or variance) generated by the sampling algorithm and $\hat{x}$ represents the value generated by the SOU propagation algorithm. These values were then averaged over all nodes to determine the averages in Tables B.1, B.2, and B.3. To accelerate SOU-JT and to ensure numerical stability, we applied zero-compression techniques to the calculations of covariances [53]. These tests were conducted using a HUGIN-based inference engine. More recently, we developed a Lazy Propagation engine [52] featuring several optimization techniques. This has

BNs	Maximum mean deviation	Average mean deviation	Average variance deviation
Average	2.14%	0.45%	4.22%
Loopy Lattice	3.08	0.64	7.27
Large sparse	0.61	0.19	1.45

Table B.1: Best performance

proven so far to be 30 time faster, on average, than the HUGIN-based engine.

B.7.2 Results

Table B.1 shows our best results for networks with no evidence entered. The "Average" row in the table represents an average over all tested real-world BNs available. Many choices of zero-compression threshold were tried, including ones based carefully on the distribution of values within the matrix, but the best results were obtained with a simple constant-value threshold of 0.01. That is, all probabilities with mean values less than this are discarded. It can been seen that the the results from SOU-JT and the sampling algorithms are close. Hence the SOU-JT algorithm is effective in mean and variance propagation.

Table B.2 shows the performance of the algorithm with random evidence entered. In experiments, 5 or 20 percent of nodes were chosen at random, and an observation distribution over the states in the node imposed. The averages in the tables were taken over all of the nodes which did not have evidence entered. Each table entry have two numbers: the first is for the 5 percent case and the second in parentheses is for the 20 percent case. We note that adding evidence improves the performance of the networks, though the effect is less noticeable for densely connected networks (lattice network) than for sparse ones. This is explained by the fact that the values of evidence variables are unaffected by the message passing algorithm, and therefore

BNs	Maximum mean deviation	Average mean deviation	Average variance deviation
Average	2.95 (1.83)%	0.61 (0.40)%	3.92 (3.16)%
Loopy Lattice	3.41 (3.03)	0.77 (0.79)	8.04 (7.41)
Large sparse	0.40 (0.26)	0.12 (0.09)	1.25 (1.01)

Table B.2: Evidence on 5% and 20% of Nodes

BNs	Compilation	Message passing
HAILFINDER	8.2 sec	1.1 sec
ALARM	5.9	1.0
WIN95PTS	14.6	3.1
Loopy lattice	137.0	46.6
Large sparse	1.1	0.1

Table B.3: Running time

serve as a firebreak in the propagation of errors.

We show the timing performance on each network using a 1.83 GHz Pentium system in Table B.3. We ran the sampling algorithm about four hours on each network. For SOU-JT, in the table, the compilation time is the amount of time it takes to generate the junction tree and do an initial round of message passing to bring the network to a consistent state. The message passing time is the time taken to re-establish consistency after evidence has been entered. We see that most queries can be answered in (nearly) real-time manner from SOU-JT, except those on the highly dense lattice network in which each node has eight neighbors.

This validation process provides solid experimental evidence for the correctness of our SOU-JT algorithm. Under many different network conditions, the means and variances computed by SOU-JT closely match the figures obtained from the brute-force simulation. Combining the timing result, we conclude that SOU-JT is able to efficiently provide accurate inference solutions to SOU-BNs.

In addition, the results we obtained suggest that SOU-JT does not suffer from the

same problem affecting interval propagation. In interval propagation the a-posteriori intervals tend to diverge towards the [0, 1] interval as inference proceeds along the network. We did not observe this effect so far with the computation of covariances. Further tests are being conducted to verify this behavior.

B.8 Conclusion

In this chapter, we described a new extension to increase the expressiveness of BNs. We used a second-order probability representation for encoding conditional dependencies among variables. We then discussed probabilistic inferences in this context and their mean/covariance representation. We developed the first clique tree algorithm to propagate mean and covariances for SOU-BNs. To make the algorithm operational, we defined the fundamental operators including extension, multiplication, division, sum-out (marginalization) and normalization for SOU potentials. For algorithm validation, we generalized the likelihood weighting approach to accommodate SOUs and provide ground-truth inference results. Our experimental results showed that the proposed algorithm can efficiently produce high-quality inference results. These appear not to be affected by the divergence problem typical of interval propagation.

Appendix C

Apoptosis, Neurogenesis, and Information Content in Hebbian Networks [16]

C.1 Introduction

The hippocampus, a brain structure located in the medial temporal lobe, appears to be responsible for establishing novel associations during the learning process. As the brain forms new associative memories, hippocampal neurons change their stimulus-selective response patterns [100]. This change in response patterns suggests similarities to the learning processes of artificial neural networks. Since the detailed internal behavior of neurons *in vivo* is difficult to investigate, we seek to gain insight by studying the behavior of their simulated parallels.

Adult neurogenesis occurs in man as well as other mammalian species [43, 62]. This phenomenon appears most robustly in certain brain regions, particularly the dentate gyrus (DG) of the hippocampus and in the olfactory system [31, 57]. The

functional significance of neuronal plasticity (such as apoptosis and neurogenesis in the brain) is not easily observable with biological methods [1]. Neural network simulations, therefore, can provide a salient procedure for direct analysis of learning and memory properties of plasticity in neural systems. For a review of earlier hippocampal network modeling and simulation, see [36].

In [10], computer simulations were used to study the possibility that replacing neurons could favorably impact cognition as well as a variety of other brain functions informed by hippocampal activity (e.g. short and long term memory formation, adaptations to sex and stress hormones, and various forms of mental illness). Those simulations modeled learning tasks employing a three-layer neural network model of the hippocampus. These layers modeled, in turn, the entorhinal cortex, the dentate gyrus, and CA3. The network was made to learn a representation of the Roman alphabet, the first task. Upon completion of this task, the network was made to learn a representation of the similar but not identical Greek alphabet, the second task. The postulate that neurogenesis favorably influences the second task was demonstrated.

We take information in the brain (i.e. memory traces) to be recorded in a distributed manner in the synapses of the relevant neuronal assemblies. The recording mechanism takes the form of adjustments to the strength of these many synaptic connections. This synaptic adjustment proceeds by means of a dynamic learning process that must be simulated as part of the model. In the previous study [10], the model used the back-propagation algorithm, a method of learning (of the so-called supervised type) commonly employed in neural net simulations [46]. Here we shall invoke Hebbian learning, an unsupervised form of neural net learning dynamics that more realistically models potential brain circuitry. We shall show that neurogenesis favorably informs learning (i.e. memory trace formation) in the more realistic modeling context of Hebbian learning. We still find that the rate of learning of the

Greek alphabet vis-a-vis the rate of apoptosis and neurogenesis, on average, forms a U-shaped curve. A moderate amount of churn in the lifecycles of individual neurons enhances the learning ability of the network, while too much reduces the ability of a network to retain and exploit information.

The use of an unsupervised form of learning requires development of an intrinsic representation (an endogenous encoding) of the memory traces. That is, in place of guiding the model's output to take on an extrinsically (and arbitrarily) specified encoding of the information to be recorded (as with the supervised learning protocol of back-propagation), we take as a critical aspect of the learning ability, the capacity of the model (i.e. of the hippocampus) to implement an intrinsic and autonomous method for encoding of the information being presented. That a neural system has the functionality to do this is a key and novel feature of the present approach to the study of neurogenesis, and one that we believe accurately models memory establishment in the brain. We are aware that our observations may inform issues of natural selection.

Using an autonomous learning model in this fashion, we can begin to approach questions of why this cycle of cell death and rebirth increases learning plasticity. Our simulation allows us to characterize and investigate the conditions under which a task such as memory formation succeeds and fails. We uncovered surprising behavior with regards to the ease with which a network learned the two alphabets. Briefly put, when we present a randomly-weighted network with the Roman alphabet, sometimes convergence occurs very swiftly, other times more slowly, and yet others not at all. We discovered that quick-converging networks have a much more difficult time when presented subsequently with the Greek alphabet than do those that were required to invest more time in learning the Roman.

A computational model such as ours allows us to investigate why this might

be so. We demonstrate patterns based on the neuron participation rate and the level of saturated neurons, and suggest an information-theoretic explanation for how apoptosis and neurogenesis might help the hippocampus to escape situations where its plasticity and information capacity are compromised. In other words, cytotoxicity prevents situtations where a neural network settles into a configuration where all decisions are made by relatively few neurons, to the detriment of its overall capacity, flexibility and power.

This chapter is organized as follows: Section C.2 describes the architecture of our hippocampus model, including a discussion of the Hebbian dynamics used in the learning process. Section C.3 explains the specific process and parameters used during the alphabet learning tasks, and presents our results. Section C.4 discusses the experimental results and what they demonstrate about the efficacy of the apoptosis and neurogenesis mechanisms. In addition, we demonstrate how our results support an information-theoretic argument as to why these two processes should be so important to the flexibility of the learning process. Finally, we sum up our results and contributions in Section C.5.

C.2 Experimental setup

C.2.1 The neural net architecture, I/O dynamics, learning dynamics

The neural circuit

The hippocampus comprises three layers. The first layer, the entorhinal cortex, is connected by the perforant path to the second layer, the dentate gyrus. The latter in turn is connected by the mossy fibers to CA3, the third layer. In the model,

we shall refer to these as layer k ($k = 1, 2, 3$ respectively). The kth layer has a number, N_k of neurons. The perforant path and the mossy fibers are simulated by forward excitatory (synaptic) connections between the model's layers. The model also includes lateral inhibitory connections within layers 2 and 3. (See Figure C.1). Each synaptic connection is characterized by an associated weight w_{ij}^{kl}, a real valued parameter, where the indices denote a connection from neuron j in layer l to neuron i in layer k. Of course the only meaningful superscript pairs are 21 and 32 (for forward connections) and 22 and 33 (for lateral connections), and a synaptic weight corresponding to any other indices that may appear is taken to be zero. When a neuron is connected to another neuron, it is connected to all of the neurons in the latter's layer. We shall describe such an arrangement as being fully-connected.

Setting an associated synaptic weight to zero accommodates missing connections and allowed us to experiment with different levels of connectivity. In the simplest arrangement, each node was completely connected – each neuron's output connected to every neuron in its own layer and every neuron in the succeeding layer. Connections could disappear if the learning process drove their associated weights to zero, but every synapse began the process with a nonzero value. In other tests, each weight had a certain chance of being set to zero during the network's initialization, representing a nonexistent connection, and we furthermore prevented these weights from changing during the learning process. We used values of 25%, 50% and 75% respectively for the proportion of these missing connections. Motivated by the process of neurogenesis, we are modeling the case where neurons don't necessarily spring into being fully connected with all of their counterparts.

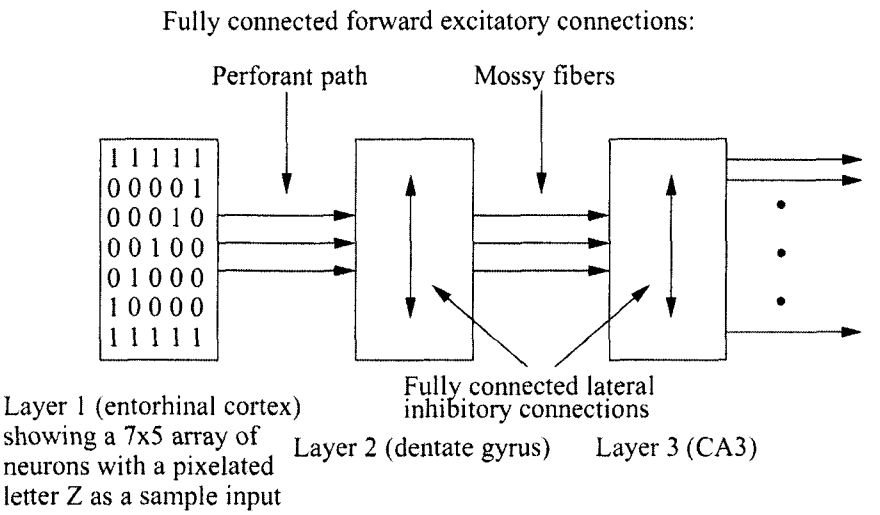

Figure C.1: Schematic of the neural network. Arrows indicate a fully connected set of synapses, inter- or intra-layer, as the case may be.

Input/output dynamics

Layer 1 is the input layer to the neural net, while the output of layer 3 is taken to represent the output of the network as a whole. The neuronal input/output dynamics are defined as follows. Let y_j^l denote the output of neuron $j(j = 1, \ldots, N_l)$ in layer $l(l = 1, 2, 3)$. Then v_i^k, the total weighted input to neuron i in layer $k(k = 2, 3)$ is specified as follows.

$$v_i^k = \sum_{j=1}^{N_l} w_{ij}^{k,k-1} y_j^{k-1} + \sum_{j=1}^{N_k} w_{ij}^{kk} y_j^k \tag{C.1}$$

The neurons are taken to be McCulloch-Pitts neurons. This implies that for $k = 2, 3$, the output y_i^k is

$$y_i^k = \begin{cases} 1, & v_i^k \geq \Theta^k \\ 0, & v_i^k < \Theta^k \end{cases} \tag{C.2}$$

where Θ^k is the neuronal firing threshold.

Input/output sequencing

Each exogenous input (an alphabetic character) is presented at layer 1. (See Figure C.1) The outputs of all neurons in layers 2 and 3 are specified using (C.1) and (C.2). These neurons are taken to fire in the following order.

$$y_1^2, y_2^2, \cdots, y_{N_2}^2, y_1^3, y_2^3, \cdots, y_{N_3}^3 \tag{C.3}$$

So layer 1 fires first and then layer 2 fires, and as indicated in (C.3), the neurons fire in sequence within each layer as well. The numbering of the neurons within a layer is chosen arbitrarily.[1]

Learning

Synaptic weights are initialized randomly, and they change according to *Hebb's law*, that is, according to correlation between input at a synapse and subsequent firing/not-firing of that synapse's neuron. This law is implemented as follows.

$$\Delta w_{ij}^{kl} = \tau(a_0^{kl} y_i^k y_j^l + a_1^{kl} y_i^k + a_2^{kl} y_j^l) \tag{C.4}$$

where τ is the learning rate. The coefficients a_j^{kl} appearing in (C.4) are chosen so that the computed weight change is consistent with the correlations that arise during the learning stage. In other words, the constants enforce a weight change in the proper direction, given whether the two neurons fired together or not. A neuron's weights are updated immediately upon that neuron's firing.

[1] The averaging of many randomly initiated runs, characteristic of our approach to the simulation, accomodates our fixing of a single ordering choice for all of the neurons withn a layer; a choice made for reasons of simplicity, causing no loss of generality.

Clock cycle

The learning algorithm proceeds with a clock timing indexed with n, say. To indicate that a variable changes with clock cycle, this time index n will be appended to that variable accordingly. A tick (an advance of the clock) is specified in the next section.

C.2.2 The learning tasks

The alphabets and their representation

The neural net is to learn the upper case characters of two different alphabets, the Roman and the Greek. These are represented in pixelated form, an example of which (the letter "Z") can be seen in Figure C.1. Illustrations of the complete set of inputs can be found in [10]. Note that these two alphabets have 14 character symbols in common.

The internal coding

As a first task, the neural net (neural circuit) is to learn the characters of the Roman alphabet, and to accomplish this task, the net is repeatedly presented with representations of those characters lexicographically. Referring to Figure C.1, we see that layer 1 is modeled as a pixelated retina. The characters are presented in pixelated form on that retina, and so this representation defines the successive inputs $y^1(n), n = 1, 2, \cdots$ as binary vectors. (Note that this assignment of the time index implies that the clock ticks once after the last neuron in layer 3 fires and the latter's weights are updated.) The net creates an *evolving* binary encoding of each character as the latter is input. This encoding is defined as the corresponding vector of network outputs $y^3(n), n = 1, 2, \cdots$, and so, a character's encoding is a vertex of the unit binary 2^{N_3}-cube. The alphabet itself is encoded by a collection of such

vertices. We take the net to have learned the alphabet (ceasing thereby the learning presentations) if the following two conditions are met.

1. The output encoding the alphabet is a collection of M (where M is the alphabet size) vertices, that is, each character corresponds to a unique vertex.[2]

2. This encoding is repeated exactly without exception during presentation of the entire alphabet an agreed-upon number (say $R > 0$) of times.

The weights associated with the neurons continue to change according to Hebb's Law as long as we continue to present the alphabets. Thus, the learning process continues even after the network has settled on a unique encoding. However, once such an encoding had established itself and repeated the number of times specified by R, we halted the test run and began another.

The learning task changes

After the net has learned the Roman alphabet, the task is switched to learning the Greek alphabet. The learning of the Greek alphabet proceeds as in the earlier manner for the Roman. Finally, for some of our experiments, the network attempts a third task – that of relearning the Roman alphabet.

C.2.3 Modeling apoptosis and neurogenesis

Finally, we modeled cytotoxicity and neurogenesis by assuming that neurons with highly-saturated weights were more likely to perish from overuse. Whenever we conducted a round of apoptosis, we assigned to each node a probability of death. Each node added the absolute values of all of its weights together, and the probability

[2]A unique encoding for each letter was the goal, but we found that convergence occurred much more quickly and often if we allowed a small number of redundant encodings. This was formalized as the "remission factor" f, described in section C.3.1.

of cell death increased linearly from zero, according to the amount by which the weight sum exceeded a threshold.

To implement this, let $w_i^{21}, i = 1, \cdots, N_2$ denote the vector of synaptic weights corresponding to forward connections into neuron i in layer 2, and let w_i^{22} be the analogous vector corresponding to lateral connections within layer 2 into that neuron. Then we compute

$$T_i = ||w_i^{21}|| + ||w_i^{22}|| \tag{C.5}$$

Here $||z||$ denotes the Euclidean norm of a vector z. T_i is a measure of the cytotoxicity of the corresponding neuron. In addition to the probabilistic cell death mentioned above (used in the experiments described in section C.3.2), we also performed experiments (described in section C.3.2) where we controlled exactly how many neurons would be replaced. Neurons were sorted according to their T_i values, and the n neurons with the highest cytotoxicity were replaced by new, randomly initialized neurons.

C.3 The simulation

C.3.1 Simulation protocols and parameter values

Layer sizes

We take $N_1 = 35$, corresponding to a 7x5 input retina. N_2 and N_3 are allowed to vary. Values of N_2 are chosen from $\{16, 20, 24, 28, 32\}$ and N_3 from $\{11, 12, 13, 14\}$.

Synaptic weights, initial values, floor and ceiling

The magnitudes of the initial values of synaptic weights are chosen randomly from a specified interval I, with the forward excitatory weights positive and the lateral inhibitory weights negative. The weights develop according to (C.4), the learning formula, but they are not allowed to change sign nor are their magnitudes permitted to exceed a ceiling C. Specifically, we take $I = [0, 0.1]$ and $C = 0.125$. If a computed weight change would cause the value of the weight changed to exit the interval $[0, C]$ on the left or right, its actual value is truncated and taken to be the floor or ceiling value 0 or C, as the case may be.

Learning rate adjustment

The displacements calculated by iterative systems, such as the dynamical systems ((C.1), (C.2), and (C.4)) in our simulation, change as the learning progresses, typically trending smaller as convergence is approached. It is useful to vary the learning rate τ, decreasing and increasing it to stimulate the weight displacements to trend smaller or larger more responsively.[3] Among the many ways to install such a feature, the following autonomous choice was taken. Specifically the learning rate τ is varied according to the following rule.

$$\tau_{n+1} = \tau_n \frac{||y^3(n) - y^3(n-1)||_H + 1}{||y^3(n-1) - y^3(n-2)||_H + 1} \tag{C.6}$$

Here $||y^3||_H$ denotes the Hamming norm of $y^3 = (y_1^3, \cdots, y_{N_3}^3)$, the binary valued vector of layer 3 outputs (that is, the net's output vector).

[3]Global convergence of iterative dynamical systems is accelerated when the displacements they compute, usually trending larger-to-smaller, are artificially exaggerated by varying the learning rate. This commonly used algorithmic technique is sometimes referred to as *over/under relaxation.*

Hebb rule parameters

For the forward connections from layer l to layer k (i.e., for $(k,l) = (2,1)$ and $(3,2)$), we take $a_0^{kl} = 1.5$, $a_1^{kl} = -0.5$, and $a_2^{kl} = -0.5$. For the lateral connections (i.e., for $(k,l) = (2,2)$ and $(3,3)$), we take $a_0^{kl} = -1.5$, $a_1^{kl} = 0.5$, and $a_2^{kl} = 0.5$. These choices are seen to accomodate the correlation requirements of Hebb's rule.

Learning epoch, repeat parameter, and remission factor

The process of displaying the characters of an entire alphabet in lexicographical order on the input retina (each character display followed by the specified neural firings and weight updates associated with that display) is called a learning epoch.[4] During the set of experiments described in Section C.3.2, the number of such epochs allowed in a training run was limited arbitrarily to 400. For the tasks described in Section C.3.2, we removed from consideration all trials that failed to converge within 200 epochs.

The value of the repeat number (for encodings) is set arbitrarily to $R = 3$. We don't expect learning always to be perfect (complete) in a reasonable number of epochs, and so we also introduce a learning remission factor denoted by f. That is, we specify a fraction f of the alphabet that, if learned, is considered adequate. In one experiment, values for f are chosen from $\{0.8, 0.85, 0.9\}$, while in the others, only $f = 0.9$ is used. Allowing the network to misclassify more letters improved the number and speed of convergences, of course, but not enough to be worth sacrificing accuracy. Besides, when investigating differences in convergence timing, reducing the classification threshold would reduce the disparities between high- and low-performing networks, blunting our perception of the difference.

[4]Since there are $M = 24/26$ characters in the Greek/Roman alphabets, a learning epoch takes 24/26 clock ticks.

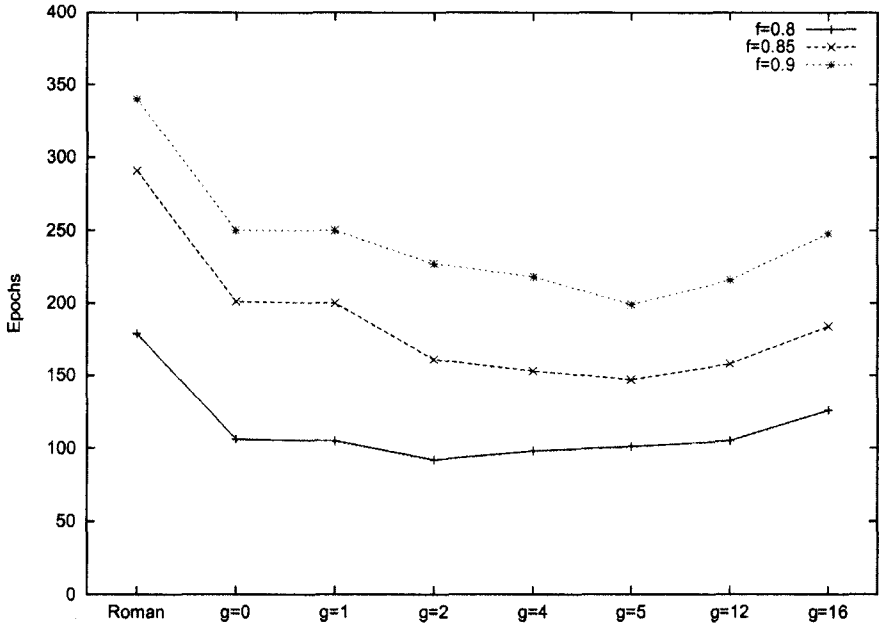

Figure C.2: Average number of epochs required to reach convergence

Threshold parameters

The specific choices of thresholds are $\Theta^2 = \Theta^3 = 0.1$.

C.3.2 Simulation results

Partial connectivity

An attempt to model partially-connected networks failed to produce any useful results. With only 25% connectivity, we never managed to produce a network that converged within the time limit. Performance improved at the 50% and 75% connectivity levels, but the only noticeable difference between the results from a fully connected network and those from the partially connected ones was that the latter required more time to reach the same results.

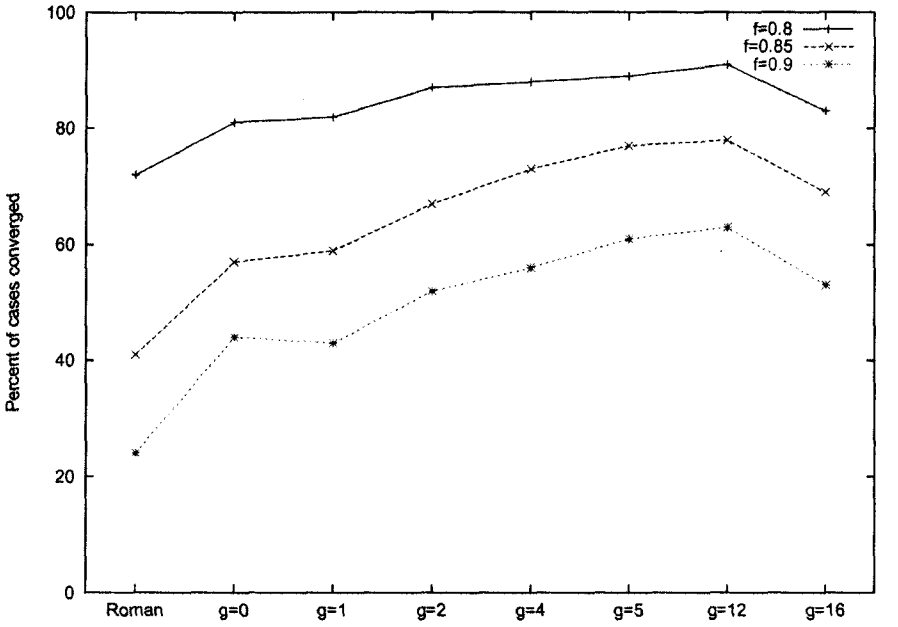

Figure C.3: Fraction of runs completing learning (i.e., converging within 400 epochs)

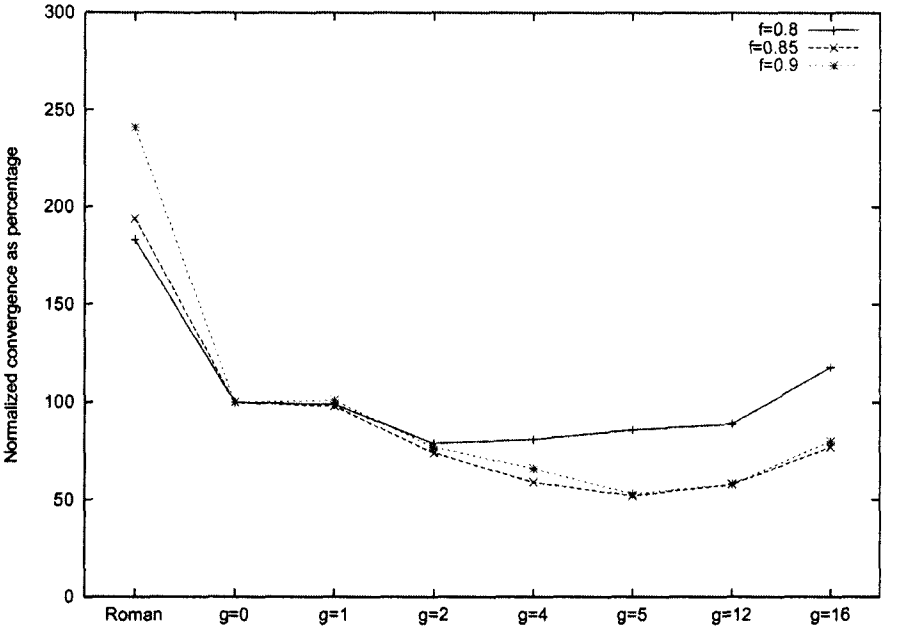

Figure C.4: Normalized convergence fraction (ratio of Figures C.2 and C.3)

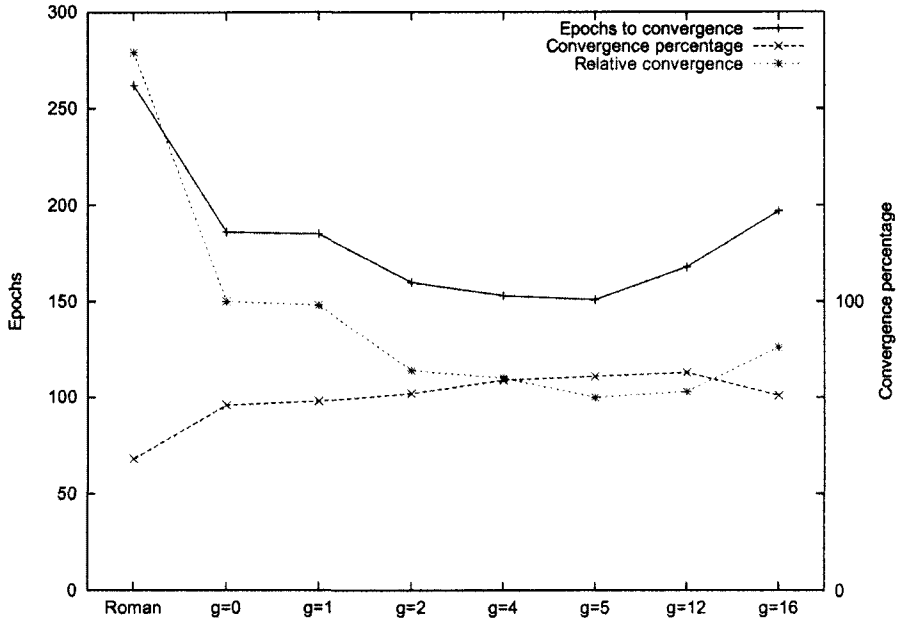

Figure C.5: Summary of Figures C.2 - C.4, averaged over all values of f

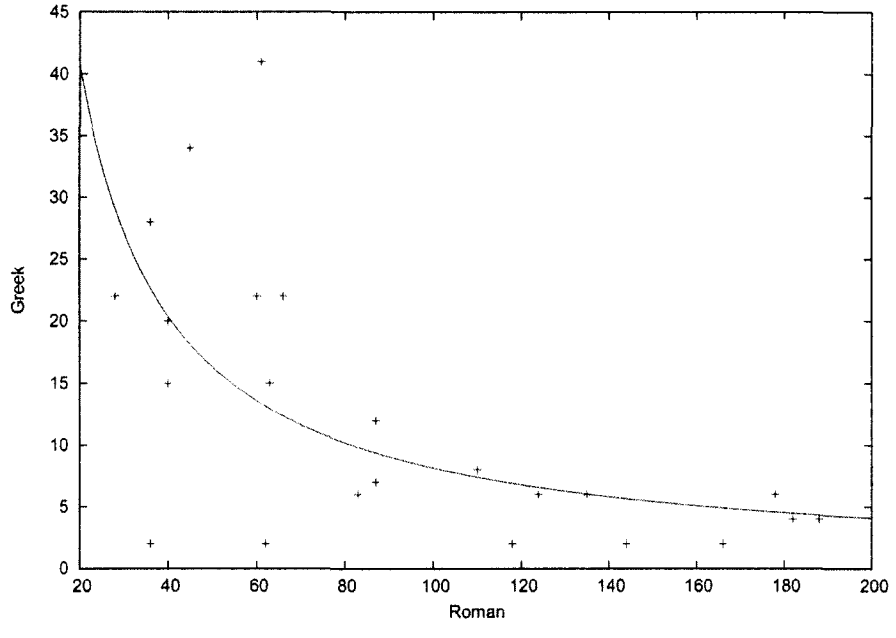

Figure C.6: Convergence performance with 16 DG nodes

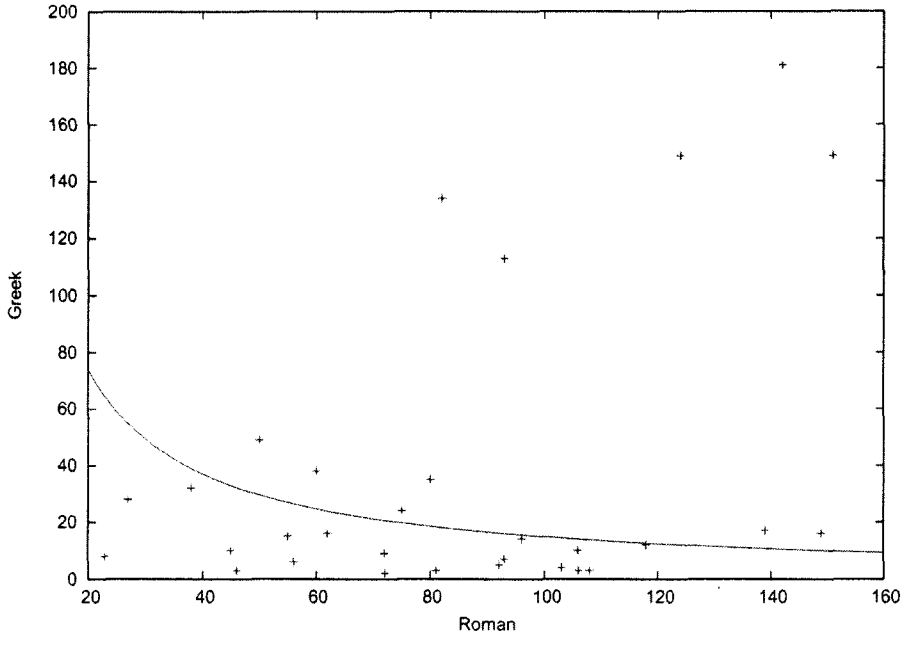

Figure C.7: Convergence performance with 24 DG nodes

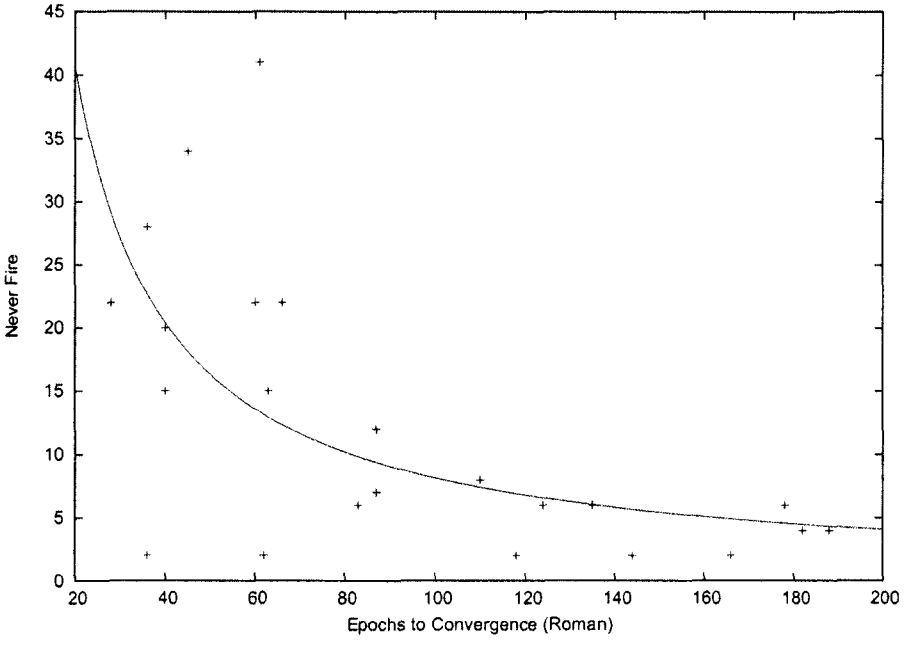

Figure C.8: Percentage of DG neuron nonparticipation with 16 DG nodes

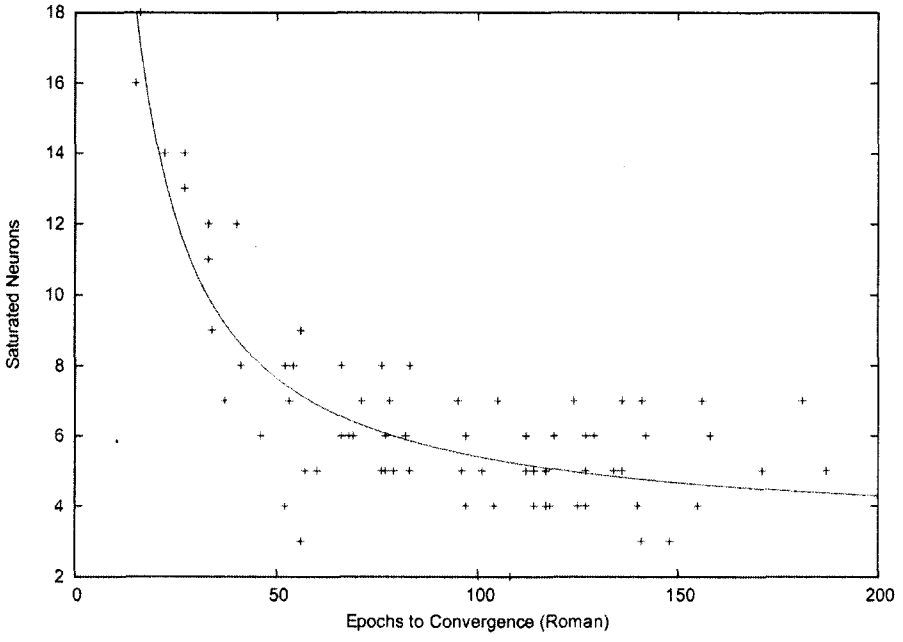

Figure C.9: Number of saturated DG neurons after Roman alphabet learning (out of 24 total DG nodes)

Learning Greek with cytotoxicity

In this experiment, a neural network is exposed to the Roman alphabet until it converges on a unique output encoding for each letter. From this configuration, the network then embarks on the task of learning the Greek alphabet, with varying levels of apoptosis and neurogenesis occurring.

A particular simulation case corresponds to a triplet (f, N_2, N_3). (60 = 3x5x4 cases in all.) Each case was run 20 times with newly chosen random starting weights each time, 1200 runs in all. Any particular one of these runs is indexed by the symbol[5] $(f, N_2, N_3)(r), r = 1, \ldots, 20$. The results are presented in 4 plots. These are the 3 plots corresponding to $(f, \bar{N}_2, \bar{N}_3)(\bar{r})$, where the upper bars represent averaging over all values of the barred symbol, and one plot corresponding to $(\bar{f}, \bar{N}_2, \bar{N}_3)(\bar{r})$. The first three plots, therefore, represent an average over 400 runs each, while the

[5]Other parameters (such as a and Θ) associated with a run are not displayed as an index, since those parameters are fixed in value.

last represents an average over 1200 runs. The plots, each containing three curves, are given in Figures C.2 - C.5. For the first three plots, the three curves correspond respectively to the three choices of f, namely $\{0.8, 0.85, 0.9\}$.

The abscissas of the plots represent the different learning tasks. The first data point on each curve represents the initial task of learning the Roman alphabet. Each succeeding point denotes the task of learning the Greek alphabet with g neurons replaced ($g = \{0, 1, 2, 4, 5, 12, 16\}$)[6] in layer 2.

Figure C.2 is a plot of the convergence time in epochs needed for the learning of an alphabet to occur. In this experiment, if learning in a run does not occur before the limit of 400 epochs is reached, then 400 is taken as a default value of the number of learning epochs required for that run. The U-shape of these curves shows that replacing an increasing number of neurons leads to an increase in the ability to learn new information, up to a point. The increase in learning plasticitiy from neurogenesis dominates the forgetting effect of apoptosis until the latter effect becomes sufficiently great.

Figure C.3 is a plot of the fraction of runs (out of 20) that complete the learning before reaching the 400-epoch limit. (Recall that completion of learning means the invariance of the intrinsic encoding achieves the repetition requirement (R=3) subject to the remission factor f. In this case, the U-shape is inverted, but demonstrates the same effects as Figure C.2. To a point, apoptosis and neurogenesis increase the ability of the network to learn new information. As too many neurons die, however (at the far right of the figure), too much information about the common characters between the Greek and Roman alphabets is lost, and the network's prior knowledge of the Roman alphabet provides less assistance in quickly converging on the Greek

[6]Replaced, meaning as customary, the apoptosis of g neurons followed by their replacement through neurogenesis, moreover with randomly-selected synaptic weights.

alphabet.

The effects shown in Figures C.2 and C.3 are related and should be examined together. This is done in Figure C.4, which is a plot of the normalized fraction of cases that converged. In particular, we first compute the ratio of the corresponding curves in Figure C.2 by those in Figure C.3. This curve is in turn normalized by its own value at the second point (where Greek is learned without neurogenesis). By doing so, the U-shaped curve, which shows the initial increase and subsequent decrease of learning effectiveness as the rate of apoptosis and neurogenesis increases, becomes even more pronounced.

Figure C.5 is a plot corresponding to $(\bar{f}, \bar{N}_2, \bar{N}_3)(\bar{r})$. Namely it is a plot of the average of the three curves in each of Figures C.2 (Epochs to convergence), C.3 (Convergence percentage) and C.4 (Relative convergence). These averages therefore illustrate the networks performance over all 1200 runs.

Patterns of learning and network activity

Having investigated the alphabet-learning behavior over a wide variety of network configurations and initial parameters, we turned to investigate the learning process and parameters in greater detail. We sought patterns between the rates of learning of the two alphabets, the activity level of the various neurons involved, and the distribution of neuronal weights within individual network neurons. In this experiment, we used only two network configurations, because we were interested in clarifying the relationship between the process of learning the two alphabets. Thus each simulation again corresponds to a triplet (f, N_2, N_3), but these take on only the values $(0.9, 16, 13)$ and $(0.9, 24, 13)$. Also, we conduct each trial with only 200, as opposed to 400, epochs.

Figure C.6 plots the number of epochs it took for a network to learn the Greek

alphabet against the number of iterations it previously spent learning the Roman alphabet. The continuous plot (in this and all subsequent figures) is a least-squares best-fit of the function $f(x) = ax^b + c$. In nearly every case, the Greek was easier for a network to learn after having been exposed to the Roman. However, the key result is that a network that happened to converge very quickly with the Roman alphabet almost always took a comparatively long time to converge on the Greek, while networks that had to work for a long time before finally learning Roman letters took to Greek very quickly.

Figure C.7 shows the same kind of data, but for a network with a larger dentate gyrus (24 as opposed to 16). Here, the time trade-off is much less pronounced, and several outliers disturb the picture. Disregarding those, the trend still exists.

Figure C.8 shows the fraction of neurons within the dentate gyrus which *never* fire after converging on the Roman alphabet. In other words, these neurons fail to participate in letter identification in any way. Not one of the 26 input patterns elicits any activity from them – they may as well not be there, as far as the letter-recognition task is concerned. Networks that converge quickly have a far greater incidence of these non-participatory neurons.

Figure C.9 shows the number of neurons that are saturated after converging on the Roman alphabet. A saturated neuron is one whose weights exceed the apoptosis threshold, and are thus candidates for potential death. Once again, the same curve type emerges. Quick convergence leads to a large number of saturated neurons.

Relearning and persistence of memory

The final set of experiments involved allowing the networks to revisit the Roman alphabet after they had converged on encodings for the Greek letters. We were interested in the persistence of memory, and whether apoptosis would adversely affect

recall of old memories, just as it enhances learning new data. This does not seem to be the case. There were no statistically significant differences in the relearning rate, whether apoptosis occurred or didn't, and whether Roman or Greek learning went quickly or slowly.

C.4 Analysis

The ability to learn the Greek alphabet after the Roman has been learned is favorably informed by apoptosis and neurogenesis. After having learned the Roman alphabet, learning the Greek alphabet usually takes 30-40% fewer learning epochs. The Roman learning places the network in a favorable posture for the subsequent learning of the Greek. By favorable posture, we mean that the synaptic weights developed by the learning of the Roman characters already reflect information about the 14 upper-case characters that the alphabets have in common.

Neurogenesis increases the ability to learn the Greek alphabet. The curves in Figure C.2 descend with each increase of the number g of new neurons made available by the neurogenesis to the Greek learning, up to a point. When too many neurons are replaced the improvement levels off and then begins to reverse. We interpret this by noting that while the new neurons are aiding the learning of the Greek by replacing older neurons which have become saturated,[7] the apoptosis of the old neurons trained on the Roman alphabet causes a progressive loss of that Roman alphabet information (the commonality of the alphabets) that was responsible for the initial learning-performance gain.

The experiments described in section C.3.2 throw more light on this tradeoff,

[7] As previously mentioned, a neuron is taken to be saturated when the Euclidean norm of the vector of input weights reaches a certain threshold. This happens when the weights have been driven by the dynamics to be near the ceiling C.

and exactly what is happening within the network structure. Rapidly converging networks start out with a few neurons that, by chance, have weights that help a great deal to differentiate among letters of the Roman alphabet. These neurons tend to dominate the network, quickly suppressing competing neurons to the point of quiescence, as shown in Figure C.8.

Furthermore, the weights of the neurons that do participate become saturated (Figure C.9). Thus, when the alphabet is learned quickly, the network consists of many neurons that are effectively ignored and others which fire often and indiscriminately.

From an information-theoretic perspective, the poor capacity for learning additional information exhibited by these inflexible, saturated, dominating neurons comes as no real surprise. A network in which all neurons participate, and where each synapse is weighted differently, can convey a large amount of information. A neuron with a well-distributed set of weights has a huge number of possible internal states, and therefore potentially a large amount of entropy (in the information-theoretic sense). The better-distributed the weights are, the closer the entropy will come to the theoretical maximum $\log(2K + 1)$, where K is the number of internal states available to the neuron.

Entropy, of course, translates directly to information content. In the case of the networks that resist learning the Greek alphabet after settling on the Roman, this content is quite low. Many neurons don't participate at all, so their potential to encode information is completely ignored. Furthermore, the neurons that do participate have a very limited internal state space – a huge fraction of all possible input encodings lead to exactly the same output, since many of their weights are saturated at the maximum. This being the case, the fact that they fire comes as no surprise. Low surprise means little information.

Thus networks that slowly converge on the Roman alphabet carefully refine their weights, preserving a wide diversity of possible firing patterns and exploiting the capacity of many more neurons than in the case of rapidly-converging networks. Such networks are simply more effective – they are able to encode and transmit more information than their fast-learning counterparts that overwork some of their constituent neurons and underemploy others. Our evidence suggests that this is why they can learn new information much more quickly.

By preferentially replacing ill-behaved neurons that simultaneously suppress participation by others and carry little information themselves, apoptosis and neurogenesis help to maintain a network information capacity that is closer to ideal. This informs the puzzling result that apoptosis does not have a negative impact on relearning old information. The loss of these dominating but information-poor neurons is more than made up for by the increased responsiveness and activity of the other neurons in the network.

C.5 Contributions

We employed an autonomous (i.e., unsupervised) form of learning to model and simulate the recording of information in the hippocampus. This required that the model be capable of developing memory traces that are intrinsic representations (endogenous encodings) of the information to be learned. Demonstrating that a neural system has the functionality to do this is a key and novel feature of the present approach.

We have uncovered striking patterns in the learning behavior of unsupervised neural networks, suggesting a relationship between the time and effort it takes to learn something and the flexibility and adaptibility of that knowledge once learned.

These findings provide more support and justification for the idea that apoptosis and neurogenesis in the hippocampus promote increased and sustained learning ability. We have accounted for differences in learning ability by demonstrating the deleterious effects of both saturated and silent neurons, effects that can be mitigated through the mechanism of cell death and replacement.

Finally, we have introduced some ideas about the theoretical information capacity of neural networks, as illustrated by the alphabet learning experiments. We expect these findings to inform the development of procedures and rules of thumb for extracting good performance out of neural network frameworks in the future.

Appendix D

Synchronization in Social Tasks: Robotic Drumming [17]

D.1 Introduction

Humans are particularly good at predicting and harmonizing with a rapidly changing environment in real-time. For example, many activities involving coordinated movement, such as dancing and playing catch, require fast decision-making, based on partial information. This behavior may best be understood under the general framework of synchronization [78], which has been used to model cognition in social tasks [28].

Synchronization tasks present significant difficulty to existing robot architectures. Musical performance, and drumming in particular, exemplifies the kind of synchronization challenge at which humans excel, and at which robots often fail. The musical environment, however, is relatively well-structured and constrained. This provides an excellent opportunity to explore solutions to synchronization problems, in a social setting, without oversimplifying the task or environment. As Breazeal's group [9]

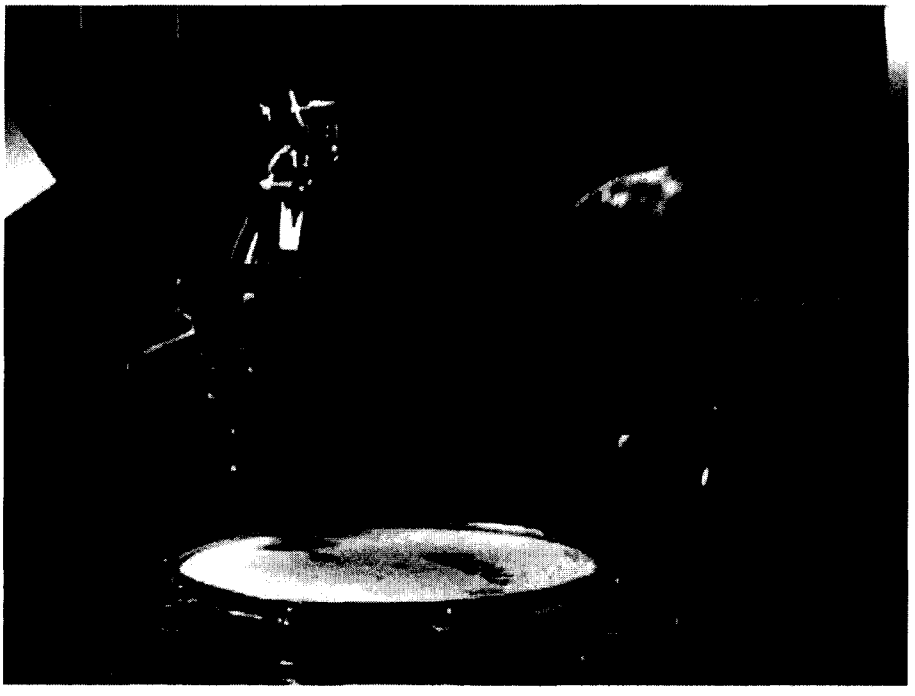

Figure D.1: Nico, drumming in concert with human performers. The robot responds to both visual and auditory stimuli from its human concert partners in order to produce the appropriate, correctly phased tempo.

suggests, this supportive scaffolding may be as necessary for humanoid robots in their development, as it is for infants in theirs.

While oscillator control provides an elegant solution to physical synchronization problems, its application in uncertain, complex sensory environments presents significant challenges. A robot attempting to drum in synchrony with human performers must perceive, classify and predict human actions, compensating for incomplete and uncertain sensory input in real time. The range of visual and auditory processing tasks involved contrasts sharply with the immediate, reliable input on which force feedback-based oscillator control architectures rely.

A successful robot drummer must spend time processing the perceptual signals it receives in order to recognize a rhythm. Given this information, it then needs to predict when the next beat should occur, taking into account variations in sensing

and processing time. Furthermore, its arm motion must begin exactly far enough in advance of the predicted beat so that the drum strike happens at the correct time, perfectly in phase with the humans. Finally, it must detect its own mistakes and use them to refine its model of the delays involved in sensing, processing and actuation.

We programmed a humanoid robot, Nico (shown in Figure D.1), to play a drum in concert with human drummers and at the direction of a human conductor. This process involved no preconceived idea of Nico's own physical dimensions or performance characteristics; the robot had to work out its internal dynamics for itself. Nico's dimensions match those of the average one-year-old male, and it has a finely articulated arm, head and shoulder [65] [90] [40]. In our experiment, Nico's ability to attune to the beat of human performers provides a clear, practical measure of the efficacy of our approach.

D.1.1 Related research

Musical perception and performance has been recognized as a fundamental aspect of human development, the study of which provides insight into speech development, acculturation, emotional development, and group social interaction [94]. Musical tasks have also proved a rewarding testbed for humanoid robotics. Robot drumming was previously explored by Hajian [44] and Williamson [99]. The drumming robot described by Hajian [44] exploited the dynamics of the task by using a low impedance drumstick holder, taking advantage of the drumstick's bounce to produce high frequency drumming. Robotic musicianship was also explored by the AI research group at Waseda University [55]. Their Wabot-2 robot had 50 degrees of freedom and was able to read a musical score and reproduce it on the piano.

The history of entertainment robotics includes many examples of musical robots, the exemplar of which is the player piano [73]. In recent years, music performance

has been a key element in the demonstration and marketing of commercial robots. Notably, Toyota's Partner robot has a large number of actuators for lips and breath control, enabling it to play wind instruments (e.g. the trumpet) with great precision [93].

From the humanoid robotics literature, our work can be best understood as an extension of the work of Williamson [99], who used oscillators (implemented as neural networks) to exploit the natural dynamics of rhythmic action such as drumming. Appropriate to non-social tasks like crank turning, this research focuses on the synchronization of the robot's control system to proprioceptive signals and external physical oscillators. In a social context, however, it becomes necessary to detect and synchronize with the rhythmic activity of humans, and by extension, their intentions and perceptions. Accordingly, our focus is on the attentional dynamics of the system.

None of these musical robots, have incorporated a social dimension, demanding intimate interaction with humans and the environment. Rather, even the most sophisticated robot performers follow preprogrammed scripts. By contrast, Nico's performances, though musically rudimentary, emerge entirely from watching, listening to, and participating in its environment. The robot learns and adjusts its behavior by observing and reasoning about itself and its human partners in real time. Our work attempts to model the psychophysical integration of auditory and visual information, using this biological principle to enhance robotic proprioception, as suggested by Williamson [99]. Likewise, our multi-oscillator architecture and approach are informed by Dynamic Attending Theory [28]. This theory proposes that humans, when listening to a complex audio sequence, begin by focusing their attention based on an internal *referent rate*, and dynamically shift this attentive rate to form a comfortable internal rhythmic representation of the sounds they hear. It includes the concept of *hierarchical levels* of rhythmic attention, which may provide a framework for the

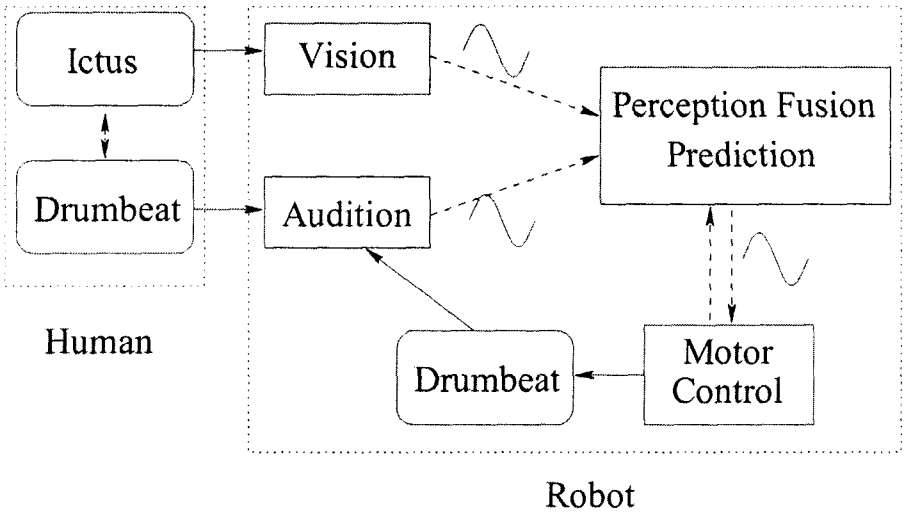

Figure D.2: High-level system design

development of further sophistication in Nico's perceptual capabilities.

Furthermore, our approach reflects recent work in developmental studies of rhythm learning in infants [45]. By the age of one, an infant can discriminate between sounds on the basis of their implied meter and rhythmic structure. Our developmental formulation of Nico's behaviors and musicianship finds parallels in early human development.

The architecture of Nico and the design decisions behind it are extensively discussed in several papers [65] [90] [40]. Gold [40] investigates the use of correlation in timing between the actions that Nico takes and the movements that it sees, to develop the capacity for self-recognition. Our research extends this idea, using the self-knowledge thus obtained to generate precisely-timed actions in a complex environment.

D.2 Methodology

The overall architecture of the system is summarized in Figure D.2. The vision and audition subsystems each produce a stream of event detections, while the motor control subsystem consumes a stream of arm-swing commands. The vision subsystem recognizes the arm motions of a human drummer or conductor, while the audition subsystem detects the drumbeats of both human and robotic performers. A high-level control subsystem (marked "Perception Fusion and Prediction" in Figure D.2) ties the others together, using learning, analysis and prediction, to generate high-level arm commands. Finally, those commands are translated into low-level motor directives, which cause Nico to tap a snare drum with its end effector.

D.2.1 Visual perception

The problem of visual perception is a difficult one, in a robotic context or indeed in a human one. A musician in a band or orchestra must, while playing an instrument and listening to his surroundings, obtain precise timing information from a conductor who may be standing at a distance, in difficult lighting. Thus, conducting gestures are designed to communicate timings as clearly as possible. This has the happy effect of simplifying Nico's recognition problem as well.

Our visual perception routine looks for the *ictus* of the conductor's beat, the point in time at which the conductor's hand "bounces" off an imaginary line, indicating the beat. In order to be useful, however, the ictus must be detected quickly. Our algorithm, as described below, performs very simple threshold-based filtering, and then computes the average vertical position of the moving, skin-colored objects in the visual field, as influenced by its previous position estimate. It uses a history of these positions to generate a trajectory, which it then analyzes to determine the proper

timing of detected beats. The system does not currently attempt to distinguish the beat pattern (differentiating the downbeat, say, from other beats in a measure).

1. For each pixel of a 320x240 frame,
   (a) *Skin detection*: If the red component of the pixel is higher than the blue and green components, and
   (b) *Motion detection*: If the difference between the combined RGB intensity of this pixel and the one in the same position in the previous frame is more than 100 (on a 0-255 scale), then
   (c) Mark this pixel.
2. Average the vertical positions of every marked pixel within 50 pixels of the last computed vertical centroid to form the current vertical centroid.
3. If the current vertical centroid is higher than the previous centroid, and
4. If the previous centroid was lower than any of the previous 10 centroids, then
5. The ictus of a beat has been detected.

This algorithm takes advantage of the fact that a conductor's arm movements are smooth and regular, and that human skin, specifically the conductor's hand, is relatively easy to detect through simple color filtering. In many settings, the reddest pixels, especially if they also contain some green, turn out to be skin [58].

D.2.2 Audition

Nico has two small microphones for ears. A simple intensity analysis is sufficient to accurately detect the sound of a drum beat in the audio stream. First the absolute value of the data is taken to find its intensity. Then the intensity values are

"smoothed", by taking the average of a window of a fixed number of samples. If a value is lower than a minimum threshold, it is replaced with 0, otherwise it is replaced with 1.

This is defined by the analysis function $a = t \circ s$ where w is the smoothing window size, m is the minimum intensity threshold, $r(x)$ is the audio sample at time x, s is the smoothing function and t is the threshold function.

$$s(x) = \frac{\sum_{n=(x-w)}^{x} |r(n)|}{w}$$

$$t(x) = \begin{cases} 0 & \text{where } x \leq m \\ 1 & \text{where } x > m \end{cases}$$

Finally, a search for edges within this binary stream is performed. Any change in value that lasts for longer than a minimum time threshold is taken as either the beginning of a new drum hit (when switching from 0 to 1) or the end of one (when switching from 1 to 0).

Some limitations of this scheme are that 1) all drums sound the same to Nico, and 2) any sufficiently loud noise may produce a false positive in drum beat detection. The second limitation is not too severe, at least in the friendly environment of the lab. By quieting the audience and by adjusting the microphone gains and the thresholds used in the audio analysis algorithms, false positives can be minimized. The first limitation, however, requires extra, high-level processing to distinguish the drum beats that are detected from Nico's action from those played by the musicians accompanying Nico. This classification is discussed further in section D.2.4.

D.2.3 Motor control

Nico's arm contains six revolute joints, which roughly correspond to the joints in a human infant's arm. Basic motor control for Nico's drumming motion is implemented

in software, whereby a "strike" command is translated into low-level physical motor directives. The robot calibrates its arm position against the placement of the drum – the drum need not be in exactly the same position for each trial.

The arm placement routine raises the arm to a fixed position over the drumhead, relative to the arm's resting position. It then manipulates its wrist motor until the robot's end effector comes into contact with the drum head. Fixing its wrist in this position, it produces a drum beat by raising and lowering its forearm from the elbow, so that the drum is struck each time the forearm is at its lowest position. The robot monitors the proprioceptive feedback of the elbow and wrist motors, allowing it to determine roughly the point at which it strikes the drum.

This motion is simplistic – human drumming motions are generally more efficient and employ more joint movement – yet adequate for the task. The entire drumming motion takes about 800 ms, limiting Nico's drumming speed to about 75 bpm.

D.2.4 Sensory integration and prediction

As Nico's control program runs, it collects a history of its inputs and outputs as a set of arrays of timestamps – one for visual ictus detections, one for audible drumbeat detections, and one for arm motion commands. These three streams are sufficient for the learning required for the robot to attune its drumming performance with its human partners. In addition, it correlates the audible drumbeat detections with the proprioceptive sensation of extending its arm to the point of impact with the drum.

The system has several sources of error and indeterminacy, for which Nico gradually learns to compensate. First of all, the arm's tapping motion takes a significant and variable amount of time from the moment the controller sends the arm command to the time when Nico's end effector actually strikes the drum head. Nico learns to adjust for this delay by examining the time difference between when it swings its

arm and when it detects a corresponding drumbeat in its audio stream, as well as the proprioceptive sense of hitting the drum.

In an ensemble performance, this task is made more difficult by the need to classify sounds as self-generated or external. Accordingly, we provide a warmup period during which Nico is allowed to beat its drum by itself. After it has learned its own internal timings, it can use this self-knowledge to classify the drumbeat events it encounters during a performance. The audio stream should contain a pattern which differs from the arm-motion command history only by phase. This difference is, roughly, the amount of time that it takes to swing the arm, and it is precisely this value that Nico learns in the warmup period.

Nico must also learn to associate the streams of visual and auditory events that it encounters. The visual stream contains only inputs from others – Nico's gaze is directed toward the conductor or drummer, and away from itself. The auditory stream, on the other hand, contains the effects of both human and robot action. Only by looking at both streams at the same time can Nico properly classify its audio input, and thereby estimate the tempo (both frequency and phase) to which it must attune.

The warmup period is followed by a period of observation where Nico only listens and watches its counterparts, so as to learn the association between ictus and drumbeat events. After a few seconds, Nico's attentional oscillators are attuned to the ensemble's tempo. Only then is its arm engaged, allowing the robot to join the performance.

This causal relationship between stimulus and action does not mean that Nico's attentive oscillators merely act in resonance to external stimulus. Nico is programmed with a referent period (60 bpm) and its oscillators are active and self-sustained. While an initial "push" is required to start Nico's arm moving, it will

continue to play its drum after the external stimulus terminates, at the tempo to which it has attuned. Likewise, during performance, sudden, brief changes in tempo do not disturb the robot's playing as they would if it were programmed to passively resonate to stimulus.

The following algorithm outlines the processing task.

1. *Sensory Integration*: Ictus events are associated with corresponding audio events. If events from two different sensory systems correspond, then this likely indicates a valid beat perception.

2. *Self-Awareness and Classification*: Arm motion events (both motion commands and proprioceptive sensations) are associated with corresponding audio events. This pattern is then compared against the beats found in the prior step, to determine the degree to which Nico is synchronized with its partners.

3. *Attunement*: An approximation of the external tempo is determined from the median of the intervals between the beats that have been deemed reliable in the first step.

4. *Prediction*: This tempo is used to calculate the next timepoint at which a beat will occur.

5. *Action*: Nico's self-knowledge is then used to predict the best time to next initiate an arm motion. If the perceived tempo is too fast, given its learned physical limitations, then the controller places the motion initiation time two beats (or more, if necessary) in the future.

The final step of this algorithm assures that robot performs effectively even when its physical limits are surpassed. When the tempo is too fast, Nico will still keep time, albeit on alternate beats. In the terms of Dynamic Attending Theory, it finds

a lower *hierarchical level* to which it may comfortably attune. This behavior nicely parallels the results from Drake's synchronized tapping experiments [28].

D.2.5 Performance

We developed two types of trials and variations, which we repeated several times, noting behaviors that seemed consistent in each. The first type of trial involves an interaction between a human musician and Nico. Both the human and Nico play the same drum. In the second type of trial, there are two humans – a conductor and a drummer who plays a different drum than Nico does. The conductor makes traditional conducting gestures, and the human drummer follows this direction as closely as he can. In order to ensure that Nico followed the humans' lead, rather than vice versa, the conductor and drummer used a silent, flashing-light metronome (invisible to Nico) to enforce tempo accuracy.

Each trial begins with the "warmup" period discussed in section D.2.4, where Nico has a chance to independently explore its arm motion. Then the human drummer (and conductor, if there is one) begin(s) playing at a steady tempo. After an "observational" period during which Nico quietly watches and listens to its human companions, it begins to play. In the the most forgiving variation, the humans continue playing at a fixed tempo for an extended period of time, after which they stop playing. In another variation, the humans shift to higher and lower tempos at various points during the trial, spending tens of seconds at each tempo. The third variation on the testing procedure involves gradual tempo shifts between low and high values, over an extended period of time.

D.3 Results

We collected our performance data from the event streams that Nico uses for learning and prediction, giving us precise timing relationships for every action Nico perceived or initiated. In addition, we captured the performances on video, allowing us to reconstruct and validate the event streams by hand. The results shown here come from two representative test runs, out of 14 for which we recorded program data. Both tests involve one person drumming with Nico, and no conductor. In this test configuration, the "ictus" was generated by the motion of the drummer's arm, rather than the conductor's. In the figures below, "Arm Swing" indicates the timepoints at which Nico's control software generated a command to initiate its arm swing, "Drum Detect" indicates the timepoints at which the audio processing subsystem detected a drumbeat, and "Ictus" indicates the timepoints at which the visual processing subsystem detected the change in the direction of motion of the human drummer's arm (corresponding to the drum being struck). The horizontal axis of each graph is measured in seconds.

In the first test, the human drummer (using a silent, flashing metronome to keep the beat) played the drum at a fixed tempo of 50 bpm. 12 seconds later, at the 66-second mark of the recording, Nico makes its first arm swing, and achieves the correct beat immediately. The data from the 70 - 80 second interval shows the high accuracy of Nico's performance (Figure D.3). In several places during this interval, the ictus signal is lost, but the performance is not disturbed.

In the second test run, a human drummer started with a low tempo and then switched to a higher tempo after about 50 seconds had elapsed. In a subsequent interval (Figure D.4), Nico loses the beat momentarily, and then recovers it.

Not long after, however, the drummer changes to a new, faster tempo, and Nico

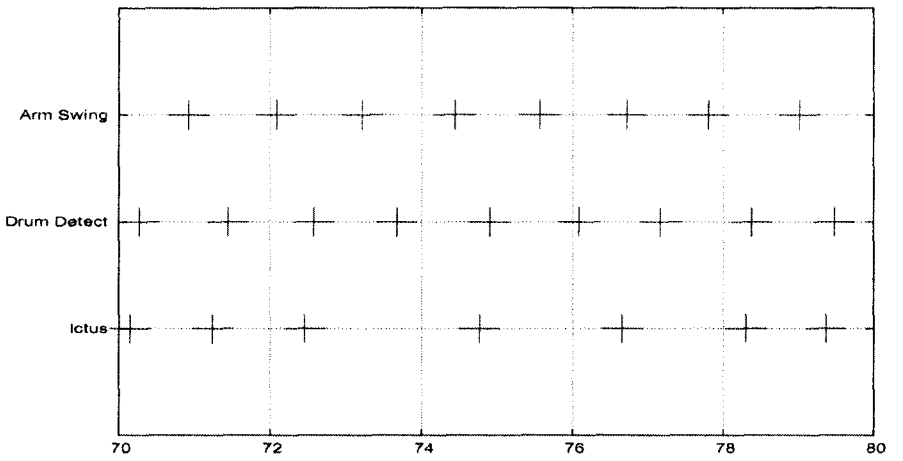

Figure D.3: Test run #1, seconds 70-80. Nico performs in perfect synchrony with the human musician. Nico's synchronization with the human drummer is so exact that their drumstrokes sound as one. Accordingly, Nico's audition subsystem detects only one drum beat every 5/6 second. The interval between arm swing initiation, in software, and the corresponding drumbeat detection, can also be seen by comparing the pattern in the "Arm Swing" and "Drum Detect" event streams.

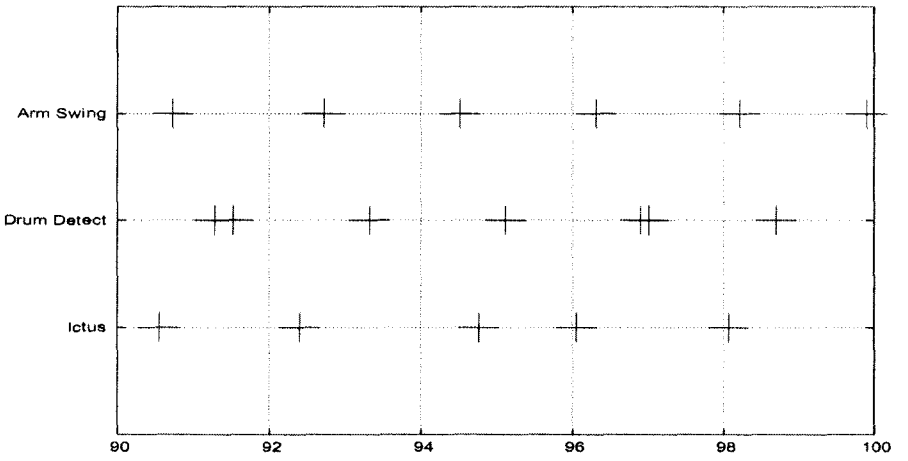

Figure D.4: Test run #2, seconds 90-100. 30 seconds after the human drummer started playing, the ensemble is still at a low tempo. In intervals 90-92 and 96-98, Nico is not in perfect synchrony with the human drummer. Each of these intervals contains one arm swing initiation, but *two* drum beat detections. In these instances, Nico has struck its drum either before or after its human counterpart, by a noticeable amount, thereby generating the "extra" drumbeat.

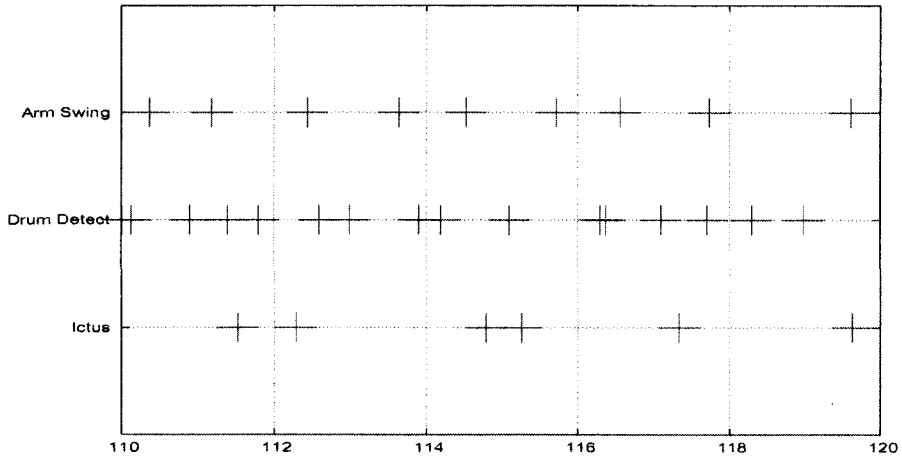

Figure D.5: Test run #2, seconds 110-120. Although the human drummer is drumming regularly in this period, Nico's drumming is out of phase and frequency with its human counterpart. As a result, the pattern of drumbeat detections is irregular. The irregular arm swing data from this section confirms that Nico has yet to converge on a stable pattern. This behavior continues for 30 seconds after the interval shown here.

loses the beat for a second time (Figure D.5). This state of discord continues until the drummer slows a bit, and Nico begins to regain synchrony (Figure D.6). A few seconds later, Nico returns to a perfect unison performance with the human drummer.

D.4 Discussion

In the simplest scenario, when the oscillating patterns presented to the robot's eyes and ears maintain a constant tempo, Nico's entrainment usually occurs within a few seconds. When the tempo shifts, however, Nico may lose the beat temporarily, as it struggles to account for the varying intervals between recent reliable beats. Nico makes all of its calculations based on actual beat detections, so it has no way of knowing that an accelerando is happening until it detects a beat earlier than it expects. Human performers, by contrast, note the accelerating downstroke of a

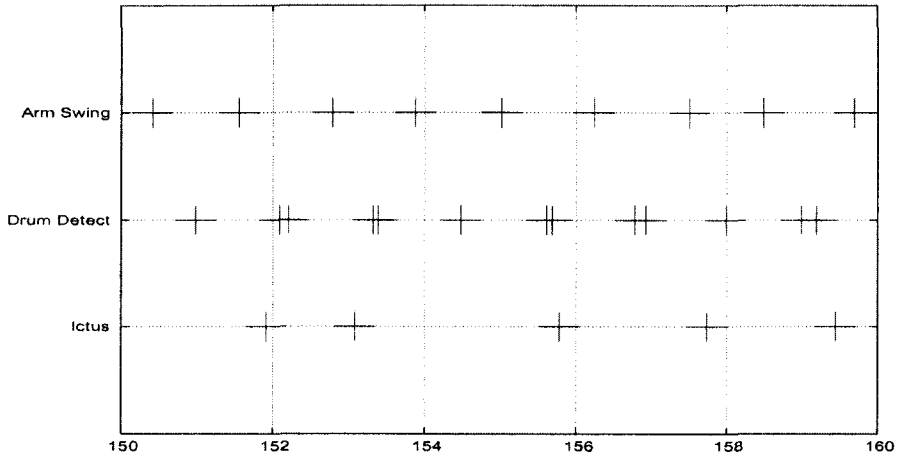

Figure D.6: Test run #2, seconds 150-160. This interval shows Nico in the process of re-converging on a stable pattern. The consistent intervals between arm swings indicate that the robot has established the basic frequency, and is now experimenting to lock in the precise phase shift. Nico's beats occur alternately just ahead, just behind or exactly in time with the human drummer's. A few seconds after the interval shown here, Nico works out the correct timings and enters perfect synchrony with the human drummer. No figure is shown for that period since it is very similar to the interval shown in Figure D.3.

conductor's arm *before* the actual beat occurs, and thus can follow the indicated tempo shift without the lag time that Nico shows, even in its best performances. Drake [28] found that the synchronization performance of humans degrades in an impoverished sensory environment – for example, synchronization with a metronome is much more difficult than with the beats of a real musical performance. Likewise, Nico has access to very little of the context of the oscillatory input it receives.

Interplay between the auditory and visual inputs helps provide some of the necessary context. As the tempo accelerates, ictus detection becomes less and less reliable, and the beneficial effect of sensory integration lessens. When Nico's oscillator is well-entrained to the beat provided by its human partners, the loss of reliable visual perceptions doesn't much matter. Even without support from the visual system, Nico can hear that drumbeats occur exactly when it expects them, and therefore maintains the proper frequency and phase. In this difficult environment, once Nico

loses the beat, it has a hard time finding it again – especially if it manages to correlate an erratic ictus indication with one of its own erroneous drumbeats. It will tend to play erratically until it chances to get several solid perceptually-fused beats in a row, which it can then use to fix its own behavior. This accounts for the pattern we see in performance, where Nico plays very badly for half a minute or so, and then suddenly catches itself and plays perfectly for a while.

When Nico is having trouble finding the beat, it often settles into the same error conditions that Dynamic Attending Theory [28] would predict. Nico will often entrain a harmonically-related frequency to the one which its human partners present. For example, if the tempo shifts suddenly from a fast one to a slow one, Nico often entrains a frequency exactly double the true one. The robot's auditory input provides a steady, accurate-sounding oscillation (even though half the beats are entirely self-generated), and the video input confirms every other beat. Since the robot assumes that it will sometimes fail to detect an ictus, it assumes that it is playing correctly, even though fully half of its drumbeats are off.

Finally, Nico encounters grave difficulties when the tempo is close to its maximum physical ability to beat. If the tempo is *very* fast, Nico reliably chooses to play every other beat, and problems don't emerge. If Nico tries to play two beats in quick succession, however, the arm sometimes fails to respond in time, and the drumbeat is delayed. To Nico, this beat does not look like a self-generated one, because the timing is wrong. It thus credits this delayed beat to its drumming partner, and tries to adjust the tempo accordingly.

Musicmaking is a cooperative, social task, and as such everyone involved wants it to succeed. As mentioned previously, we used a metronome to enforce a strict tempo among the human performers, but to do so challenged our ingrained propensity to adapt our drumming whenever Nico started to have difficulty. Human ensembles do

this all the time – when a performer makes a mistake, the ensemble tries to accommodate it as smoothly as possible, adjusting their own playing to compensate. This mutual behavioral reinforcement is a characteristic of teaching with play [9]. Nico's active, self-sustained behavior presents, to the human participants, an opportunity for attunement and play. Interestingly, this engagement is evoked by the relatively simple, oscillator-based mechanism of Nico's control system.

D.5 Conclusion

Our research demonstrates an effective method for fusing diverse sources of oscillatory input of varying accuracy and phase shift in order to produce a reliable response in a social robot. Drumming, as a social task, proved to be well-suited for this experiment, as it requires sensory integration to compensate for the robot's limited perceptual ability. Our work also highlights what we believe are the essential aspects of competence in music making – perception and prediction, in contrast to the many examples of robotic musicianship that focus on mechanics. In addition, we have reconfirmed the benefit of developmental and behavioral approaches in the solution of real-time social tasks.

Functionally, it would be useful to extend the architecture described here with simple abstract models for music representation such that Nico may play a piece of music more sophisticated than simply striking the drum on every beat. Learning may be used to associate tempo values with sections of music, to provide better prediction in the performance of a given score. A richer representation would also provide more context for Nico's entrainment, hopefully replicating the improved performance that such context allows in humans [28].

Currently, our design is well-informed by current psychological models, but the

implementation is only vaguely biologically-inspired. Specifically, we may be able to increase performance and reliability by refining our architecture for sensory integration based on more sophisticated psychophysical models. A formalized feedback control system, such as employed by Mukherjee [69], would allow much more sophisticated drumming motions and would improve Nico's ability to characterize different performance regimes.

Finally, the architecture explored here may be applied to other social tasks. For further research involving Nico, it would be relevant to explore the synchronization tasks in which a typical one-year-old might engage. This might include object manipulation, reaching, and game playing.