How A.I. and multi-robot systems research will accelerate our understanding of social animal behavior

Tucker Balch, Frank Dellaert, Adam Feldman,
Andrew Guillory, Charles Isbell, Zia Khan,
Andrew Stein, and Hank Wilde

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

Abstract

Our understanding of social insect behavior has significantly influenced A.I. and multi-robot systems' research (e.g. ant algorithms and swarm robotics). In this work, however, we focus on the opposite question, namely: "how can multi-robot systems research contribute to the understanding of social animal behavior?." As we show, we are able to contribute at several levels: First, using algorithms that originated in the robotics community, we can track animals under observation to provide essential quantitative data for animal behavior research. Second, by developing and applying algorithms originating in speech recognition and computer vision, we can automatically label the behavior of animals under observation. Our ultimate goal, however, is to automatically create, from observation, *executable* models of behavior. An executable model is a control program for an agent that can run in simulation (or on a robot). The representation for these executable models is drawn from research in multi-robot systems programming. In this paper we present the algorithms we have developed for tracking, recognizing, and learning models of social animal behavior, details of their implementation, and quantitative experimental results using them to study social insects.

1 Introduction

Our objective is to show how robotics research in general and multi-robot systems research in particular can accelerate the rate and quality of research in the behavior of social animals. Many of the intellectual problems we face in multi-robot systems research are mirrored in social animal research. And many of the solutions we have devised can be applied directly to the problems encountered in social animal behavior research.

One of the key factors limiting progress in all forms of animal behavior research is the rate at which data can be gathered. As one example, Deborah Gordon reports in her book that two observers are required to track and log the activities of one ant: One person observes and calls out what the ant is doing, while the other logs the data in a notebook [20]. In the case of *social* animal studies, the problem is compounded by the multiplicity of animals interacting with one another.

One way robotics researchers can help is by applying existing technologies to enhance traditional behavioral research methodologies. For instance computer vision-based tracking and gesture recognition can automate much of the tedious work in recording behavior. However, in developing and applying "existing" technologies, we have discovered new, challenging, research problems. Vision-based multi-target tracking, for instance, is far from being a solved problem. Algorithms capable of tracking several targets existed when we began our work in this area, but research with ants calls for software that can track dozens or hundreds of ants at once. Similarly, algorithms for labeling the behavior of one person (or animal) existed several years ago, but now we must label social behavior between two or more animals at once. The point being that the application of robotics algorithms to a new domain also drives new research in robotics.

Now, in addition to helping animal behavior researchers speed up their traditional modes of research, we can also provide them with entirely new and powerful tools. In particular, mobile robotics can offer new ways to represent and model animal behavior [44, 45]. What is the best model of behavior? It is our position that an *executable* model provides the most *complete* explanation of an agent's behavior. By "executable" we mean that the model can run in simulation or on a mobile robot. By "complete" we mean that all aspects of an agent's behavior are described: from sensing, to reasoning, to action. Executable models provide a powerful means for representing behavior because they fully describe how perception is translated into action for each agent. Furthermore, executable models can be tested experimentally on robots or in a multi-agent simulation to evaluate how well the behavior of a simulated "colony" matches to the behavior of the actual animals. Other types of models cannot be tested in this way.

Robotics researchers are well positioned to provide formalisms for expressing executable models because the programs used to control robots are in fact executable models. Furthermore, researchers who use behavior-based approaches to program their robots use representations that are closely related to those used by biologists to describe animal behavior. Our objective in this regard is to create algorithms that can learn executable models of social animal behavior directly from observations of the social animals themselves.

In this paper we report on our progress in this area over the last six years. A primary contribution is the idea that we ought to apply A.I. and robotics approaches to these problems in the first place. However, additional, specific contributions include our approaches to solving the following challenges:

- **Tracking multiple interacting targets:** This is an essential first step in gathering behavioral data. In many cases, this is the most time consuming step for a behavioral researcher.
- **Automatically labeling behavior:** Once the locations of the animals are known over time we can analyze the trajectories to automatically identify certain aspects of behavior.
- **Generating executable models:** Learning an executable model of behavior from observation can provide a powerful new tool for social behavior research.

The rest of the paper follows the order listed above: We review the challenges of multi-target tracking and our approach to the problem. We show how social behavior can be recognized automatically from trajectory logs. Then we close with a presentation of our approach to learning executable models of behavior. Each topic is covered at a high level, with appropriate citations to more detailed descriptions. We also review the relevant related work in each area in the beginning of each corresponding section.

2 Tracking Multiple Interacting Targets

Tracking multiple targets is a fundamental task for mobile robot teams [32, 41, 24]. An example from RoboCup soccer is illustrated in Figure 1: In the small-size league, 10 robots and a ball must be tracked using a color camera mounted over the field. In order to be competitive, teams must find the location of each robot and the ball in less than $\frac{1}{30}$ of a second.

One of the leading color-based tracking solutions, CMVision was developed in 2000; it is used now by many teams competing at RoboCup [10]. We used CMVision for our first studies in the

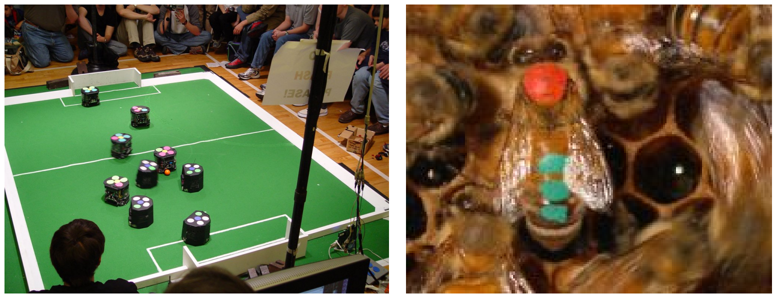

Figure 1: Multi-target tracking in robot soccer and social animal research. Left: In the RoboCup small size league robots are tracked using an overhead camera (image courtesy Carnegie Mellon). Right: A marked honey bee is tracked in an observation hive.

analysis of honey bee dance behavior [17]. However for bees, color-based tracking requires marking all the animals to be observed (Figure 1) — this is something of a tricky operation, to say the least. Even if the animals are marked, simple color-based tracking often gets confused when two animals interact.

We now use a probabilistic framework to track multiple targets by an overhead video camera [29]. We have also extended the approach for multiple observers (e.g. from the point of view of multiple sensors or mobile robots). Comprehensive details on these approaches are reported in [4, 28, 29, 33].

Our objective in this first stage (tracking) is to obtain a record of the trajectories of the animals over time, and to maintain correct, unique identification of each target throughout. We have focused our efforts on tracking social insects in video data, but many of the ideas apply to other sensor modalities (e.g. laser scanning) as well. The problem can be stated as:

- **Given:**
  1. Examples of the appearance of the animals to be tracked; and
  2. The initial number of animals.
- **Compute:** Trajectories of the animals over time with a correct, unique identification of each target throughout.
- **Assume:**
  1. The animals to be tracked have similar or identical appearance;
  2. The animals' bodies do not deform significantly as they move; and
  3. The background does not vary significantly over time.

2.1 Particle Filter Tracking

Traditional multi-target tracking algorithms approach this problem by performing a target detection step followed by a track association step in each video frame. The track association step solves the problem of converting the detected positions of animals in each image into multiple individual trajectories. The multiple hypothesis tracker [36] and the joint probabilistic data association filter (JPDAF) [5, 18] are the most influential algorithms in this class. These multi-target tracking algorithms have been used extensively in the context of computer vision. Some example applications are the use of nearest neighbor tracking in [15], the multiple hypothesis tracker in [13], and the JPDAF in [35]. A particle filter version of the JPDAF was proposed in [38].

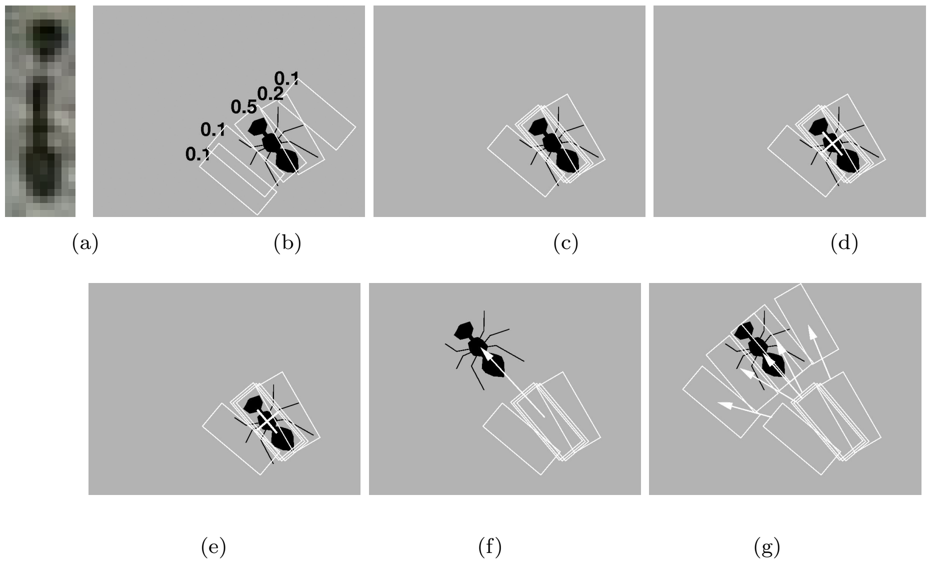

Figure 2: Particle filter tracking. (a) The appearance model used in tracking ants. This is an actual image drawn from the video data. (b) A set of particles (white rectangles), are scored according to how well the underlying pixels match an appearance model. (c) Particles are resampled according to the normalized weights determined in the previous step. (d) The estimated location of the target is computed as the mean of the resampled particles. (e) The previous image and particles. (f) A new image frame is loaded. (g) Each particle is advanced according to a stochastic motion model. The samples are now ready to be scored and resampled as above.

Traditional trackers (e.g. the Extended Kalman-Bucy Filter) rely on physics-based models to track multiple targets through merges and splits. A merge occurs when two targets overlap and provide only one detection to the sensor. Splits are those cases where a single target is responsible for more than one detection. In radar-based aircraft tracking, for instance, it is reasonable to assume that two planes that pass close to one another can be tracked through the merge by predicting their future locations based on their past velocities. That approach doesn't work well for targets that interact closely and unpredictably. Ants, for instance, frequently approach one another, stop suddenly, then move off in a new direction. Traditional approaches cannot maintain track in such instances. Our solution is to use a joint particle filter tracker with several novel extensions.

First we review the operation of a basic particle filter tracker, then we introduce the novel aspects of our approach. The general operation of the tracker is illustrated in Figure 2. Each particle represents one hypothesis regarding a target's location and orientation. For ant tracking in video, the hypothesis is a rectangular region approximately the same size as the ant targets. In the example, each target is tracked by 5 particles. In actual experiments we use hundreds of particles per target.

We assume we start with particles distributed around the target to be tracked. A separate algorithm (not covered here) addresses the problem of initializing the tracker by finding animals in the first image of the sequence. After initialization, the principal steps in the tracking algorithm

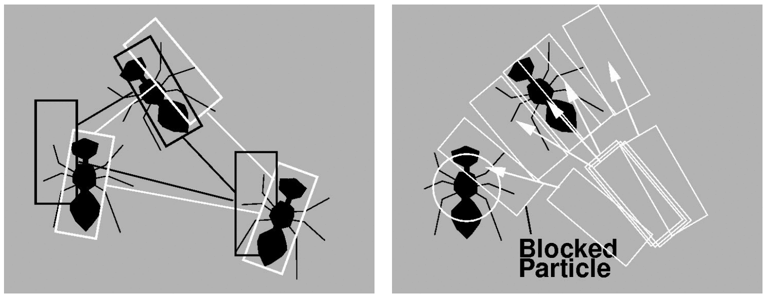

Figure 3: Joint particles and blocking. When tracking multiple animals, we use a joint particle filter where each particle describes the pose of all tracked animals (left). In this figure there are two particles – one indicated with white lines, the other with black lines. Particles that overlap the location of other tracked targets are penalized (right).

include:

1. **Score:** each particle is scored according to how well the underlying pixels match an appearance model.
2. **Resample:** the particles are "resampled" according to their score. This operation results in the same number of particles, but very likely particles are duplicated while unlikely ones are dropped.
3. **Average:** the location and orientation of the target is estimated by computing the mean of all the associated particles. This is the estimate reported by the algorithm as the pose of the target in the current video frame.
4. **Apply motion model:** each particle is stochastically repositioned according to a model of the target's motion.
5. **Load new image:** read the next image in the sequence.
6. **Go to Step 1.**

2.2 Joint Particle Filter Tracking

The algorithm just described is suitable for tracking an individual ant, but it fails in the presence of many identical targets [29]. The typical mode of failure is "particle hijacking" whereby the particles tracking one animal latch on to another animal when they pass close to one another.

Three additional extensions are necessary for successful multi-target tracking: First, each particle is extended to include the poses of all the targets (i.e. they are *joint* particles). An example of joint particles for ant tracking is illustrated in Figure 3. Second, in the "scoring" phase of the algorithm, particles are penalized if they represent hypotheses that we know are unlikely because they violate known constraints on animal movement (e.g. ants seldomly walk on top of each other). Blocking, in which particles that represent one ant on another are penalized, is illustrated in Figure 3. Finally, we must address the exponential complexity of joint particle tracking (this is covered below). These extensions, and evaluations of their impact on tracking performance, are reported on in detail in [28, 29].

The computational efficiency of joint particle tracking is hindered by the fact that the number of particles and the amount of computation required for tracking is exponential in the number of targets. For instance, 10 targets tracked by 200 hypotheses each would require $200^{10} = 10^{23}$ particles in total. Such an approach is clearly intractable. Independent trackers are much more efficient, but as mentioned above, they are subject to tracking failures when the targets are close to one another.

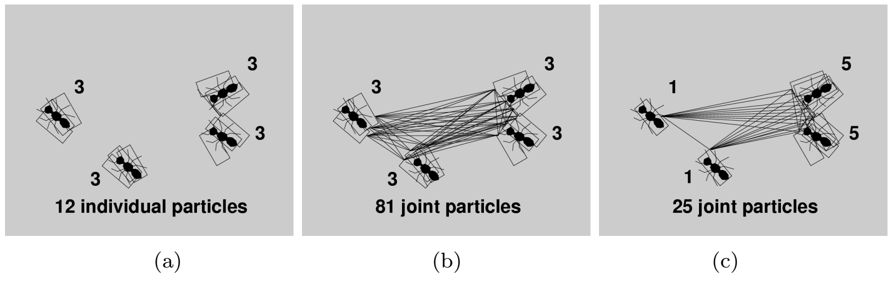

Figure 4: MCMC sampling. (a) Each ant is tracked with 3 particles, or hypotheses. With independent trackers this requires only 12 particles, but failures are likely. (b) With a joint tracker, to represent 3 hypotheses for each ant would require 81 particles altogether. However, MCMC sampling (c) enables more selective application of hypotheses, where more hypotheses are used for the two interacting ants on the right side, while the lone ants on the left are tracked with only one hypothesis each.

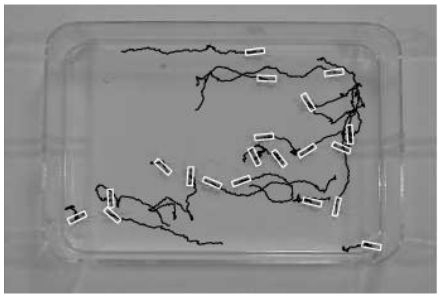

Figure 5: Example tracking result. 20 ants are tracked in a rectangular arena. The white boxes indicate the position and orientation of an ant. The black lines are trails of their recent locations.

We would like to preserve the advantages of joint particle filter tracking, but avoid an exponential growth in complexity. In order to reduce the complexity of joint particle tracking we propose that targets that are in difficult to follow situations (e.g. interacting with one another) should be tracked with more hypotheses, while those that are isolated should be tracked with fewer hypotheses. We use Markov Chain Monte Carlo sampling to accomplish this in our tracker [28, 29]. The approach is rather complex, but the effective result is that we can achieve the same tracking quality using orders of magnitude fewer particles than would be required otherwise. Figure 4 provides an example of how the approach can focus more hypotheses on challenging situations while using an overall smaller number of particles. An example result from our tracking system using MCMC sampling is illustrated in Figure 5. In this example, 20 ants are tracked in a rectangular arena.

After trajectory logs are gathered they are checked against the original video. In general, our tracker has a very low error rate (about one of 5000 video frames contains a tracking error [29]). However to correct such errors and ensure nearly perfect data, we verify and edit the trajectories using TeamView, a graphical trajectory editing program also developed in our laboratory [3].

2.3 Using Tracking for Animal Experiments

Most of our efforts in tracking are focused on social insect studies. These are examined in more detail in the following sections. However, we have also been deeply involved with the Yerkes Primate Research center to apply our tracking software to the study of monkey behavior. In particular, we are helping them evaluate spatial memory in rhesus monkeys by measuring the paths the monkeys take as they explore an outdoor three dimensional arena over repeated trials. In this study we use two cameras to track the movements of monkeys inside the arena and generate 3-D data (see Figure 6). Over the course of this study we have collected over 500 hours of trajectory data – this represents the largest corpus of animal tracking data we are aware of. The design of our tracking system for this work is reported in [30].

In the initial study our system only tracked one monkey at a time. We are now moving towards an experiment where we hope to track as many as 60 monkeys at once in a 30 meter by 30 meter arena. To accomplish this we are experimenting with a new sensor system: Scanning laser range finders (Figure 7). Each laser scans a plane out to 80 meters in 0.5 degree increments. Example data from a test in our lab is shown in the figure on the right. In this image the outline of the room is visible, as well as 5 ovals that represent the crossections of 5 people in the room. We plan to extend and adapt our vision-based tracking software to this new sensor modality.

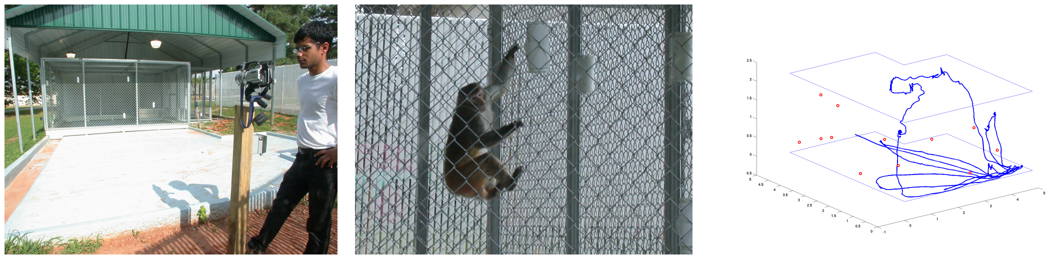

Figure 6: Left: Experimental setup at Yerkes Primate Research Center for tracking monkeys in an outdoor arena. Center: A monkey in the arena. Right: An example 3-D trajectory.

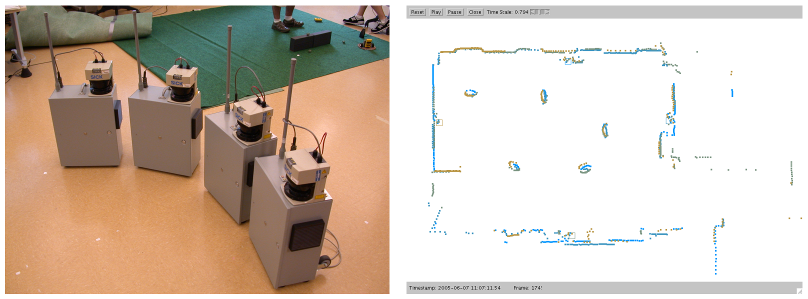

Figure 7: Left: Scanning laser range finders mounted on portable boxes for easy field deployment. Right: Data from an experiment in our lab. The outline of the room is visible, along with 5 oval objects – people in the room.

3 Automatic Recognition of Social Behavior

Behavior recognition, *plan recognition* and *opponent modeling* are techniques whereby one agent estimates the the internal state or intent of another agent by observing its actions. These approaches have been used in the context of robotics and multi-agent systems research to make multi-robot teams more efficient and robust [22, 43, 23]. Recognition helps agents reduce communications bandwidth requirements because they can infer the intent of their team members rather than having to communicate their intent. In other cases the objective is to determine the intentions of an opposing agent or team in order to respond more effectively. Related problems are studied in the context of speech recognition, gesture recognition, and surveillance [26, 14, 25]. Social animal research offers a similar challenge: how can we automatically recognize the behavior of animals under observation?

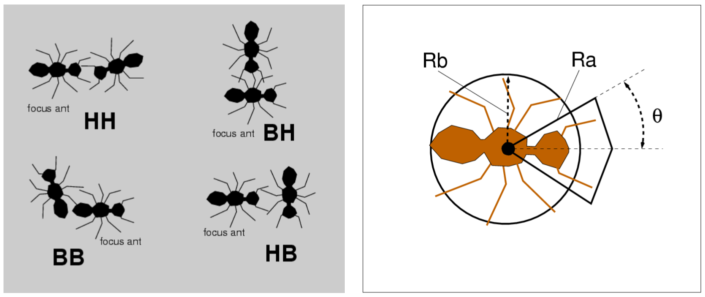

Figure 8: Left: We have examined the problem of detecting different types of encounters between ants: Head-to-Head (HH), Head-to-Body (HB), Body-to-Head (BH), and Body-to-Body (BB). Right: In one approach, we use a geometric model of an ant's sensory system. Parameters of the model include R_b the radius of the animal's "body" sensor, R_a the range of antenna sensing, and θ the angular field of view of the antennae.

As an example, in ants, encounters between animals are thought to mediate individual and colony level behavior, including for instance, forager recruitment, response to external disturbances and nest selection for the colony [42, 21, 19, 34]. Typically, behaviors such as these are investigated experimentally by having human observers record observed or videotaped activity. Generating such records is a time-consuming task for people however, especially because the numbers of interactions can increase exponentially in the number of animals.

Computer vision-based tracking and machine analysis of the trajectories offers an opportunity to accelerate such research. Once we have trajectories of the animals' movements, we can analyze them to infer various aspects of their behavior. We seek to recognize *social* behavior, so we focus on aspects of the animals' movement that reveal social interaction.

We have explored two approaches to this problem. In the first approach we take advantage of a model of the geometry of the animal's sensory apparatus to estimate when significant interactions occur. In the second approach we use example labels provided by a human expert to train the system to label activities automatically.

3.1 Using Sensory Models to Recognize Social Behavior

When ants approach one another, they often (perhaps deliberately) meet and tap one another with their antennae. We are interested in detecting and differentiating between different types of these interactions. Stephen Pratt has identified three types of encounters as the most significant for quorum detection in *Temnothorax curvispinosus* [34]. The interaction types, perceived from the point of view of a particular ant (the focus ant) are: Head-to-Head (HH), Head-to-Body (HB), and Body-to-Head (BH). It may also be the case that Body-to-Body (BB) interactions are important as well, so we include them for completeness (see Figure 8). It is not known for certain the extent to which the different types of encounters affect the behavior of the animals differently — in fact that is one purpose of the research we are conducting. Formally, we seek to address the following problem:

- **Given:**
  1. Timestamped trajectories of the pose of multiple interacting animals; and
  2. A geometric model of their sensory field.
- **Compute:** Appropriate labels for the trajectories, including encounters, automatically.
- **Assume:** The geometric models accurately reflect the sensory apparatus of the animals.

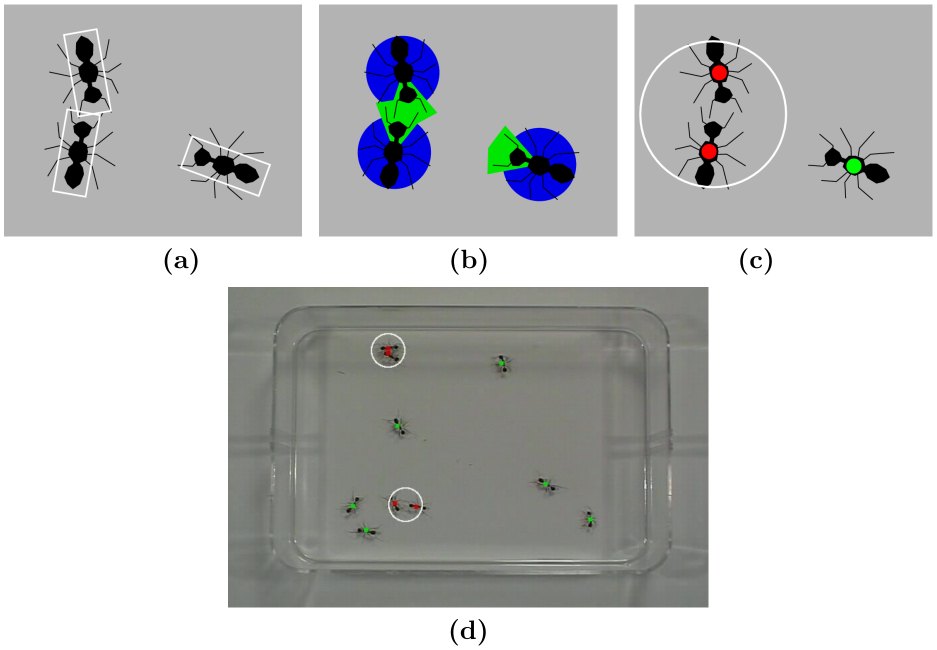
Figure 9: Ants are tracked and encounters between them are detected automatically in video images. (a) A vision-based multi-target tracker locates and tracks ants in video; (b) Geometric models of animal sensory fields are checked for overlap; (c) A head-to-head encounter is detected and indicated by a circle around the interacting animals; (d) Example video in which encounters are detected and highlighted.

To accomplish this we approximate an ant's antennal and body sensory fields with a polygon and circle, respectively (Figure 8). An encounter is inferred when one of these regions for one ant overlaps a sensory region of another ant. Our model is adapted from the model introduced for army ant simulation studies by Couzin and Franks [12]. Note that for this work we do not consider the visual field of the ant. While the visual field is certainly important for guiding ants we consider that it is not critical for determining whether physical contact occurs between them.

The following are details of our sensory model for *Aphaenogaster cockerelli*. Recall that the tracking software reports the location of the center of the ant and its orientation. We estimate the front of the head to be a point half a body length away from the center point along the centerline. From the head, the antennae project to the left and right at 45 degrees. We construct a polygon around the head and antennae as illustrated, with one point at the center of the ant's thorax, two at the outer edges of the antennae and an additional point, one antenna length away, directly along the ant's centerline in front of the head. The inferred "antennal field of view" is the resulting polygon defined by these points. We assume a body length of 1cm and antenna length of 0.6cm. We estimate an ant's "body" sensory field to include a circle centered on the ant with a radius of 0.5cm.

We assume that any object within the head sensory field will be detected and touched by the animal's antennae, and any object within the range of the body can be detected by sense organs on the legs or body of the ant. By considering separate sensory fields for the body and the the antennae, we are able to classify each encounter into one of the four types described above. To determine when an interaction occurs, we look for overlaps between the polygonal and circular regions described above.

Computationally, determining an overlap consists of checking for intersections between line segments and circles. Additionally, we require that two ants' orientations must differ by at least 5 degrees before an interaction can count as "head-to-head." Without this constraint, we noticed

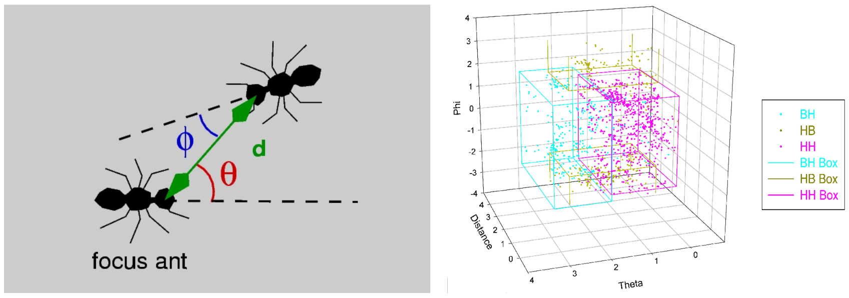

Figure 10: Left: Three features used to classify interactions between ants. Right: Training data is plotted in feature space. The boxes illustrate the thresholds used to identify each type of interaction.

many spurious "encounters" logged by the system for ants that were simply standing next to one another without actually interacting.

In order to evaluate the encounter detection system, we had a human observer and our detection software label interactions on the same experimental data (a videotape of ants interacting in an arena). In cases where the logs differed, we reviewed the video carefully at that time using slow-motion playback to determine whether or not an encounter actually occurred. We categorized each difference into one of four classes: *a)* computer false positives in which the computer identified an encounter that did not occur; *b)* computer false negatives in which the computer failed to identify an encounter that actually occurred; *c)* human false positives in which the human observer logged an encounter that did not actually occur; and *d)* human false negatives in which the human did not record interactions that actually occurred. We repeated this examination for three different videotaped experiments. Quantitative results with standard deviations are provided in the following table:

Computer		Human	
false positives	false negatives	false positives	false negatives
7% (2%)	0% (0%)	15% (2%)	29% (12%)

We can clearly benefit from automatic labeling in this application. Behavior labeling is faster and more accurate than human labeling. The primary weakness being that some detected encounters have not actually occurred as indicated by a 7% false positive rate.

3.2 Using Trainable Models to Recognize Social Behavior

The geometric sensory model-based method described above performs well, but it is difficult to apply in general because a new model would have to be devised for each application. Accordingly, we have also developed recognition methods that take advantage of input provided by human experts. In these systems, a human labels some of the experimental data (referred to as the training set). A model of the human's labeling is created, then used as a reference to label new examples. The power of this approach is that it does not require the human expert to elucidate the considerations involved in the classification of an event into one of several categories, it only requires examples of correctly classified observations. This is of course a well-studied problem. Variations of the problem are studied in computer vision for gesture recognition, face recognition, and activity recognition [40, 11, 14]. In our application, again, we are specifically interested in *social* behavior, so the problem for human-trainable recognition algorithms is specified as follows:

- **Given:**
  1. Timestamped trajectories of the pose of multiple interacting animals; and
  2. A set of quantitative features defined over (and between) the trajectories; and
  3. Labeled example trajectories.
- **Compute:** Appropriate labels for the trajectories, including encounters, automatically.
- **Assume:** The features are suitable for discriminating between the behaviors that should be identified, and the training examples provided are correct.

For the problem of detecting different types of interactions between ants, we selected three features of a potential encounter that can be easily determined from the timestamped trajectory data. The features are illustrated and described in Figure 10. We use the following steps to learn and apply behavior labels:

1. **Track:** Animals are tracked using the algorithms described above.
2. **Label:** At each frame, the behavior of each animal is labeled by a human observer (e.g. HH, BH, HB, or X).
3. **Cluster:** The data gathered above is processed in order to determine disjoint regions in feature space that enclose each type of interaction.
4. **Classify:** In online operation we determine in which region of feature space a particular observation lies, and label it accordingly.

In our present system, the third step of the procedure is a hill-climbing optimization. For each type of interaction (set of thresholds) we seek to maximize the number of data points with that label within the bounding box *minus* the number of data points with different labels inside the box. Each of the six thresholds is optimized in turn, with iterations continuing until there is no improvement between subsequent loops. The resulting thresholds determined from a 5 minute labeled sequence are illustrated in Figure 10.

When the system is run on new (unlabeled) data, it detects and labels interactions by computing the quantitative features at each timestamp, then finding which bounding box (if any) the point is enclosed by. It is possible that one ant may experience several interactions at once with more than one other ant (the system examines the three closest other ants). If there are multiple interactions, we select the highest priority type according to the order: HH, HB, BH, X. (Note that in these experiments we did not include Body-to-Body (BB) interactions).

Observe that this is a rather simple approach. There are more sophisticated algorithms that could also be applied including, for instance, kernel regression or decision trees [39, 1]. We compared the quality of our results with kernel regression and found no significant advantage for one over the other. We suspect that the feature space in this application lends itself well to the bounding box approach.

We evaluated the system by having a human expert label two five minute videos of *Leptothorax albipennis* searching a new habitat. In each segment ants enter the field of view, interact in various ways, then depart. There are hundreds of interactions in each video. We used the labeling of one video as training data for the system, which then labeled the other video. We then compared the automatic labeling of the test data to the human labeling of the same data. The automatic method correctly identified 94% of the interactions. We then reversed the order of training: we trained the system on the second video, then tested it on the first. In that case the system correctly identified 87% of the interactions. Example frames from video labeled by our system are reproduced in Figure 11.

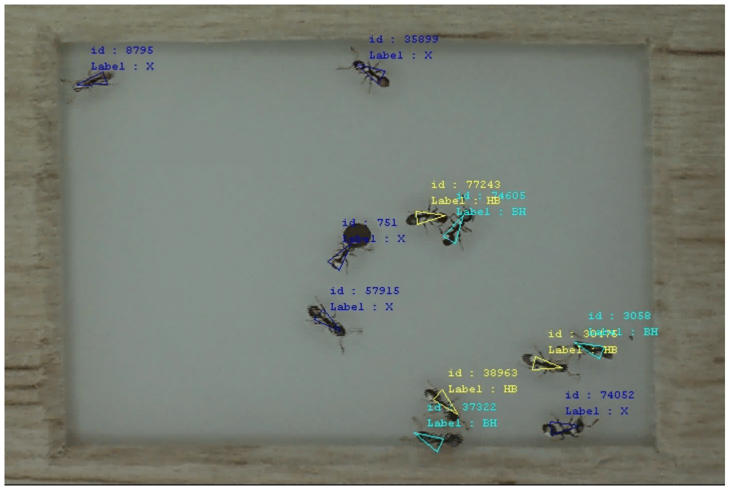

Figure 11: This image shows an example frame of video of *Leptothorax albipennis* labeled automatically by our human-trainable system. The colored triangles over the animals are coded for the different types of interaction that animal is experiencing (cyan: BH, yellow: HB, magenta: HH, blue: X).

This performance is good, and it compares favorably with similar systems used for gesture recognition (e.g. [40]). However, as many as 43% of the interactions reported by the system are false positives; meaning that the system detects many interactions that never occurred.

All is not lost however. In similar work in which we automatically labeled honey bee dance behavior, we achieved a similar 93% accuracy, but with a substantially lower false positive rate. The difference is that in the honey bee work we used a Hidden Markov Model (HMM) in addition to the feature-based labeling. HMMs provide a "smoothing" influence that tends to suppress brief "noisy" detections. As this work with ants progresses we will incorporate HMMs as well.

3.3 Using Automatic Recognition to Study Behavior in Animals

Our objective in this section is to show how multi-target tracking and automatic social behavior recognition can be used in the context of an animal study. We present some preliminary results from our own experiments, but we expect even more impact from our collaborative work with biologists and psychologists.

We are using our behavior recognition methods to investigate interaction rates in live ant colonies. In our own lab we are investigating the behavior of colonies of *Aphaenogaster cockerelli*. In work with Stephen Pratt at Princeton, we are exploring the means by which *Leptothorax albipennis* selects a new nest site. Evidence suggests that the colony comes to a consensus on which site to move to as the individuals experience a higher rate of encounters with each other in a desirable site [34]. Our work is focused on discovering the parameters of this decision process. Here we report on our experiments with *Aphaenogaster cockerelli* because the work with *Leptothorax albipennis* is still underway.

We would like to ascertain whether ants deliberately seek or avoid encounters with other ants, and whether they prefer certain types of encounters over others. From casual observation it seems that the ants are more effective at exchanging information when they meet head-to-head. It also seems that when a head-to-head encounter is feasible, the ants approach one another along a trajectory to ensure a successful meeting. We are interested in discovering whether the ants make these choices deliberately or whether their actions are essentially random. The importance of encounter rates has been explored by others, especially Deborah Gordon [21, 19, 20], but to our knowledge no one has examined whether certain types of encounters are preferred over others. In order to explore this

	Encounters per ant		
Encounter type	10 ants	20 ants	Change
head-to-head	12.6	24.3	+93%
head-to-body	36.4	48.4	+33%
body-to-body	50.0	62.8	+26%
body-to-head	39.2	48.2	+23%
total	138.2	183.7	+33%

Table 1: Automatically detected encounters for 10 and 20 ant experiments.

question, we consider the following null hypothesis:

> **Null Hypothesis:** The rate and type of encounters between ants arises from random motion of the animals.

If our data contradict this hypothesis, the assertion that ants move deliberately is supported.

For the experiments, foragers (ants outside the brood chambers, on or near food), were selected randomly with forceps and placed in an otherwise empty 10 cm by 15 cm arena with vertical walls. The walls were treated with fluon to prevent escape. Two 30 minute episodes were recorded, the first with 10 ants, the second with 20 ants. Both sets of ants were kept separated from the rest of the colony until the experiment was completed. In both trials the ants moved actively around the arena during the entire 30 minutes with no apparent degradation in their enthusiasm for exploration. Additional details regarding the animals and their habitats are included in the Appendix.

The system counted the total number of each type of encounter for trials with 10 and 20 ants. There were a total of 5,055 encounters detected: 1,382 encounters with 10 ants and 3,673 with 20 ants. The average number of encounters *per ant* is reported in Table 1. Observe that as the density of ants doubles from 10 to 20 ants the total number of interactions increases by only 33%. Gordon asserts that if the ants moved according to Brownian motion, we should expect the number of interactions to double. Head-to-body, body-to-body and body-to-head encounters each increase by 20-30%. However, the average number of head-to-head encounters nearly doubles, growing from 12.6 to 24.3 encounters per ant. All of these proportional increases are statistically significant.

Observe the following regarding these results:

- Although the density of ants increases by 100% from the first experiment to the second the total number of encounters increases by only one third.
- Of the four various types of interactions, head-to-head interactions make up a markedly small proportion (9% for 10 ants and 13% for 20 ants).
- In sharp contrast to the other types of interaction, the number of head-to-head interactions *nearly doubles* as the density of ants doubles from the first experiment to the second.

The first result confirms earlier work by Deborah Gordon in which she explored a number of variations affecting ant density, and similarly observed that the number of interactions did not increase in proportion to the density of animals in the arena [21]. The result that head-to-head interactions make up a relatively small proportion of the interactions raises more questions, namely: Does this represent a deliberate choice on the part of the ants, or is it merely a consequence of geometry and random motion? And why does the number of head-to-head interactions *double* when the other types only increase by about one third?

We created a simulation of ant movement to explore these questions. In particular we wanted to examine the assertion that we would observe the same distribution of encounters for agents that just move randomly. Our objective for the simulation was to create ant-like (but *random*) trajectories, where the simulated ants move without regard to the location and orientation of other ants. The

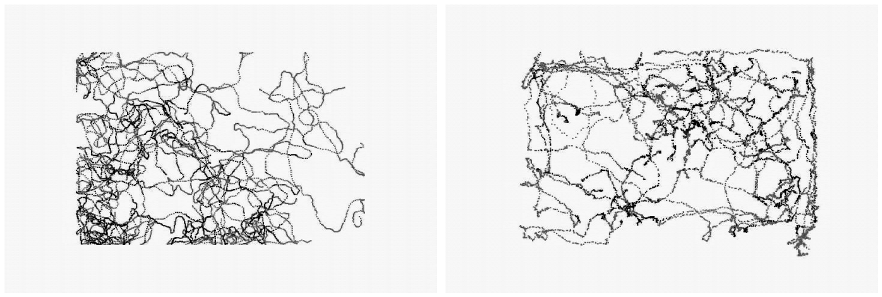

Figure 12: Simulated ant trajectories (left) compared with real ant trajectories (right).

simulated ants do not turn towards each other to initiate an interaction, or turn away in avoidance. In all other respects the simulation mimics the movement of real ants, based on statistics gathered from observation. Figure 12 illustrates the the types of trajectories generated by the simulation in comparison to those created by live ants. (Additional details of the simulation are provided in the appendix.)

	Encounters per ant in simulation		
Encounter type	10 ants	20 ants	Difference
head-to-head	3.1	4.8	+55%
head-to-body	36.0	62.4	+73%
body-to-body	46.7	76.2	+63%
body-to-head	36.9	62.7	+70%
total	122.7	206.1	+68%

Table 2: Quantitative results from simulation for 10 and 20 ants.

	Encounters per ant		
Encounter type	10 ants (real)	10 ants (simulation)	Difference
head-to-head	12.6	3.1	-75%
head-to-body	36.4	36.0	-1%
body-to-body	50.0	46.7	-7%
body-to-head	39.2	36.9	-6%
total	138.2	122.7	-11%

Table 3: Comparison of simulation experiments with animal experiments. Automatically detected encounters for 10 ants.

We ran 10 simulation experiments for both 10 and 20 ant cases. We then ran the same analysis on this data as we ran on the real ant data. Quantitative results from the simulation experiments are reported in Table 2. We also compare the quantitative behavior of real ants with the behavior of simulated ants in Table 3.

Consider Table 3 first. This data shows how well the behavior of the real and simulated ants corresponds when 10 ants are in the arena. Overall there is only an 11% difference in the total number of interactions, but there is a 75% difference in the number of head-to-head interactions. This indicates that there is a substantial difference in the way real ants and "random" ants interact head-to-head. Now consider Table 2. For simulated ants, when the number of ants doubles from 10 to 20 we see a 68% increase in the total number of interactions. In the real ant experiments however, we see only a 33% increase in the number of interactions. We suggest that the data supports the following conclusions:

- Ants seek head-to-head encounters at statistically significant higher rate than if they moved randomly.
- As the number of ants increases, they deliberately avoid encounters to maintain a lower encounter rate than if they moved randomly.

This experiment illustrates how automatic behavior recognition can facilitate social animal studies. The results arise from a few minutes of video, yet they represent over 5000 interactions of four different types between closely interacting animals. Our use of computer vision-based tracking and automatic recognition of social interactions enabled a statistical analysis that would have been very difficult or impossible to conduct otherwise. We expect to report additional and more comprehensive results like this in the future.

4 Learning Executable Models of Behavior

What is the best model of behavior? It is our position that an *executable* model provides the most *complete* explanation of an agent's behavior. By "executable" we mean that the model can run in simulation or on a mobile robot. By "complete" we mean that all aspects of an agent's behavior are described: from sensing, to reasoning, to action.

Robotics researchers are well positioned to provide formalisms for expressing executable models because the programs used to control robots are in fact executable models. Furthermore, researchers who use behavior-based approaches to program their robots use representations that are closely related to those used by biologists to describe animal behavior.

Our objective is to create algorithms that can learn executable models of social animal behavior from observations of the social animals themselves. In this section we will propose a formalism for representing executable models, and support it with examples from biology and robotics. Later we will introduce algorithms for learning executable models, and illustrate their use.

4.1 Representing Executable Models of Behavior

Behavioral ecologists seek to explain complex behavior by developing and testing predicative models. There are a number of methods for modeling social animal behavior at the individual and colony level. Ethograms are among the most effective and frequently used approaches [37]. A sample ethogram of ant behavior is provided in Figure 13. The nodes of this diagram represent the behavioral acts of individual animals. The links between the nodes show how behaviors are sequenced. The frequency of observed transitions is also recorded and represented.

Computer scientists will recognize a similarity between the diagram in Figure 13 and a Markov Process (MP). The nodes representing behavioral acts in an ethogram correspond to states in an MP. Transitions between behaviors correspond to the probabilistic transitions of an MP. Researchers are already investigating methods for automatically learning Markov Models, including some who apply the approach to learning models of behavior [27]. A goal of this work is to employ a similar approach to the task of learning animal behavior models.

The Markov Process is also quite similar to methods for programming sequences of behavior for mobile robots in the behavior-based paradigm, and we will take advantage of that similarity. The central idea of behavior-based robotics is to closely couple a robot's sensors and actuators so as to avoid the trouble of maintaining a map of the world or deliberating over it. Rodney Brooks' subsumption architecture and Arkin's motor schemas are the best known examples [2, 9]. Both of these approaches were inspired by biological models.

In the behavior-based approach *primitive* behaviors express separate goals or constraints for a task. As an example, important behaviors for a navigational task would include primitive behaviors

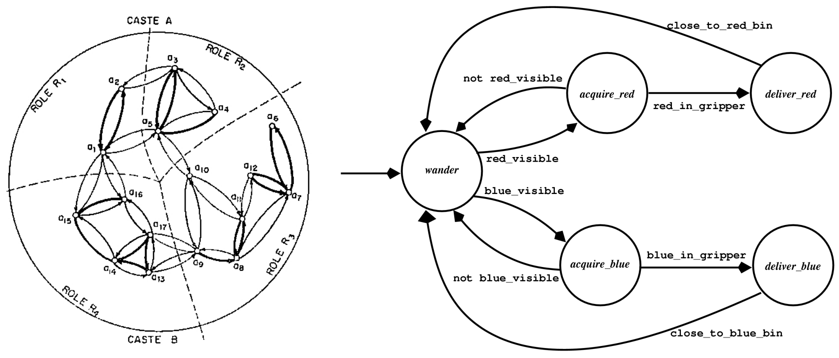

Figure 13: Representations of behavior in ants (left) and robots (right). Left: This representation of individual ant behavior (called an ethogram) is quite similar to a stochastic Markov Process. The Markov Process is one formulation used by AI researchers to model the behavior of intelligent agents. In this ethogram, behavioral acts, a_i, are linked by arcs indicating transitions from act to act. Thicker lines indicate higher probability transitions. From [Holldobler & Wilson 1990]. Right: This diagram illustrates a programmed sequence of behaviors for a foraging robot.

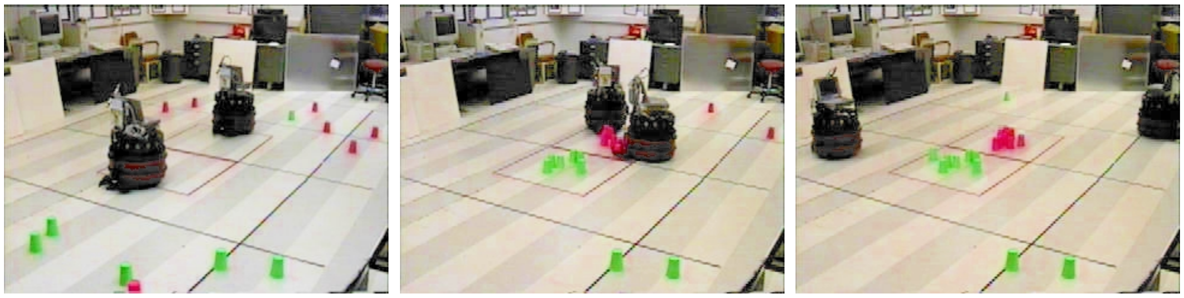

Figure 14: A team of two robots executes a behavior-based foraging program coded as an FSA.

like **avoid_obstacles** and **move_to_goal**. Behaviors may be grouped to form more complex, emergent behaviors. Groups of behaviors are referred to as *behavioral assemblages*. One way behavioral assemblages may be used in solving complex tasks is to develop an assemblage for each sub-task and to execute the assemblages in an appropriate sequence. The steps in the sequence are separate *behavioral states*. Perceptual events that cause transitions from one behavioral state to another are called *perceptual triggers*. A resulting task solving strategy can be represented as a Finite State Automaton (FSA). This technique is referred to as *temporal sequencing*.

An example FSA for executing a foraging task is illustrated on the right in Figure 13. A team of robots executing this program are illustrated in Figure 14. The robots step through a sequence of assemblages including a search of the environment (i.e. **wander**), acquisition of a detected food object, then delivery of the object to homebase. It is not necessary to understand the particulars of the behaviors at this point. What is important however, is the idea that transitions from one behavioral assemblage to another are triggered by perceptual events. As an example, the **deliver_blue** behavior, in which a blue item is carried by the robot back to the homebase (or "nest") is activated when the robot has a blue item in its gripper.

Note that these representations of sequenced behavior are strongly similar to the representations used by biologists to model behavior. This is at least partly because the approaches were inspired by biological mechanisms in the first place.

4.2 Learning Executable Models of Behavior from Observations

Consider the task for an ecologist seeking to describe the behavior of an ant or bee. Typically, in studying a particular behavior, a scientist will first define a set of primitive behaviors that can be observed and recorded. For example, in social insect foraging, such primitive actions include searching for and exploiting a food source [42]. Having defined these actions, the scientist must then determine the high-level structure of the behavior; that is, what causes the animal to switch between low-level behaviors. More formally we define our problem as:

- **Given:**
    1. A set of behavioral assemblages (i.e. controllers) an agent may use;
    2. A model of the agent's perception apparatus;
    3. A set of perceptions that may cause the agent to switch behaviors; and
    4. Example trajectories of the agent interacting in its environment.
- **Compute:** An executable model composed of the behavioral assemblages and the perceptual triggers that cause transitions between them.
- **Assume:**
    1. The agent acts according to a Markov Process with inputs.
    2. We have an accurate representation of the primitive behaviors the agent uses and the perceptions that cause transitions.

Stated another way: Assume we already know the constituent behaviors that reside at the nodes (states) of an FSA (like the one depicted on the right in Figure 13). The task then is to learn the transitions between the nodes by observing animals executing the behaviors.

Our assumptions may sound rather strong; how can we know the details of the behavioral assemblages that ants use? As it turns out, in related work we have developed a means for learning such low level controllers. The approach has been demonstrated by learning controllers from live ant data, and running them in simulation and on a mobile robot [16]. Our eventual goal is to combine these two approaches in order to learn the entire control system. At present, however, we will focus on learning the high-level switching, given the lower-level controllers.

This work is in contrast to work in *activity recognition* where learning is used to build models of activity. These models are often generative, but their goal is recognition, not re-creation. If domain knowledge is assumed, it is typically high-level domain knowledge of the structure of the activity being recognized, for example in the form of topology restrictions or high-level grammars. Such techniques have been used to classify bee behavior [17, 31] for example. Here we assume that we have knowledge of detectable low-level actions that we can use to understand the structure of a behavior sequence and reproduce the behavior in new situations.

Furthermore, we consider that it is important for the learned model to be understandable by humans – thereby making the learned models useful to biologists. In this way our work is similar to imitation learning. The notion of primitives in [8] is similar to our notion of behavioral assemblages.

4.2.1 The IOHMM Formalism

Hidden Markov Models (HMMs) [26, 6] represent the probability of an observation sequence $P(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T)$ where $\mathbf{y}_t$ is the observation vector for time t. Input/Output Hidden Markov Models (IOHMMs) [7] are a generalization of this model that represent the conditional probability of an observation sequence given an input sequence $P(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T | \mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_T)$ where $\mathbf{u}_t$ is the input vector for time t For each state $i = 1, ..., n$ in an HMM there is a transition distribution $P(x_t | x_{t-1} = i)$ and an output distribution $P(\mathbf{y}_t | x_t = i)$ where x_t is the discrete state at time t. In an IOHMM there is a conditional transition distribution $P(x_t | x_{t-1} = i, \mathbf{u}_t)$ and a conditional output distribution $P(\mathbf{y}_t | x_t = i, \mathbf{u}_t)$. The standard algorithm for training IOHMMs is an Expectation Maximization (EM) algorithm that is a straightforward extension of Baum-Welch for HMMs. After training, the model can be interactively executed on an input sequence in a manner similar to a finite state machine.

In our case, $\mathbf{u}_t$ is our state, including the position of the agent. The corresponding output $\mathbf{y}_t$ is the position of the agent after moving. We think of the inputs as corresponding to the agent's sensors and the output as the agent's low-level actions expressed in terms of observables. By using the model interactively in a simulator environment we can recreate the global behavior of the agent. We incorporate domain knowledge by modeling the output distributions as mixtures over the known low-level actions:

$$P(\mathbf{y}_t | x_t = i, \mathbf{u}_t) = \sum_j c_{i,j} P(\mathbf{y}_t | a_t = j, \mathbf{u}_t)$$

where a_t is the action at time t and $c_{i,j}$ is the mixing weight for state i and action j.

The actions are known, so only the mixing weights need to be estimated. Assuming we have calculated $P(\mathbf{y}_t | a_t = j, \mathbf{u}_t)$ for each time step and action—a task we discuss in the next section—the mixing weights can be re-estimated using the standard formulas for mixtures within HMMs [6]. These mixing weights provide a soft mapping between the states of the model and the known actions. The mapping is not necessarily one-to-one, admitting models with several states corresponding to the same low level action and with states that correspond to a mixture of multiple low-level actions.

Our variation of IOHMMs is less of a discriminant model than the standard IOHMM formulations because the actual input-output mappings are fixed. Our primary reason for selecting IOHMMs over, say, HMMs is that the conditional transition distributions allow us to more easily represent behaviors where switching between actions is triggered by sensors.

In our experiments we condition transition distributions on a set of binary sensory features (e.g. "bumped" or "see_target"), using the standard formulas for re-estimation [7]. We represent the transition distributions as simple look up tables (as opposed to the neural networks used in some IOHMM work). We also experimented with an encoding where we ranked the features and took as the input the index of highest ranking non-zero feature. The ranking method greatly reduces the number of parameters and allows for simple diagrams to be drawn of the resulting models, but choosing a valid feature ranking can be difficult and requires domain knowledge of the relative importance of the features.

4.2.2 Detecting Low-Level Actions

To estimate the mixing weights during EM we need $P(\mathbf{y}_t|a_t, \mathbf{u}_t)$ at each time step; this is equivalent to detecting which behavioral assemblage (or blend of them) is active at each step. Unfortunately motion is complicated by interactions with the environment, especially near walls and irregularly-shaped objects, so detection is not easy or unambiguous.

However, because we have executable versions of the behavioral assemblages, we can approximate the distribution through sampling, observing that sampling from $P(\mathbf{y}_t|a_t, \mathbf{u}_t)$ is equivalent to simulating a single time step of the agent's motion. Formally we estimate the probability values using kernel density estimation:

$$P(\mathbf{y}_t|a_t = j, \mathbf{u}_t) \approx \frac{1}{m}\sum_{k=1}^{m}\frac{1}{V_{Dh}}K(\frac{\mathbf{y}_t - \mathbf{y}_{tk}}{h})$$

where $\mathbf{y}_{tk}$ is sampled from $P(\mathbf{y}_t|a_t = j, \mathbf{u}_t)$, K is the kernel function, h is the bandwidth of the kernel, and V_{Dh} is the volume of the kernel function for the bandwidth and the dimension of the data, D. In our experiments we used a Gaussian kernel. At each step and for each behavioral assemblage we run a simulation many times to produce a distribution of points predicting the next position of the agent if that behavior were active. We then compare these point distributions to the actual next position. One advantage of this method is that it assumes nothing concerning the representation of the behavioral assemblages, so long as they can be executed in simulation. The method fully captures interactions with the environment to the extent that the simulation used can recreate them.

In experimental practice, this method of detecting which behavior is active provides 95% accuracy (detailed results are provided in the next subsection). Despite perfect knowledge of the behavioral assemblages, we still have errors because there are many situations where the outputs of several behaviors are indistinguishable. For example, when the agent bumps into a wall, most behaviors direct it to move away from the wall. In this case action probabilities become almost uniform, indicating that little can be said about the actions at such times. We expect situations like these to be typical in real world data.

4.3 Using IOHMMs to Learn an Executable Model of Foraging

We created an executable model of foraging inspired by the behavior of social insects, using the TeamBots simulation platform and motor schema-based control. The simulation includes 12 agents and 10 objects to collect. We ran the model in simulation for 5 minutes with 12 agents at 33 frames per second, waiting for all of the targets to be carried to base, recording at each frame, the position and orientation of all agents as well as the position of all targets. (Note that the output of the simulation log file is equivalent in format to the output of our vision-based tracking software.) Our objective is to learn a duplicate of the original model by examining the activities of the simulated agents recorded in the log file.

Figure 15 shows the state diagram for the original model and simulation screenshots. The agents are provided four behavioral assemblages to accomplish: loitering at the base (the center), exploring for targets, moving toward the closest target, and moving back to base. There are four binary features that trigger transitions: bumping into something, seeing a target, being near the base, and holding an object. If an agent is exploring and bumps into a loitering agent, that agent also begins to explore. Some transitions are also triggered randomly (*e.g.*, agents eventually return to the base if they do not find a target).

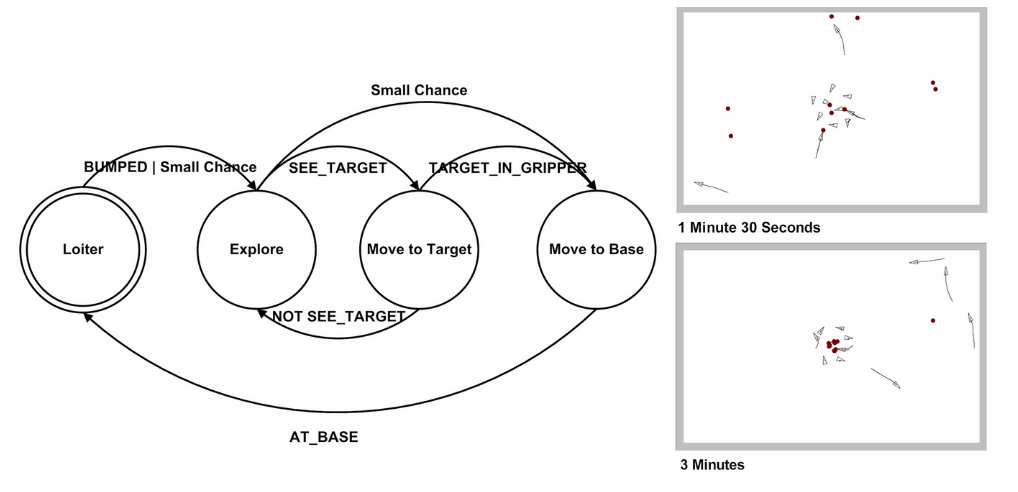

Figure 15: Left: The original foraging model. Right: Two frames from the running simulation that show the agents gathering food items and dropping them at homebase in the center of the arena.

We ran 100 trials using Expectation Maximization (EM) to learn 100 executable models (for both ranked and unranked input). Some of the learned models correspond well with the original models, but some do not. In order to distinguish poor models from good ones, we calculated the likelihood scores for all the models using the standard forward probability recursion values [7]. 14 ranked and 20 unranked input trials yielded models with likelihood scores higher than the original model's. We noted that all models that successfully learned appropriate mixing weights had high likelihood scores. The highest scores for the ranked and unranked input models were essentially identical to the original model. We executed the models with the lowest likelihood scores that were still greater than the likelihood score for the original model (in some sense the worst of the successful trials), and found these models were able to recreate the behavior of the original model. As seen in Figure 16, these models also recovered the structure of the original model.

There were flaws. In some models, bumping into an object while exploring would sometimes cause the agent to transition prematurely into the move-to-base state, slowing foraging. This appears to be a result of detection inaccuracies from bumping. Similarly, agents would drop a target early if they bumped into something while returning to base. Also if an agent was moving towards a target but another agent picked it up first, the first agent would loiter in place. One reason for these apparent failures is due to the fact that some of these situations did not occur in the training run.

Models with low likelihood scores did not perform well. Their behavior was not at all similar to the original model. The most common failure mode was the absence of an important behavioral assemblage as a state in the model (this is more or less equivalent to learning the wrong mixing weights). One example "bad" model is illustrated in Figure 17. Note that it includes two loiter states and it lacks a move-to-target state.

So far we have applied this method only to data from simulation (versus live ant data). One important reason for exploring simulation results first is that by using a simulation with agents that we program, we are able to directly compare the learned models with the ground-truth original. We will not be able to do this with ant data because it is impossible to know for sure how ants are really "programmed."

Nevertheless, we expect to apply this learning technique to the problem of learning models of real ant behavior in the near future. We will also augment the approach with low-level behavior learning so we can relax our assumption that the low-level behavioral assemblages are given.

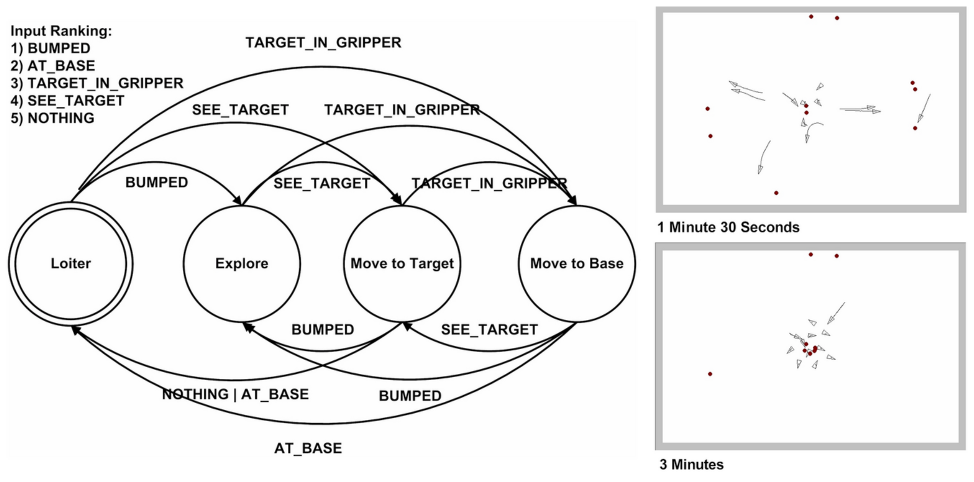

Figure 16: Left: A learned foraging model. Note that there are more transitions between the behavioral states than in the original model. These transitions, in concert with the ranked inputs result in functional equivalence between this learned model and the original. Right: Two frames from the running simulation that show the agents gathering food items and dropping them at homebase in the center of the arena. The behavior is nearly identical to the original.

5 Conclusion and Limitations

The central assertion of this paper is that robotics has much to offer social animal researchers. Many of the problems studied in robotics, and multi-robot systems research, have a dual in social animal research. We focused on three problems: multi-target tracking, automatic behavior recognition, and learning of executable models.

We have developed and demonstrated capable multi-target trackers, but we have not yet addressed a number of remaining challenges in this area. Ant tracking, for instance, is easier than bee tracking because the field of view is generally more sparse in the case of ants, and the background provides high contrast. In the case of bees however, the field of view is covered by bees, with very little background visible. And the background that is visible is the same color as the bees. We are developing new algorithms to address these problmes. We are also beginning to investigate new sensor modalities besides vision. One sensor in particular that we want to explore is the scanning laser range finder, and in particular, multiple distributed sensors.

With regard to behavior recognition, we have focused on recognizing social behavior – those activities that occur in the context of multiple animals interacting. Using geometric models of an animal's sensor system and labeled data, we are able to correctly label behavior with an accuracy of 90% to 95%. In some cases, however, our success is mitigated by a high false positive rate – we identify actions that did not take place. We believe these failures can be reduced using HMMs and similar techniques.

Finally, we showed how Input Output HMMs (IOHMMs) can be used to represent executable models of behavior. We also showed how standard Expectation Maximization techniques can be used to learn executable models from observational data. Our system successfully learned a model of foraging behavior that was run in simulation. The learned model also runs in simulation and it emulates the original very well. Currently the weakness in the approach lies in the fact that we assume we know the low-level behaviors already, and that we need only learn the higher-level sequencing. However, in other work, we have been able to learn low-level models, and we expect to integrate the two approaches in the future.

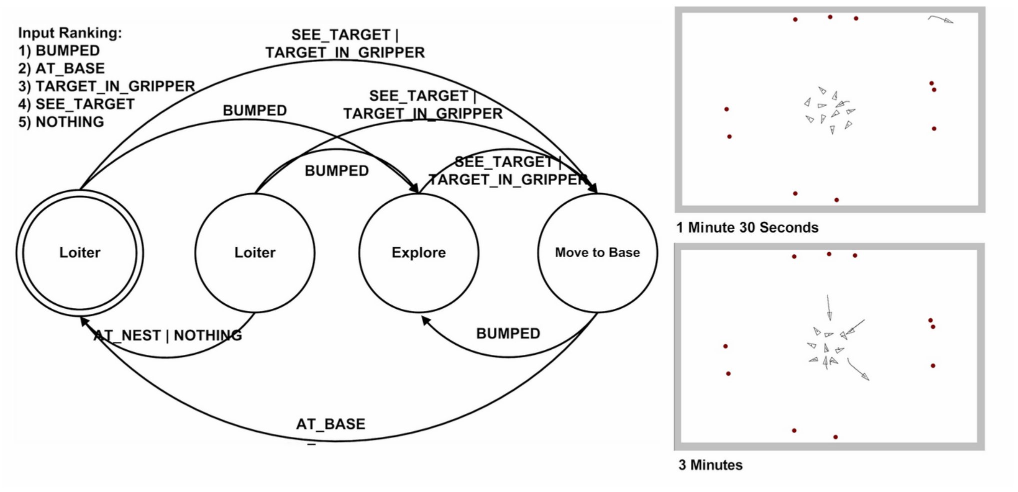

Figure 17: Left: A low-likelihood learned foraging model. This model includes two loiter states and lacks a move-to-target state. Right: Two frames from the running simulation that show that the behavior of agents running this model perform poorly.

6 Acknowledgments

The authors are grateful for the suggestions of Manuela Veloso, Deborah Gordon, Stephen Pratt, John Bartholdi, and Tom Seeley. This work was supported by NSF Award IIS-0219850.

7 Appendix

7.1 Details Regarding Ant Experiments

The colony was raised from a fertilized queen captured in September 2000 near Tucson, Arizona. At the time of experiments the colony was composed of about 400 workers and a queen. The colony did not include any alates of either sex. The colony is housed in two 10 cm square plastic brood chambers, each with a single 0.5 cm diameter entrance hole. The chambers are kept in a larger, 25 cm by 40 cm container in which the ants are free to roam. The brood chambers include floors made of hydrostone into which water is occasionally squirted through a tube to maintain humidity. The colony is fed a diet of fresh apple slices once per week and two or three crickets each day. Fresh water is always available from a test tube with a cotton stopper. Temperature was maintained at about 21 degrees C throughout the experiment. We have found greater success in terms of colony growth and alate production by keeping them at 27 degrees C.

For the experiments, foragers (ants outside the brood chambers, on or near food), were selected randomly with forceps and placed in an otherwise empty 10 cm by 15 cm arena with vertical walls. The walls were treated with fluon to prevent escape. Two 30 minute episodes were recorded, the first with 10 ants, the second with 20 ants. Both sets of ants were kept separated from the rest of the colony until the experiment was completed. In both trials the ants moved actively around the arena during the entire 30 minutes with no apparent degradation in their enthusiasm for exploration.

Video was captured by a JVC camcorder onto mini-DV tape. The camera was set to capture images at 30 Hz in a progressive scan (non-interlaced) mode. A 345 second (5 minute, 45 second) clip was selected at random from each 30 minute experiment. These clips were analyzed to produce the results reported below. The video segment used to validate the encounter detection software (outlined above) was from the same experiment, but from a different time period than was used to gather the quantitative data.

7.2 Details Regarding Simulated Ant Experiments

The simulated ants are initialized at random locations in a simulated arena the same size as the real ant experiments. Because our video data is acquired at 30Hz, the simulation also progresses in discrete steps of $\frac{1}{30}$th of a second. The simulated ants act according to the following simple "program:"

```
1. Set heading to a random value.
2. Set speed and move_duration to random values.
3. Set timer = 0.
4. While timer < move_duration do
    (a) Move ant in direction heading for distance $\frac{\texttt{speed}}{30}$.
    (b) If collision with wall, set heading directly away from the wall, then left or right by a
        random amount.
    (c) If collision with another ant, set heading directly away from other ant, then left or right
        by a random amount.
    (d) Increment timer by $\frac{1}{30}$ second.
5. Change heading by a random value.
6. Go to 2.
```

In order to make appropriate random selections for the variables `speed, heading` and `move_duration` we used data from the movements of the real ants. Ignoring encounters with other ants, we observed that the animals tended to move for some distance in a more or less straight line, then they would stop, occasionally pause, then move off in another slightly different direction. We calculated a mean and standard deviation for the time, speed and change in direction for each of these movement segments. These parameters were then used to draw random numbers for the variables according to a Gaussian distribution. For `speed` and `move_duration` negative values were clipped at zero. An example of trajectories generated by the simulation is illustrated in Figure 12.

References

[1] E. M. Arkin, H. Meijer, J. S. B. Mitchell, D. Rappaport, and S. Skiena. Decision trees for geometric models. *International Journal of Computational Geometry and Applications*, 8(3):343–364, 1998.

[2] R. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.

[3] T. Balch and A. Guillory. Teamview. `borg.cc.gatech.edu/software`.

[4] T. R. Balch, Z. Khan, and M. M. Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In *Agents*, pages 521–528, 2001.

[5] Y. Bar-Shalom, T. Fortmann, and M. Scheffe. Joint probabilistic data association for multiple targets in clutter. In *Proc. Conf. on Information Sciences and Systems*, 1980.

[6] Y. Bengio. Markovian models for sequential data, 1996.

[7] Y. Bengio and P. Frasconi. Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, September 1996.

[8] D. Bentivegna, A. Ude, C. Atkeson, and G. Cheng. Humanoid robot learning and game playiing using pc-based vision, 2002.

[9] R. Brooks. A robust layered control system for a mobile robot. *IEEE Jour. of Robotics and Auto.*, RA-2(1):14, 1986.

[10] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color vision for interactive robots. In *Proc. IROS-2000*. IROS, 2000.

[11] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[12] I. "Couzin and N. Franks. Self-organized lane formation and optimized traffic flow in army ants. *Proceedings of the Royal Society of London B*, 270:139–146, 2002.

[13] I. Cox and J. Leonard. Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66(2):311–344, April 1994.

[14] T. Darrell, I. A. Essa, and A. Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1236–1242, 1996.

[15] R. Deriche and O. Faugeras. Tracking line segments. *Image and Vision Computing*, 8:261–270, 1990.

[16] M. Egerstedt, T. Balch, F. Dellaert, F. Delmotte, and Z. Khan. What are the ants doing? vision-based tracking and reconstruction of control programs. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, April 2005.

[17] A. Feldman and T. Balch. Representing honey bee behavior for recognition using human trainable models. *Adaptive Behavior*, December 2004.

[18] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8, July 1983.

[19] D. Gordon. The organization of work in social insect colonies. *Nature*, 380:121–124, 1996.

[20] D. Gordon. *Ants at Work*. The Free Press, New York, 1999.

[21] D. Gordon, R. Paul, and K. Thorpe. What is the function of encounter patterns in ant colonies? *Animal Behaviour*, 45:1083–1100, 1993.

[22] K. Han and M. Veloso. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the 9th International Symposium of Robotics Research (ISSR'99)*, pages 199–204, October 1999. Snowbird, Utah.

[23] M. Huber and E. Durfee. Observational uncertainty in plan recognition among interactingrobots. In *Proceedings of the 1993 IJCAI Workshop on Dynamically Interacting robots*, pages 68–75, 1993.

[24] V. Isler, J. Spletzer, S. Khanna, and C. Taylor. Target tracking with distributed sensors: The focus of attention problem, 2003.

[25] Y. A. Ivanov and A. F. Bobick. Recognition of multi-agent interaction in video surveillance. In *ICCV*, pages 169–176, 1999.

[26] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA, 1997.

[27] N. Johnson, A. Galata, and D. Hogg. The acquisition and use of interaction behaviour models, 1998.

[28] Z. Khan, T. Balch, and F. Dellaert. Efficient particle filter-based tracking of multiple interacting targets using an mrf-based motion model. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, 2003.

[29] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Pattern Anal. Machine Intell.*, 2005. To appear.

[30] Z. Khan, R. A. Herman, K. Wallen, and T. Balch. A 3-d visual tracking system for the study of spatial navigation and memory in rhesus monkeys. *Behavior Research Methods, Instruments & Computers*, 2004. in press.

[31] S. M. Oh, J. M. R. an T. Balch, and F. Dellaert. Data-driven MCMC for learning and inference in switching linear dynamic systems. In *AAAI*, 2005.

[32] L. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3), 2002.

[33] M. Powers, R. Ravichandran, and T. Balch. Improving multi-robot multi-target tracking by communicating negative information. In *Proceedings of the Third International Multi-Robot Systems Workshop*, Washington, DC, March 2005.

[34] S. "Pratt. Quorum sensing by encounter rates in the ant temnothorax curvispinosus. *Behavioral Ecology*, 16:488–496, 2005.

[35] C. Rasmussen and G. Hager. Probabilistic data association methods for tracking complex visual objects. *PAMI*, 23(6):560–576, June 2001.

[36] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. on Automation and Control*, AC-24(6):84–90, December 1979.

[37] W. Schleidt, M. Yakalis, M. Donnelly, and J. McGarry. A proposal for a standard ethogram, exemplified by an ethogram of the bluebreasted quail. *Zietscrift fur Tierpsychologie*, 64, 1984.

[38] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2001.

[39] A. J. Smola and B. Schlkopf. On a kernel-based method for pattern recognition, regression, approximation, and operator inversion. *Algorithmica*, 22(1/2):211–231, 1998.

[40] T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[41] A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, 2001.

[42] D. "Sumpter and S. Pratt. A modeling framework for understanding social insect foraging. *Behavioral Ecology and Sociobiology*, 53:131–144, 2003.

[43] M. Tambe. Tracking dynamic team activity. In *National Conference on Artificial Intelligence(AAAI96)*, 1996.

[44] B. Webb. What does robotics offer animal behaviour. *Animal Behaviour*, 60:545–558, 2000.

[45] B. Webb. What does robotics offer animal behaviour. *Nature*, 417:359–363, 2002.