USING OBSERVATIONS TO RECOGNIZE THE BEHAVIOR OF INTERACTING MULTI-AGENT SYSTEMS

A Dissertation
Presented to
The Academic Faculty

by

Adam Feldman

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in Computer Science

Georgia Institute of Technology

August, 2008

USING OBSERVATIONS TO RECOGNIZE THE BEHAVIOR OF INTERACTING MULTI-AGENT SYSTEMS

Approved by:

Dr. Tucker Balch, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Irfan Essa
School of Interactive Computing
Georgia Institute of Technology

Dr. Charles Isbell
School of Interactive Computing
Georgia Institute of Technology

Dr. Thad Starner
School of Interactive Computing
Georgia Institute of Technology

Dr. Kim Wallen
Department of Psychology
Emory University

Date Approved: May 12, 2008

For my father and friend,
Stuart Feldman

ACKNOWLEDGEMENTS

The work presented herein would not have been possible without a great deal of assistance from others. I am equally indebted to those individuals who have helped me stay sane as I plotted my course through graduate school as I am to those who directly helped drive my research along these past six years.

As my advisor almost from day one, Tucker Balch was a great catalyst for the inspiration which led me to the eventual topics of my thesis. He makes graduate research fun by allowing his students to focus on their interests, while still providing ample prodding whenever necessary. One key lesson he teaches is a healthy balance between the theoretical and the practical. I know my graduate studies would have been a very different, and not likely improved, experience had I wandered into a different lab in my first year at Georgia Tech.

I have also received the benefit of four additional exceptional committee members. Irfan Essa always found time to sit and chat about my latest developments and prompt me to think on the implications of each. Charles Isbell helped me look at the bigger picture from a different point of view than I usually saw in my lab. Thad Starner and his wearable never let me forget anything I ever said, but he did help me find new and better ways of analyzing my data and interpreting my results. Kim Wallen helped me feel like this was all worthwhile, by showing me how my work might, in some small way, help other researchers become more productive.

I would also like to thank every member of my lab, past and present, a more talented bunch of people I have never met. Especially, thanks to Frank Dellaert who

provided great inspiration and guidance for many of the techniques I adapted to my needs. Zia Khan provided the video tracking which generated the data I used in much of my research. Without Andrew Guillory's Teamview program, I cannot even imagine the extra time analysis of my results would have taken. My first forays into laser-based tracking relied on support from Wesley Wilson, Jorge Saguier, Summer Adams, and Hai Nguyen. I cannot begin to count all the projects, big and small, which Dan Walker helped me engineer my way through.

My research and I are also indebted to several individuals outside my lab. Tom Collins helped me get my feet in dealing with the laser range finders (which often seemed as unintuitive to use as possible). Maria Hybinette helped prepare a number of paper submissions, even when her name was not attached. Amy Henry and Janice Hassett at Emory were always quick to provide insights about the primates in their care. Most of my results would not have been possible without all of the wonderful volunteers who participated in my experiments. I cannot say your names, but you know who you are.

Thanks also must go to my friends, both at Georgia Tech and abroad, for helping with my studies and life in general. Gillian Hayes, one of my first new friends in this strange land called Georgia, helped me through some rough times. I think Keith O'Hara attended all of my practice talks – and managed to provide useful feedback each and every time. Tracy Westeyn has been a constant sanity check for all of my crazy ideas and continues to be a great friend. My Yankee friends David Hill, Joshua Kershner, and Chris Naegele provided escapes from lab just frequently enough to keep me lucid.

Diana Wey has been my constant companion. She was constantly patient when my work made me boring or frustrated me until I was annoying to be around, yet always knew just when to provide a necessary distraction. Without her love and support this would have been a much less enjoyable road, indeed. Thank you for always being there for me.

Finally, I offer acknowledgement to my parents, without whom *I* would not have been possible. Their love and encouragement has made me the person I am today. Thanks to my mother, Judy Feldman, for being a proper Jewish mother – and all that entails. She has always been there when I needed her. Last but not least, thanks to my father, Stuart Feldman, for his unwavering support in all that I do. This accomplishment certainly would not have happened without him.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

SUMMARY

Behavioral research involves the study of the behaviors of one or more agents (often animals) in order to better understand the agents' thoughts and actions. Identifying subject movements and behaviors based upon those movements is a critical, time-consuming step in behavioral research. This task consists of using a pen and paper to note the observations, and is especially onerous in studies involving multiple, simultaneously interacting agents (such as ants in a colony or players on the field.

To successfully perform behavior analysis, three goals must be met. First, the agents of interest are observed, and their movements recorded. Second, each individual must be uniquely identified. Finally, behaviors must be identified and recognized. I explore a system that can uniquely identify and track agents, then use these tracks to automatically build behavioral models and recognize similar behaviors in the future.

I address the tracking and identification problems using a combination of laser range finders, active RFID sensors, and probabilistic models for real-time tracking. The laser range component adds environmental flexibility over vision based systems, while the RFID tags help disambiguate individual agents. The probabilistic models are important to target identification during the complex interactions with other agents of similar appearance.

In addition to tracking, I present work on automatic methods for generating behavioral models based on supervised learning techniques using the agents' tracked data. These models can be used to classify new tracked data and identify the behavior exhibited by the agent, which can then be used to help automate behavior analysis.

CHAPTER 1

INTRODUCTION

Imagine being given a pencil and paper sketch of the locations of all the people currently on a basketball court. What can be deduced from this simple information? The number of people tells if a game was being played, if there was a timeout in progress, or if the halftime show was commencing. The location of each individual provides even more information. It would be obvious if the ball was in play or if someone was about to throw it in from out of bounds. No one would have trouble knowing which team, if either, was about to shoot a free throw.

Now imagine having a series of these sketches, each showing the same court, but indicating the positions of the players over time. More information could be gleaned as more sketches were revealed: which players were on the same team, which team was on defense, and perhaps even which player possessed the ball. Finally, by examining a number of these sketch series (or tracks), an astute person could learn to recognize behaviors that are frequently carried out; for example, a given play which is repeatedly executed.

Just as a person can deduce a wealth of data from this simple information, so, too, could a computer algorithm. The automatic detection of such information could be used in a number of applications which are not conducive to human labeling. This will save time and/or create entirely new capabilities which are beyond the means of human labelers. In addition to sports, areas that could benefit from this automatically identified data include robotics, biology, and even security.

1.1 Research Questions

One area of artificial intelligence that has garnered much interest is behavior recognition. That is, observing one or more agents and determining in which behavior each is engaged from the trajectories of their movements. For example, such recognition would allow for the identification of bees performing a waggle dance in an observation hive or suspicious human activity in a subway terminal. One subset of behavior recognition concerns itself specifically with the identification of behaviors which involve multiple agents interacting, for example players (human or robotic) in a team sport or ants encountering one another in an arena. The research question asked by this work could be expressed as:

- How can observations of a multi-agent system be used to model and recognize the behavior of that system's interacting agents?

In order to explore this question, several sub-questions must be addressed. Four in particular guide the research presented herein. They are:

1. How can multiple sensor observations be used to generate tracks of multiple agents' positions as they enter, move through, and exit the environment?

2. How can the observations of multiple sensor types be combined to provide more accurate identified tracks?

3. How can this information then be used to create models of the interacting behaviors of the individual agents?

4. How can these models be used to recognize and/or predict the agents' true behaviors?

Each of these questions focuses on one of the steps necessary to accomplish the aforementioned behavior recognition. First, one or more types of sensors must be used to collect information about the agents being studied. This information is used to generate tracks of each agent as it moves through the environment (Question #1). Each track must then be associated with the specific agent that it represents (Question #2). Finally, models can be created based on the behaviors evident in the tracked agents' movements (Question #3) – these models will then be available to recognize the subsequent behaviors of the tracked agents (Question #4).

1.1.1 Question 1: Tracking

Tracking is accomplished using a number of laser range finders. Each of these laser range finders scans, in half degree increments, an arc of 180 degrees, out to a range of up to 80 meters. By placing several scanners around the edges of the environment of interest (say, a basketball court or football field), pointed inwards, the entire area is covered from multiple viewpoints. This is important because occlusion is a major restriction of laser range finders. This means that a scanner cannot "see" agents which are blocked by other agents or stationary objects. However, by using multiple sensors, the effects of occlusion are greatly reduced.

Tracking will rely solely on the laser-based data. However, RFID measurements will later be used to match up each track with the unique agent represented by that track. Therefore, each track must represent exactly one agent, from start to finish. Yet, because the active RFID tags only send a signal periodically, it is important that the tracks generated by the laser data be as long as possible, while still maintaining confidence that a single track represents a single agent. All tracks made by a given agent can then be combined to form a single track of that agent's activity over the length of the experiment.

The tracker described in this dissertation functions by matching instances of one or more models (or templates) to detection-based data (such as from laser range finders), using iterative closest point (ICP) to determine the best location of each track. Implementation of models allows the tracker to better differentiate near-collisions, as well as being able to track agents of multiple sizes and shapes. Providing information as to the type of agent of each track will assist in the track/agent association discussed below.

1.1.2 Question 2: Track Association

The laser scanners used for tracking provide incredibly accurate measurements, but do not have the ability to distinguish between agents. That is, one agent looks very much like any other in the laser data. Therefore, to uniquely identify each agent over time, another sensor must be used. Active radio frequency identification (RFID) has been selected. The benefit of this sensor is that the tags, each placed on an agent, provide perfectly unique identification (in the form of a serial number which is repeatedly broadcast). Unfortunately, localization of the tags is very rough, preventing this sensor from being used alone. Thus, a mechanism for fusing the RFID data with data from the laser scanners is developed.

Because the laser data is so accurate, it alone will be used to generate tracks. The tracks, each representing a single agent, are labeled based on the probability that their locations correspond to the RFID readings. The probabilities are generated from building a histogram model from a set of training data, and an error minimization algorithm is used to find the best set of track/label pairings. The result is a series of tracks, each of which represents a single, specific agent from start to finish.

1.1.3 Question 3: Behavior Modeling

Given a set of uniquely identified tracks, certain behavioral information can be determined about the agents. For example, in the context of a colony of non-human primates, various interactions are detected, both affiliative and aggressive. From the pattern of these interactions, the familial relationships between the individuals of the colony can be learned. More interestingly, the hierarchical relationships of the families are then deciphered. In the domain of social insects, the informative waggle dance of honey bees can be recognized. The first step in performing this recognition is to create models of these behaviors.

1.1.4 Question 4: Behavior Recognition

Once the behavioral models have been created, they must be used to recognize subsequent occurrences of the behavior which they model. Two approaches form the basis of this work. First, the threshold technique is used to recognize interactions which, for some set of features, can be identified based on the value of those features. Detection of other behaviors will benefit from the time element of the HMM technique.

1.2 Example Domains and Motivation

There are many domains that can benefit from automated tracking and behavior recognition. The research in this dissertation has been applied to several biological domains. For instance, the behavior recognition techniques have been used to detect waggle dances in a honey bee observation hive and count and classify types of interactions between multiple ants in a potential nest. The tracking algorithms are also applied to help study primate behavior. More generally, any biological system that is currently observed by humans is a potential target for an automated tracking and behavior recognition algorithm. This would reduce or eliminate the very arduous task of manual observation and data labeling.

These algorithms could also be of benefit when applied to robotic agents. It is sometimes useful to be able to confirm the behavior that a robot is executing, such as while testing a new program. By creating a model of the robot's observed activity and comparing it to its desired behavior, the performance of the controller can be determined. On the other hand, when dealing with an unknown or enemy robot, the creation of a behavioral model would provide insights into its objectives. This would be useful in applications ranging from security to robot soccer.

The impact of automatic tracking and behavior recognition on each of these domains is further discussed in Chapter 2. Yet, while this research is applicable to many domains, the effectiveness of the techniques to answer each of the research questions in two specific areas is examined. One of these domains, a team sport setting, involves tracking the players of an amateur basketball game. These activities take place in a constrained, yet realistic, environment with multiple agents constantly engaging in social interactions. Such a setting is challenging, but not overwhelming, and provides an outlet for theoretical and practical considerations, as well as many potential uses, including augmented broadcasts, team training, and video game design. The other situation in which this research is tested is an experiment which uses human volunteers to emulate a setting consisting of several dozen small primates. The participants are free to move about their arena and interact in a variety of ways, mimicking true monkey interactions. The problems posed in this domain are slightly different, as the individuals move around less quickly than sports players, yet there are more of them and they tend to cluster in larger groups and for longer periods than the basketball players.

1.3 Preview of Contributions

The results of this research are applicable to several different domains, providing contributions to a number of groups, including robotics, biology and machine learning. Specifically, it is the goal of this research to answer the research questions by providing:

1. A method of tracking multiple, changing numbers of interacting agents using data from multiple laser range scanners. This tracker, which tracks agents moving in a single plane, must maximize the length of each track while simultaneously ensuring that each track represents no more than a single agent. Success is measured in its ability to generate such tracks which are long enough to allow unique agent/track association to be performed.
2. RFID tag usage techniques to associate each laser-data generated track with the agent that it represents. Only after performing this association will the tracks be used to model and recognize behaviors. Accuracy is measured by the percent of tracks correctly identified.

3. Models used to represent and recognize choice social behaviors in which agents might be engaged. The examined behaviors are those which can be described in the trajectories of the agents. These models are evaluated on their ability to recognize the behaviors present in the test data set.
4. Comparison of experimental results of applying these algorithms in several real-world biological domains. By building a system, from start to finish, which performs the above sub-tasks, the primary research question is answered.
5. Benchmarks of the ability of the tracking components of the system to function in real-time applications.

Although many aspects of this research have been well studied individually, work in the proposed combination of technologies is limited. Major contributions of this work include the use of multiple laser range finders to generate tracking data. Using active RFID to associate pre-created tracks with the agents that they represent is a novel approach. Furthermore, tracking applications in the sports target domain have focused on using computer vision; the use of laser range finders as the primary tracking sensor is a new alternative to existing systems. The ability to function in real time enhances this research's impact. Finally, much of the behavior recognition literature concerns itself with the activities of isolated individuals. On the contrary, the activity of interest for this research consists of behaviors representing multiple, interacting agents.

CHAPTER 2

BACKGROUND AND RELATED WORK

This research focuses on tracking multiple interacting agents (or “targets”, as they are referred to in the sphere of tracking) while they move through the environment, and then modeling and recognizing their interactions and behaviors from these tracks. Many techniques exist for accomplishing these goals, incorporating a variety of sensors. While these are far from being solved problems, some combinations have been well studied. This chapter discusses the core research in tracking and behavior modeling, closing with an examination of some of the domains in which this research has been applied. The goal is to illustrate the foundation upon which this research is built, as well as differentiate it from similar work.

2.1 Tracking

Tracking consists of determining the location of the targets present in each “frame” of data. A frame is made up of all the data from a single point in time. With video data, this corresponds to a single frame of video. For laser-based data, a frame would be the measurements from one simultaneous scan from each of the sensors. Once the laser scans are transformed into the same coordinate system, a single “picture” of the scene at that time is created, like a video frame. This frame, then, can be examined for the targets of interest.

One approach to tracking is to divide the task into two parts; finding the “interesting” objects (targets) and then tracking a given target over time (the data association problem). Obviously, the mechanism for finding the targets is based somewhat on the nature of the sensors being used. Two popular sensor types are video cameras and laser-range finders. Techniques for finding the targets in each of these types of data are discussed in Sections 2.2 and 2.3. Once any targets are found, the process is sufficiently disjointed from the sensors that many methods no longer vary based on the input. Therefore, Section 2.4 will close with a treatment of several popular ways to perform this data association.

2.2 Computer Vision

One common method of obtaining tracking data is from video. Computer vision attempts to identify the targets in a video sequence, separating them from the uninteresting background. This is very challenging, as a number of factors can adversely affect performance. Occlusion is a major problem, making camera placement very important. Vision is also susceptible to changing lighting conditions, including changes from sunny to cloudy in an outdoor environment. Nonetheless, there are ways to deal with each of these issues.

2.2.1 Finding Tracks

The first step in many tracking algorithms (both in vision and non-vision tracking) involves actually locating the "interesting" targets in the data. One approach to finding these targets is tracking by color. In this method, efficiently accomplished by Bruce et al [2000], each pixel is examined for inclusion in any of a fixed number of color classifications, based on simple thresholding. Once every pixel is classified, connected and near-connected regions of the same color are "grown," then sorted by size. This results in a list containing all occurrences of each color classification. In some cases, this is sufficient to determine the location of each target, such as small scale robot soccer players, in which the rules stipulate very specific colorations to all elements of the game [Han & Veloso 1998].

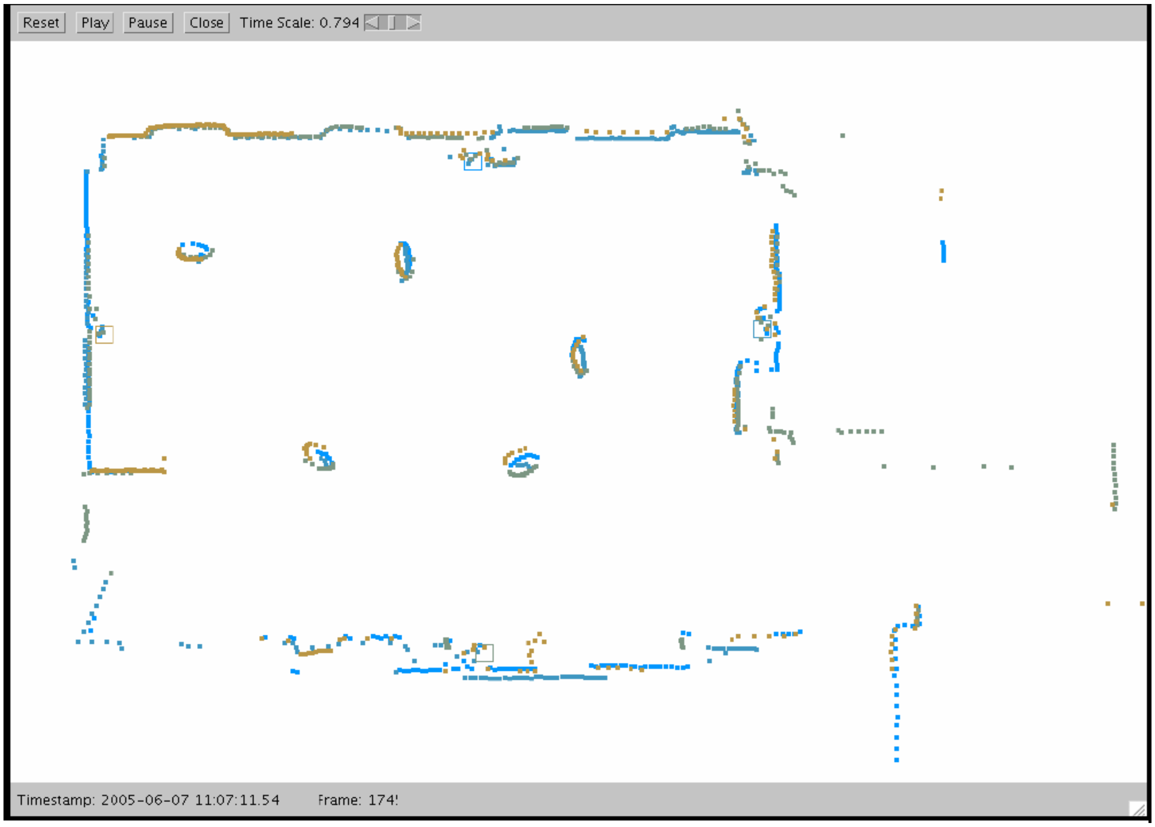
Unfortunately, such color segmentation is often not good enough; even in controlled laboratory environments, shadows and changing lighting conditions can render tracking by color techniques insufficient. An alternative to this color-based tracking is tracking by movement. At the most basic level, this consists of frame differencing, or comparing pixels in the current video frame with those of the previous frame. If a pixel has changed amply, usually with regard to intensity, it is assumed that this pixel demonstrates movement in this frame. Though effective, this method suffers from the main problems of only detecting the "wavefront" of moving objects and not detecting slow movement [Rosin & Ellis 1995].

To combat these issues, adaptive background subtraction can be used. In this approach, a representative image of the background (without any moving objects) is subtracted from the current image; what remains indicates movement. However, this is computationally intensive. Balch et al [2001] overcome this hurdle by creating a hybrid system. In this system, designed to track ants in a (mostly) white arena, color segmentation is initially done to locate areas of the frame which are "ant colored." This quick operation determines potential locations of targets. Then, these specific locations are further examined using background subtraction for indications of movement. Any location of sufficient area which demonstrates movement is considered an ant.

Another approach is to segment the data based on models of the targets. For instance, Zhao & Nevatia [2003] use human shape models to interpret the foreground in a Bayesian framework. Mittal & Davis [2003] model the characteristics of people by observing them over time. These characteristics include color models at different heights of the person. In both cases, the models are used to segment the images, resulting in the detection of all targets. Further, the latter uses occlusion analysis to allow probabilistic tracking through moments when the targets are not (completely) visible.

2.3 Laser-based Tracking

Laser-based tracking is developing as an alternative to video-based computer vision. Recently, laser range finders ("ladar") have been used in a variety of applications, including localization on mobile robots [Dellaert et al 1999], tracking in crowded environments [Prassler et al 1999], and map building [Gonzalez et al 1994]. In some respects, lasers are ideal compared to video. They are more reliable because they are less susceptible to "false positives" and "false negatives." In other words, a detected object (laser hit) almost certainly corresponds to an actual object in the world, while the lack of a hit reliably indicates that there is no corresponding object in the world. Further, laser range finders have very high spatial accuracy; the laser hit corresponds to the object's actual location, within 1.5 cm (according to the manufacturer). They are not sensitive to noise such as changing light conditions, and they have significant range [Fod et al 2002]. Yet, lasers offer some of the same challenges as computer vision, including extracting the targets from the cluttered background, differentiating one target from another, and an inability to handle occlusions. Even worse, their usage poses additional complications: lasers provide no way to uniquely identify a target (all targets "look alike" to the laser), and they generally have a field of view limited to a single plane. An example data frame from laser range finders is shown in Figure 2.1.



Timestamp: 2005-06-07 11:07:11.54 Frame: 174!

Figure 2.1: One frame of laser data from 4 laser range finders in a lab. The 5 visible ovals were caused by people walking in the lab. From [Balch et al 2005].

2.3.1 Laser Range Finders

Although some commercial laser tracking solutions exist, they are expensive and impose restrictions on the movement of the targets [Fod et al 2002]. Thus, it is more common for researchers to put together their own system, often using members of the SICK brand line of lasers, such as the LMS-291 (Figure 2.2). The basis of this sensor is a near-infrared laser pointed downward at a 45-degree angled mirror. The mirror rotates rapidly, sending the laser out in a planar sweep parallel to the ground. This sweeping arc can be up to 180 degrees, and generates measurements in one quarter, one half, or one degree increments. The maximum range is either 8 meters (at millimeter resolution) or 80 meters (at centimeter resolution). When the laser beam strikes the nearest object at

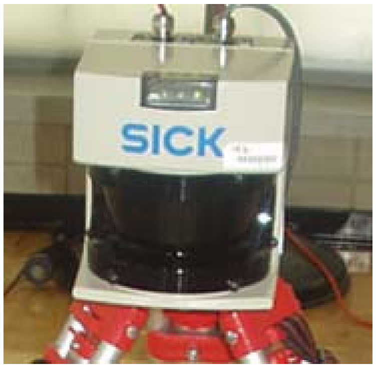


Figure 2.2: A SICK LMS-291 laser range finder.

each angle, the beam bounces off the object and returns to the sensor. The distance traveled is automatically calculated from the time of flight of the beam. If the beam is not reflected by an object within range, then a "no reading" value is returned. A full study of the LMS-200 was conducted by Ye & Borenstein [2002].

Multiple ladar can be placed around the environment to reduce the effects of occlusion and increase the system's field of view. For example, Jung & Sukhatme [2002] use multiple sensors (on robots) to track multiple targets through environments with varying occlusion characteristics. However, that work does not explicitly fuse the multiple sensor data. Much like Stroupe & Balch [2003] use data from multiple range-bearing sensors to improve accuracy, Fod et al [2002] combine data from several ladar to track people in an office environment.

As previously mentioned, the field of view of a ladar is a single plane. This is often sufficient for tracking objectives, such as robots (which do not change height) and people in an office (placing the laser at about a meter high will catch standing people at waist level and sitting people at chest or shoulder level) [Yan & Matarić 2002]. However, sometimes, one plane simply cannot capture all of the necessary data. In some of these cases, multiple lasers could be used to generate data at multiple, parallel planes. Alternatively, Kornienko & Kleeman [2007] use vertical laser scans to track human body-parts in 3-D. A more complete solution is to mount a ladar on a servomotor so that it can rotate in a direction perpendicular to the plane of the laser scans. This would create a 3-D "image" of the scene [Surmann et al 2001].

2.3.2 Finding Targets

In some sense, working with a ladar generated frame is easier than a video frame. Every laser hit directly and accurately corresponds to the location of a physical object. This makes certain tasks simpler. For example, by comparing a scan to a known map, localization is very straightforward, even in a symmetric environment [Gutmann et al 2000]. In that work, a scan is compared to a model of the soccer field (including only lines significantly longer than would be generated by another robot); with just three visible walls, the robot could be localized to one of two places. Combining this with odometry and passing it to a Kalman filter (introduced in Section 2.4.1) can consistently and uniquely identify the robot's exact position, all without having to perform complicated analysis on a video frame. However, there are still challenges in using lasers for tracking.

Background Subtraction

One important difficulty is in determining which laser hits correspond to targets to be tracked (the foreground) and which correspond to background structures (e.g. walls, desks, and other "uninteresting" obstacles). This process, which serves the same purpose as background subtraction in computer vision, can be accomplished in several different ways. In the regulated, soccer-field environment of Gutmann et al [2000], once localization is accomplished using the long scan lines of the walls, they are removed and the remaining points make up the foreground.

In less constrained environments, it is not always feasible to have an a priori model of the background. For example, Schulz et al [2003a] computes a probability grid for each local minimum in the distance histogram. This is then compared to the data from the previous frame to determine the probability that something has moved to that location, eliminating close, but static, objects from being considered.

Another approach [Fod et al 2002] derives inspiration from the computer vision background subtraction problem [Toyama et al 1999]. Instead of examining each pixel, though, each angle of the laser's scan arc is considered individually. For each scan, they assume that background is made up of the farthest known stationary object. Therefore, any measurements with a distance less than that of the current background will be treated as part of a foreground target, while farther objects are treated as background and added as part of a new background (maintaining the mean and variance to provide robustness to noise).

Of course, sometimes it is desirable to *not* subtract the background. When the sensor is mobile, then the background is important for obstacle avoidance algorithms. For example, Prassler et al [1999] describe a tracking system for use on an automated wheelchair. Although this system does not subtract the background, it does distinguish between mobile and stationary targets for the purposes of motion modeling.

Determining Structure

Once the majority of the background laser hits have been removed, it is necessary to determine the structure of the targets, usually by joining the remaining laser hits into clusters (sometimes referred to as "blobs"), each representing a single target. This is often difficult, as two or more targets in close proximity may appear to be a single target. Or, one entity may appear to be two separate targets. Obviously, this causes great difficulties in data association, which relies on being able to identify each target being tracked.

Several methods of resolving this phenomenon have been developed. The approach used by the previously mentioned Gutmann et al [2000] system is to cluster the points and denote a target as existing at the center of gravity for each cluster. Prassler et al [1999] also grouped nearby hits into distinct objects. This can be potentially problematic depending on the shape and relative sizes of the targets.

Another technique involves creating blobs, each composed of a continuous surface, and then joining blobs together to form the desired target [Fod et al 2002]. Laser hits are deemed part of a continuous surface (and thus a single blob) if they are within 10 centimeters of each other. Multiple blobs representing a single object (such as a person's torso and arms) are unified into a single target for data association. Complications could arise when dealing with large ranges. For instance, at 30 meters, laser scans striking the same surface will be over 25 centimeters apart, yet increasing the "surface threshold" could cause disjoint surfaces to be viewed as continuous, especially when the targets are frequently very closely interacting, such as in sporting activities.

2.4 Data Association

Regardless of what sensors provide the original data, once the targets are located in a frame, it is necessary to associate each target with itself in previous frames, the so-called data association problem. A simple method of accomplishing this is with greedy association [Veloso et al 1998]. Such an algorithm won the 1997 RoboCup small-robot competition. The idea is that current targets are matched to the closest target in the previous frame. Each current target/previous location pairing is examined for the pairing with the lowest distance. This pairing is made, removing the target from the current and previous frames' lists. The algorithm iterates until all targets are matched. A similar strategy is employed by Prassler et al [1999] on laser data. The only difference with their "nearest neighbor criterion" is that they set a maximum distance threshold, beyond which objects would be declared separate targets.

Although this algorithm works well in certain situations, it is theoretically not guaranteed to find optimal associations. Figure 2.3 demonstrates an example of this sub-optimal performance. In this circumstance, the greedy algorithm will incorrectly assume that Target 2 barely moved, while Target 1 moved quite far to the right.

An improved algorithm was developed [Han & Veloso 1998] to handle such a situation. In this algorithm, dubbed globally optimal association, all possible sets of

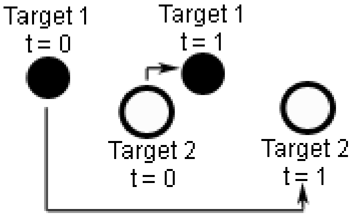


Figure 2.3: One situation in which the greedy algorithm fails.

matching are generated. For each set, the fitness is calculated as the sum of squared distance between each pairing. The set which minimizes this criterion is selected. Although still not theoretically optimal, this algorithm has been shown in a number of implementations to be quite robust ([Han & Veloso 1998] and [Balch et al 2001]). Unfortunately, with a large number of targets, this technique can slow down the tracking process.

2.4.1 Probabilistic Techniques

Color-based tracking is appropriate in many circumstances. However, there are many times it does not work well. For example, to track bees, it requires marking all of the animals to be observed, as in Figure 2.4. This is difficult and does not guarantee success; the tracker often gets confused when two animals interact. To solve these problems, a probabilistic framework can be employed. Two popular methods have been the Kalman filter and the particle filter. Each is discussed, along with some extensions.

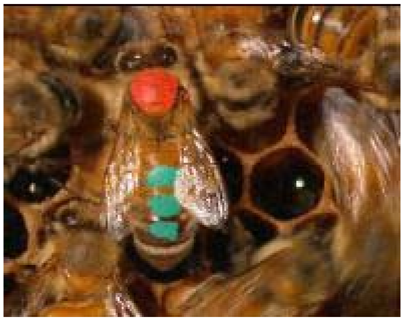


Figure 2.4: A painted honeybee in the hive. From [Balch et al 2005].

Kalman Filter

One method of probabilistically tracking a target using noisy data (such as in computer vision, but also applicable to any noisy sensor) is the use of a Kalman filter.

Originally introduced in 1960 by R. E. Kalman, the filter recursively maintains a current state estimate (of, say, the target being tracked) by performing a time update/measurement update cycle. The filter estimates the state for some time, and then receives a measurement which is used to update that estimate. Formally, the time update phase projects forward (temporally) to get the a priori estimates for the next time step. Next, the measurement provides feedback, adjusting the a priori estimate to create a better a posteriori estimate [Welch & Bishop 2004].

Kalman filters are useful in some situations, for example, to estimate depth from image sequences [Matthies et al 1989] and to lessen the affects of occlusions and sensor noise (with laser range finders) [Fod et al 2002], but they have some serious limitations. First, the basic Kalman filter is limited to a linear assumption. That is, the process model and observation model must both be linear functions. Yet, this assumption does not hold for many non-trivial systems. Therefore, the extended Kalman filter (or EKF) has been developed. This filter linearizes the estimation using the partial derivatives of the process and observation functions to calculate linear estimates, despite the non-linear relationships. Rosales & Sclaroff [1998] use an EKF with an occlusion detector built on top to improve tracking, though this system makes some assumptions on the characteristics of the shape of the targets.

Another limitation of the Kalman filter is its inability to maintain a multimodal representation. For example, if a target is likely to be in one of two non-connected areas, a Kalman filter will estimate its location to be between the two areas, as opposed to actually in either. For such a distribution, the particle filter is an attractive option.

Particle Filter

Another probabilistic approach maintains a particle filter to use multiple particles to represent the belief distribution of a tracked target [Dellaert et al 1999]. In each new frame of data, the particles are each scored according to how well the sensor data (underlying pixels, for example, in the case of vision) supports the target being at the location of the particle. The particles are then re-sampled according to their score, resulting in the same number of particles, but better chosen to reflect likely target locations. The new particles are averaged; this is where the target is said to be located in this frame. Finally, each particle is stochastically moved according to the motion model, readying them for the next iteration. Figure 2.5 illustrates this process.

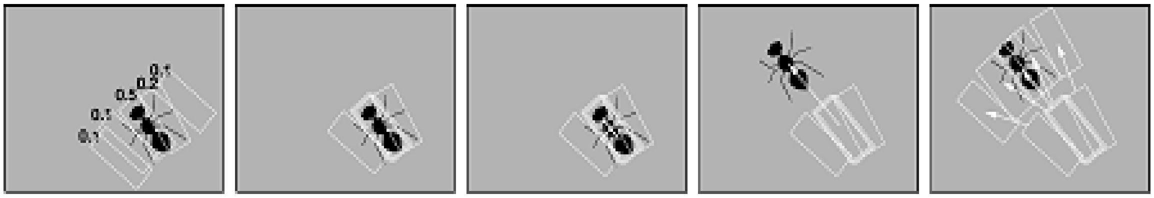


Figure 2.5: (a) White rectangles represent particles, scored based how well the underlying pixels match the model. (b) New particles are sampled according to the probabilities. (c) Estimated location (white plus) is calculated from the new particles. (d) The next image frame. (e) Particles are advanced according to the stochastic motion model. From [Balch et al 2005].

One appeal of particle filters is that, unlike the Kalman filter, they are well suited to representing data with a multimodal distribution. This ability serves to enhance the robustness of the underlying state estimation process [Gutmann 1998]. Particle filters, therefore, are used in many trackers, both computer vision-based [Khan et al 2004a] and laser-based [Panangadan et al 2004]. Particle filters are also used in other state estimation problems including mobile robot localization [Fox et al 2001] and dynamic probabilistic networks [Kanazawa 1995].

Joint Particle Filter

When multiple targets are non-interacting, using multiple independent particle filters is sufficient, but in cases when targets interact, as is common in multi-agent systems, a joint particle filter could model these interactions. However, the intense computation required makes this infeasible beyond a few targets. Khan et al [2003] takes advantage of the limited perception of their targets (ants) to create a joint MRF (Markov random field) particle filter. In effect, this adds an "interaction term," allowing more accurate motion models among close targets (e.g. two targets cannot occupy the same space) without suffering from the intractability of a full joint particle filter (Figure 2.6a and b). An extension in this research penalizes particles which overlap the location of other targets, thus violating known constraints on movement (Figure 2.6c). To achieve an even greater efficiency, work has been done to replace the traditional importance sampling step in the particle filter with a Markov chain Monte Carlo (MCMC) sampling step [Khan et al 2005]. This has also been extended to address a variable number of targets.

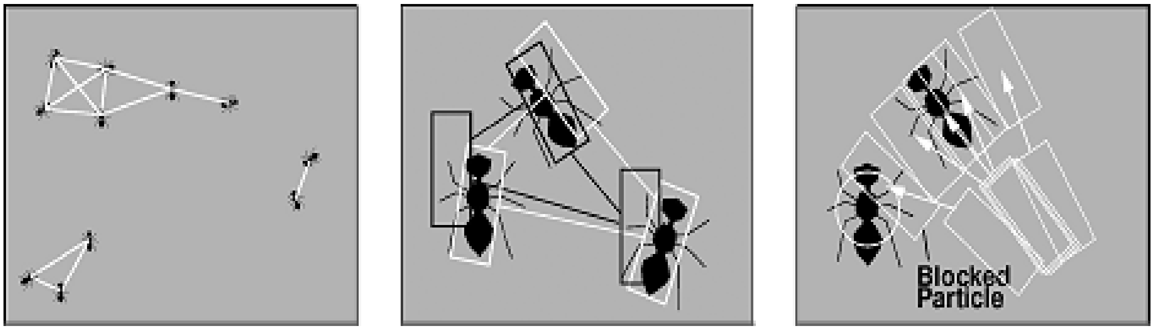


Figure 2.6: (a) Only ants in proximity are considered jointly. White lines indicate joint considerations. From [Khan et al 2003]. (b) Nearby ants make use of particles which encapsulate their poses (two particles are indicated by white and black lines. (c) Within a filter, particles which violate motion constraints are blocked. From [Balch et al 2005].

Another approach uses laser-based data with a motion model in conjunction with sample-based joint probabilistic data association filters (SJPDAFs) to track multiple moving objects [Schulz et al 2003a]. By using the motion model, the tracker is able to maintain a spreading sample set to estimate the location of occluded objects. In other research [Schulz et al 2003b], they use Rao-Blackwellised particle filters to estimate locations of uniquely identified objects, beginning with anonymous tracking and switching to sampling identification assignments once enough identification data has been gathered, resulting in a fully Rao-Blackwellised particle filter over both tracks and identification.

2.5 Unique Track Identification

Laser range finders provide excellent data for tracking targets as they move through an environment. Unfortunately, they do not have a means of differentiating between multiple targets of the same type. In other words, all people (or monkeys or other targets) look alike to a ladar. Therefore, to provide unique identification, another sensor must be used. Specifically, it must be a sensor that can differentiate targets based on their "appearance" in the data, for instance through color-indexing [Swain & Ballard 1991] or facial recognition in camera images [Stillman et al 1998] or the unique identifier provided by infrared (IR) badges [Schulz et al 2003b]. Such a sensor must be used in lieu of the ladar to provide both tracking and identification, or else combined with the laser-based data to create uniquely identified tracks.

Of course, vision tracking is only appropriate if the targets are visually distinctive. While this may work for human targets, homogenous robot or monkey targets could all look alike. On the other hand, there are a variety of different types of sensors that could be used to provide the needed functionality. Many of these have been used by a number of researchers for several years.

For instance, IR transmitters and receivers have been used by putting the receivers at known locations and transmitters on a badge worn by the person being tracked (the Active Badge system [Want et al 1992]) or vice versa [Azuma 1993]. Such systems suffer from the poor range of IR and therefore require significant infrastructure intrusion and installation, resulting in a high cost for even small scalability. Ultrasound techniques [Harter et al 2001] suffer from the same disadvantages as IR (in one experiment, 100 receivers were needed to cover an area of 280m^3), plus impose strict restrictions on the number of signals that can be received every second.

Neither IR nor ultrasound techniques are well suited to functioning in outdoor environments. On the other hand, the global positioning system (GPS) can provide precise localization, but is only effectual in outdoor environments, as buildings tend to block the satellite signals. None of these technologies are ideal for a system which must work both indoors and outdoors.

Another approach (overviewed in [Hightower 2001]) relies on electromagnetic sensing to track positions. Systems such as the MotionStar DC magnetic tracker provide phenomenal localization and identification accuracy, but require the targets to be tethered to the base station, and only support a range of less than 10 meters. While suited for motion capture applications, such systems would not work for tracking targets in their “natural” settings.

One technology that works both inside and outdoors is radio frequency (RF). There are a number of ways to use RF for tracking and localization. Section 2.5.1

describes one common RF machinery example, RFID, while Section 2.5.2 details some ways of using the signal strength of the RF signal to localize. Finally, Section 2.5.3 discusses ways in which data from multiple sensors can be fused together to provide more accurate or robust output.

2.5.1 Radio Frequency Identification (RFID)

RFID sensors are one type of ID-sensor which can be used to uniquely identify targets. In fact, RFID provides perfectly accurate identification because each sensor is given a unique identification number (like a serial number). RFID sensors consist of two parts; a tag and some sort of reader. The tag broadcasts the ID number (and potentially additional information) at a specific RF frequency and nearby readers tuned to the frequency can pick up the broadcast. RFID sensors have seen a great deal of development in recent years [Finkenzeller 2000], and can be divided into passive and active types.

Passive RFID

Passive RFID tags consist of a small integrated circuit, but no power supply. As such, they are very small and flat, often smaller than a grain of rice. Instead of being internally powered, the tags are activated when brought in close proximity to a reader. The incoming radio signal sent by the reader induces a small current in the tag; just enough to power up and transmit a response. Therefore, they have a very short range, generally not more than 6 to 24 inches, although in some cases they can be detected up to approximately 6 meters [Hähnel et al 2004].

These passive tags have made their way into a variety of common uses. They can be found in credit cards and passports, consumer goods, and even pets such as dogs and cats. In credit cards and passports, they serve a similar function to magnetic strips, containing identification information that can be used by the reader. By implanting in dogs and cats, a found pet can be scanned to locate find its owners, even if the animal was not wearing traditional tags. These tags are also often placed on consumer goods both for inventory tracking and theft deterrence [Curtin et al 2006].

Passive RFID tags are also used in intelligent systems research. For instance, the Guide project [Philipose et al 2003] uses a small reader attached to a home patient to identify the objects a person touches. The sequence of objects is used to infer the person's activity and provide support if necessary. Hähnel et al [2004] investigates generating maps of RFID tags (placed on a variety of objects throughout an indoor environment). The resulting maps can then be used for accurate localization of the robot and objects without odometry information. Unfortunately, these results require cumbersome equipment (the reader, antennas, power supply) to be placed on the mobile robot, which would not be appropriate on living targets. Additionally, with a read-range of 6 meters, a larger environment would have to be instrumented throughout, a proposition that is not always appropriate.

In general, passive RFID systems are designed to read one tag at a time. However, Vogt [2002] created a way of identifying multiple objects simultaneously, without explicitly knowing the number of objects. While such a technique would be useful in certain applications (such as supermarket checkout), the short range of passive

RFID technologies is not conducive to tracking individuals at the scale of dozens of meters.

Active RFID

Unlike the passive RFID technology, active RFID tags include a built-in power supply. The tags, therefore, broadcast their message at regular time intervals, whether a reader is within range or not. Further, because they have much more power available than the induced current of the passive tags, active RFID tags' signals can be received at much greater distances, often 100 feet or more.

The added range of active RFID tags makes them better suited for localization than are passive RFID tags. In addition to readers detecting the message which each tag broadcasts, the reader can determine the signal strength of the reading. This signal strength varies based on the distance from the reader. Further, antennas can be attenuated so as to restrict the distance at which they can receive a signal. This is useful when tracking the location of targets at the resolution of a room; with one reader per room, the reader that detects the tag indicates the tag's location. Hospitals use such technology to monitor the location of equipment, staff, and patients (especially babies) [Wang 2006].

Active RFID tags are used in a variety of industries and applications, including transportation ("E-Z pass" and other toll-road automatic payment systems), athletics (to track marathon runs on the course), and inventory management. These and other applications are detailed by Schneider [2004].

Locating targets with active RFID at the room-level is simple, but trying to determine location more finely is difficult. This process requires using the signal strength from multiple readers to triangulate in order to find the source of the signal. Worse, when the targets are moving rapidly compared to the temporal frequency of the tag signals, the system is more susceptible to noise, as multiple readings cannot be averaged to determine location. Further, the location of the target in between tag readings is unknown. A number of methods of dealing with these uncertainties exist, and are described in Sections 2.5.2 and 2.5.3.

2.5.2 Localization Estimation from RF Signal Strength

Although RF techniques (such as RFID) are ideally suited for identification, they present substantial hurdles for accurate localization over large distances. The signal strength reported by a reader is proportional to the transmitter's distance from that reader; triangulation can therefore be used to localize. However, there are a number of factors that reduce the robustness of this approach, including multipath fading and shadowing of the RF channel, as well as transmitter and receiver variability and antenna orientation [Lymberopoulos et al 2006]. They show that it is possible to map signal strengths to distances from an antenna only under the most ideal circumstances – any variation in orientation or presence of obstacles introduces large amounts of noise. Worse, even with no direct occlusions between the tags and readers, RF signals are susceptible to the presence of metal in the surrounding area [Balch et al 2004]. This is a well studied problem ([Howard et al 2003], [Ladd et al 2002], and [Haeberlen 2004]), with research progressing along several techniques.

One approach to solve this problem attempts to build a model of the signal strength propagation through space, and then calculate the expected signal strength at every location based on the distance from the readers. For instance, RADAR [Bahl &

Padmanabhan 2000] builds a radio propagation model which estimates signal strengths at each location after taking into account effects based on the number of walls between that location and the reader. RADAR also investigates an empirical method of gathering readings from a number of known locations, and then localizing new readings by finding the most similar readings (in signal strength space) in the training set. While the model method was accurate to within 4.3m (in the 50th percentile), the empirical method achieved an accuracy of 2.94m. Letchner et al [2005] also attempt to build a signal strength sensor model, though theirs is learned through the use of a hierarchical Bayesian framework, and achieves a median localization error of less than 2m.

Youssef et al [2003] use WLAN access points as the transmitters; a mobile receiver can localize based on the joint probability distribution of the *q* strongest signals. Location clustering is used to reduce the computational load. This joint clustering technique achieves an accuracy of approximately 2.3m in a large indoor environment. Roos et al [2002] also use a probabilistic approach. Instead of directly modeling the physical properties of the signal propagation, they model signal strength distribution in different geographical areas based on sample measurements. Three machine learning techniques (nearest neighbor, kernel regression, and histograms) are used to predict likely locations based on readings – they achieved an accuracy of 1.5-2m.

One problem with these model-based approaches is that signal strength propagation is very difficult to accurately model. There are a variety of environmental factors which reduce the effectiveness of generalized models. Therefore, some systems take a more example-based approach. Much like RADAR's empirical method, these techniques gather many samples of readings at known locations and attempt to match new readings to one or more examples from the training data. MoteTrack [Lorincz & Welsh 2007] extends this basic approach by improving robustness and decentralization, achieving errors of only 2m (50th percentile) even with the failure of up to 60% of the environmental beacons.

Although this method has promising results, one disadvantage is the massive amounts of training data that must be collected; error increases when generating likelihoods at locations for which no training data exists. Ferris et al [2002] overcome such limitations using Gaussian processes to generate a likelihood model for signal strength measurements from a limited number of training measurements. This process allowed them to extrapolate the model into areas for which no training was left out, with accuracy comparable to experiments that included such data. Overall, the accuracy indoors (54 rooms, hallways, and stairs across 3 floors) was about 1.5-2m, while the accuracy outside (in a 500km^2 area) was on the order of 100-200m.

Another way to get around the sparse data problem is to discretize the space into grids, and then determine the probability of being in each grid cell based on readings. By grouping nearby points into a single grid cell, the number of potential locations is reduced (from infinity), greatly limiting the number of training examples required. This is the approach used by Kantor & Singh [2002] to achieve an average accuracy of 1.62 feet (in a test area of 50 feet by 50 feet). In simulation, their results are improved to an average error of 0.77 feet by including odometry data.

Many of the above techniques have achieved localization accuracies on the order of 0.5 to 2 meters. This level of accuracy is appropriate for many applications. However, the target domains of this research requires tracking with a much lower level of error, such as is provided with the laser data. Further, the level of hardware that most of these systems require on the mobile targets is prohibitive in both sports and non-human primate domains. The smaller RFID tags that are ideal for introduction into these situations generally introduce even more noise than the RF instruments discussed above. Therefore, it is necessary to combine data from multiple sensors in order to achieve accurate, uniquely identified tracks. This process of joining data from multiple sensors is referred to as sensor fusion.

2.5.3 Sensor Fusion

By combining sensor modalities, the benefits of each type of sensor can be used to help offset the deficiencies of the other. This is done in a variety of situations. Singh et al [2002] uses both camera images and inertial data, combined with a Kalman filter, to perform simultaneous localization and mapping (SLAM). Kalman filters and other high- and low-level sensor fusion techniques are reviewed by Kam et al [1997]. Hähnel et al [2004] shows that including passive RFID readings can greatly accelerate laser-based robot global localization (compared to lasers alone).

A number of different sensor modalities are available to assist with tracking. For example, vision-based trackers are good at tracking distinctively colored objects, while laser-based trackers can easily detect moving objects (regardless of coloration). Jung & Sukhatme [2001] combine cameras and laser range-finders to capitalize on these traits, although this does not provide target identification.

Instead of combining the readings from precise and noisy sensors, another approach is to use the precise sensors (ladar) to generate the trajectories, and then use the noisy ID-sensors to perform the target identification. Schulz et al [2003b] use such an approach to track people in an indoor environment, generating anonymous tracks from ladar data, and then assigning IDs as the tracks approach short-range IR receivers. The research presented here uses a similar technique, but uses long range (though noisier) active RFID tags to allow identification in a larger area, and without instrumenting the interior of the observation area. Additionally, a much greater number of targets are tracked and identified than in Schulz et al [2003b], which also suffers from an inability to recover from losing the correct ID hypothesis.

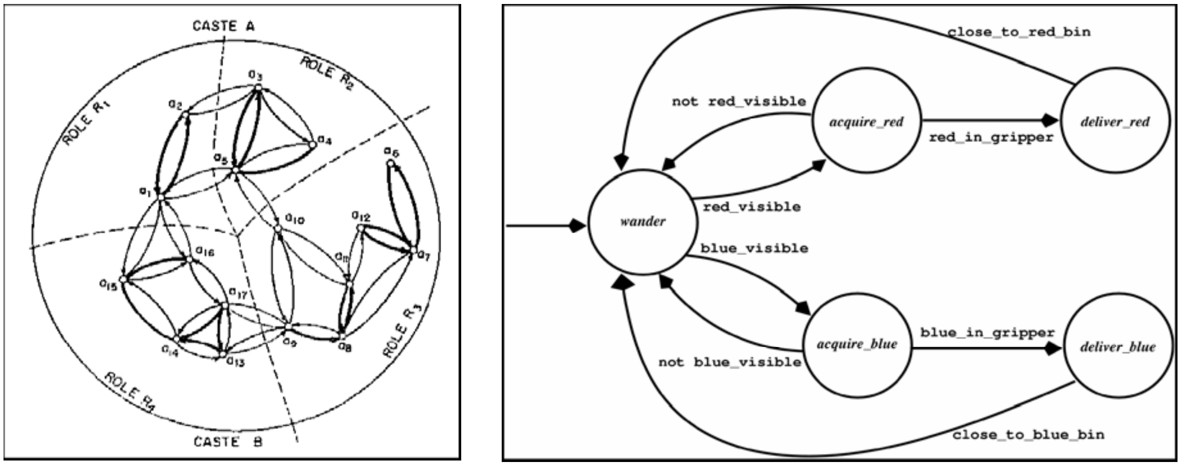


Figure 2.7: (a) Ethogram representation of ant behavior. From [Holldobler & Wilson 1990]. (b) Markov model of a robot's behavior. In both models, arrows represent transitions between actions (nodes). From [Balch et al 2005].

2.6 Behavioral Modeling

Once a track consisting of the location (and orientation) of each target (or "agent" in the recognition literature) over time has been generated, this data can then be examined to determine behavior. This identification of behavior can be important in a variety of situations. For example, in robot soccer, behavior recognition would allow for adaptive strategy (i.e. changing strategy based on what other players are doing) and automated narrative agents, which could detect what a player was doing (at a high level) and offer interesting commentary [Han & Veloso 1999]. Further, behavioral ecologists use ethograms to model animal behavior [Schleidt 1984]. As Figure 2.7 shows, these ethograms are similar to the Markov models used to represent robot behavior.

There are several ways to implement this behavioral modeling. Some techniques are relatively simple, including using thresholds to train models from labels provided by a human expert and building sensory models based on beliefs about the agent's perceptual apparatus. More complicated techniques include kernel regression techniques, hidden Markov models, and switching linear dynamic systems. The following sections will discuss each of these methods, after giving a brief overview of the sorts of behaviors that can be modeled and recognized.

2.6.1 Sensory Models

Sometimes, it is not possible to directly observe the studied behavior. For instance, one social behavior of interest to myrmecologists is the interaction between two ants [Pratt 2005]. This interaction usually takes the form of antennal contact on the part of one or both ants (the contact can be antenna to antenna or antenna to body), although contact between the bodies of two ants is also of interest. In cases such as these, one method of identifying the interactions is by modeling the sensory fields of the agents. For example, Egerstedt et al [2005] model the sensory field of an ant with two simple geometric shapes, one representing the body and one for the head, as shown in Figure 2.8. Thus, interactions were detected whenever two ants' sensory fields overlapped; the type of interaction (head-to-head, head-to-body, etc) was determined by which fields overlapped. The basis of this model was created to study army ant simulations [Couzin and Franks 2002]. One problem with this type of model is that a new model must be created for each new application, making it difficult to apply in general [Balch et al 2005].

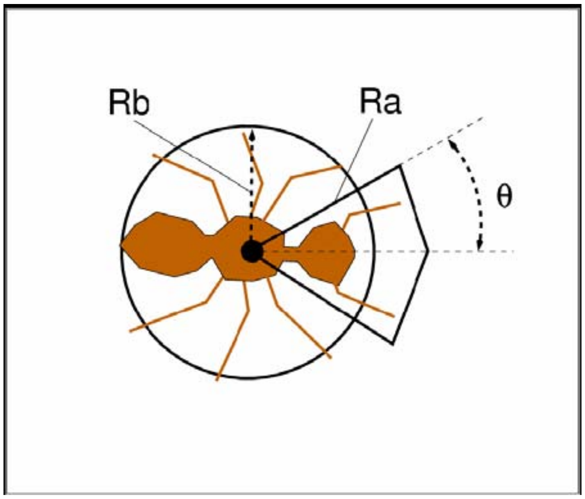


Figure 2.8: Black lines model the sensory fields of the ant. When these fields overlap with another ant's fields, an interaction is said to be taking place. From [Balch et al 2005].

2.6.2 Trainable Models

The basis of this category of techniques is in using labeled examples to create the model. This requires a human expert to label a sampling of data (called the training set) manually, but allows a general system to be built that does not rely on devising a completely new model for each application (as would be required for sensory models). Instead, a model of the human's labeling is created; this model is then used as a reference to label new examples. The advantage of this approach is that the human expert does not have to explicitly specify the considerations used in classification. Trainable models are in widespread use, in a variety of applications, including gesture recognition, face recognition and activity recognition.

One such type of trainable models is the hidden Markov model, which is discussed in detail in Section 2.6.3. Another representation of human trained models is the decision tree [Arkin et al 1993]. In a decision tree, each leaf node is given a classification, while all other nodes represent a splitting based on some attribute. Examples are classified by sorting them down the tree, from the root to some leaf node; the example is given the classification of the leaf node at which it ends. Decision trees are suited to such tasks as medical and equipment diagnosing and credit application approval [Mitchell 1997]. On the other hand, decision trees suffer from problems when dealing with continuous inputs or classifications, and overfitting is commonly a problem, though more complicated extensions have been developed to help handle these situations [Mitchell 1997].

Intille & Bobick [1999] take a different approach. They use a probabilistic framework to build a football play recognition system. Plays are defined as a series of temporally ordered goal actions carried out by the players. For a given set of trajectories, each play recognizer returns the likelihood that the play was executed – the play with the highest likelihood is chosen. One disadvantage of this system is that it requires a knowledge engineer to design the action description, effort that would have to be repeated for each specific application. Also, while their results display a high level of accuracy, the system suffers from a slow runtime, which would not be ideal for the applications being studied by this dissertation.

Another technique is *k*-nearest neighbor learning. In this method, data points are classified based on their location in an n-dimensional feature space (where n is the number of features). First, the training set is used to populate feature space.

Classification occurs by evaluating each new data point in the populated feature space. At the algorithm's simplest level, the *k* nearest data points are examined and the most common classification among these points is given to the new data point. More broadly, kernel regression works similarly, but applies a function to "score" the contribution of each data point to the classification (instead of taking the "one point, one vote" approach); for example based somehow on distance so that farther points are less decisive in the classification [Smola & Schlköpf 1997]. In this case, *k* is often set to the number of training data points (a *global*, instead of *local*, method) [Shepard 1968]. Unfortunately, this has the limitation causing the algorithm to run slowly, as all the distances have to be computed for each new data point (though a *kd*-tree can speed it up, at an increased cost in memory [Friedman et al 1977]). Also, if there is a large difference between the numbers of training points in each classification, a bias is introduced.

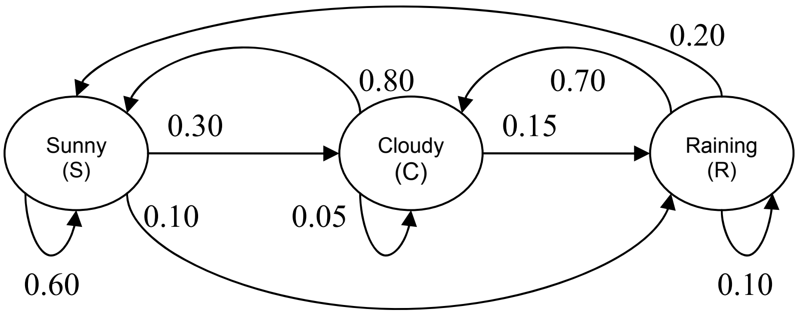
One problem with most of the techniques discussed in this section is that they consider each new data point individually, without examining the data before or after. Yet, with many data streams, information can be gained by considering the sequence as a whole. The following approaches, on the other hand, use all of the data, which can provide a "smoothing" influence that tends to suppress brief "noisy" detections.

2.6.3 Hidden Markov Models

Hidden Markov models (HMMs) are probabilistic generalizations of Finite State Automata. They can be used to represent Markov processes in which the underlying state of the system is unknown, or hidden. Instead, it is some observations emitted by the process, based stochastically on the current state, which can be seen. Further, movements of the system from state to state are modeled in the HMM by probabilistic transitions.

Formally, an HMM consists of a set of states, a set of observations, the probability distribution table of each state emitting each observation, the probability distribution table of each state transitioning to the other states (or itself, as a “self-transition”), and an initial probability distribution table indicating the probability of the system starting in each one of the states.

As an example, consider a three state HMM designed to model the weather. In this simple example world, assume that there are three possible states of the world: sunny, cloudy (but not precipitating), and raining, and that the weather on a given day is only dependant on the previous day’s weather. Further assume that there is no way to directly observe the sky; instead, it is only possible to measure the dryness of a small patch of concrete. It can be dry, damp, or wet. Figure 2.9 shows one possible HMM modeling this situation, including hypothetical probabilities. Consider, for example, that it is unlikely to become sunny directly after raining, without entering the cloudy state. This is reflected in a low transition probability between these two states. Also, assume that this HMM is designed to model the weather in a rather dry location, where short, quick rainstorms are common. This is modeled by the high self-transition probability of the sunny state and the low self-transition probability of the raining state.



		Observation		
State		Dry	Damp	Wet
	R	0.03	0.07	0.90
	C	0.50	0.30	0.20
	S	0.70	0.20	0.10

Figure 2.9: Hypothetical hidden Markov model describing the weather in a simple world. Numbers on the arrows are transition probabilities, while the table gives the observation probability of each possible observation while the world is in the given state.

How are the probabilities in the model determined? One way is to simply look at the frequency of occurrences of each observation and transition. This only works if there is a large body of data which contains the observation and actual state. If such data exists, it would be possible, for example, to determine the percent of sunny days in which the concrete is wet, damp, and dry. Likewise, the sunny transition probabilities would be calculated as the percent of instances in which day *t* is sunny and day *t+1* is sunny, cloudy, or raining. By determining each of these values, the observation and transition probability distribution tables can be completed. Finally, the initial state distribution is calculated as the percentage occurrence of each type of day (sunny, cloudy, and raining).

While the above method works well in many circumstances, sometimes the required body of "labeled" data (consisting of observations and true states) is unavailable. In these cases, the HMM parameters (λ) must be learned. This can be accomplished using the Baum-Welch algorithm, which uses expectation-maximization to compute maximum likelihood and posterior mode estimates of λ [Rabiner 1988].

Once an HMM's parameters have been decided, one of the primary uses is to determine the most likely underlying state sequence for a series of observations. For this purpose, the Viterbi algorithm can be used. Viterbi uses dynamic programming to determine the most likely sequence of states to have generated the given observations [Rabiner 1988]. In the above example, this is akin to determining the actual weather on a series of days, based on the concrete moisture level observations on these days. The closely related forward algorithm computes the probability of an observed sequence being generated by a specific model. This functionality can determine which of several HMMs most likely generated an observation sequence, useful for behavior recognition.

HMMs can also be executed, resulting in a sequence of states/observations created probabilistically.

2.6.4 Behavioral HMMs

If one assumes that an observed agent acts according to a Markov model, it is possible to employ HMM-based approaches to identify its behavior. These behavioral HMMs, or BHMMs, are ideal because the behavioral state of the agent is "hidden," while the manifestations of that behavior are observable and can act as the observations of the BHMM. A separate BHMM is created for each potential behavior, which is made up of a series of state traversals. As an observation sequence is gathered, the likelihood of being in each BHMM is determined using the Viterbi algorithm [Rabiner 1988]. Further, the exact mental state of the agent corresponds to the specific active state in the BHMM.

To account for the continuous nature of robot behaviors, which does not provide easily detectable gaps, recognizers are instantiated at regular intervals. These instantiations are then terminated after a fixed amount of time, or if a reject state (a "catch-all" state which represents states that are unlikely to occur in a given behavior) is probably reached [Han 1999].

Another example of using HMMs for behavior recognition, specifically gesture recognition, is to recognize American Sign Language gestures [Brashear et al 2003]. In this system, a "first person" camera is combined with data from accelerometer-covered gloves. HMMs are then trained to model the gestures representing a variety of words. This system was able to achieve a recognition rate of over 90% on test data, while a 94% accuracy on the training data shows how well the HMMs can model the data.

HMMs have also been used to detect the self-stimulatory (or “stimming”) behaviors which are often displayed in children with autism [Westeyn et al 2005]. Accelerometers placed on the child provide data to the system made up of models of seven stimming behaviors. When the system was presented one at a time with isolated examples of these activities, performed by a neurotypical adult, it achieved a recognition accuracy of 90.95%. Further, in continuous recognition experiments, all stimming activities were detected, though the exact start/stop frame of the data was not always correctly detected – not necessarily a problem in this and other domains, where the goal is to identify the occurrence and type of activity taking place.

Sometimes the transitions or observations of a system are dependant on some input into the system, such as sensory perceptions. For example, a behavior change may be triggered by a visual or auditory cue, such as an ant detecting a pheromone trail. In these cases, the model would be more accurate by representing behaviors as conditional responses to input. Input/output hidden Markov models (IOHMMs) provide just this functionality. They are similar to HMMs, except the transition and observation probabilities are conditional on the value of the input [Bengio & Frasconi 1996].

2.6.5 Switching Linear Dynamic Systems

Another approach to recognition uses Switching Linear Dynamic Systems (SLDS) to model behaviors. The main advantage of SLDSs over HMMs is their ability to model continuous hidden states. This is useful when additional information is desired, beyond simply knowing the state of the model. For example, it has long been known that the honey bee’s waggle dance encodes specific information related to the location of the food source being passed on to the other bees [v. Frisch 1967]. One possible area of biology research which could benefit from automation would be to not only indicate when a waggle dance is occurring, but to extract the exact information encoded in the dance. Although an HMM could detect the occurrences of a waggle dance, it could not extract the encoded information. On the other hand, an SLDS could actually provide an estimate of this information directly to the biologist [Oh et al 2005].

Unfortunately, SLDS models suffer from three main limitations: Exact inference is intractable, there are limitations in duration modeling, and there is an absence of a systematic way to quantify global parameters. Solutions have been proposed for all of these, by using a data-driven MCMC inference method with a segmental, parametric SLDS (SP-SLDS) [Oh et al 2008].

2.7 Domains of Interest

The research presented here span many application domains. Uses of these techniques are varied, and include such areas as social insect, primate, and human systems. Even within each of these areas, there is a vast diversity of spheres being studied. For example, social insect systems include ants, bees and locusts. Applications to human systems include assisting those with health problems [Starner et al 1998], providing video surveillance for security and safety [Cohen & Medioni 1999], and even monitoring or providing commentary in sports settings [Pingali et al 1998]. One commonality of these domains is that they currently benefit (or could potentially benefit) from the automation provided by artificial intelligence, both in tracking the subjects and modeling their behaviors.

2.7.1 Social Insect Systems

Historically, there have been several examples of applying lessons from the behavior of ants, bees, and other social insects to computer science problems. For example, inspiration for solutions to discrete optimization problems has arisen from the observation of social insects, such as ant colonies foraging. These “ant algorithms” have been employed in a variety of domains, from the traveling salesman problem to routing in telecommunications networks [Dorigo & Di Caro 1999]. In fact, whole books have been written about using models of biological systems as inspiration in the design of complex systems [Bonabeau et al 1999].

However, until recently, there has been little of the reverse application; using technology and computer science techniques to assist in biological research. The decades old pencil and paper direct observation method of biological research [v. Frisch 1967] has seen some progress through the use of technology. For example, Mallon et al [2001] use a video camera to record the ants being studied, permitting later review and analysis. Likewise, when a researcher today studies honey bee colony behavior, the subjects are videotaped and the resulting tape is viewed and hand-labeled [Seeley 1995].

Although this advance allows the researchers to return to the taped data, they are still forced to analyze it by hand, manually generating models of the observed behavior. Typically, this requires the observer to watch the video many times, and is a rather time-consuming process. However, if the movements and behaviors of the subjects could be recognized and identified automatically, research in these areas could be greatly accelerated by allowing the researchers to focus their time on the interpretation of the data, instead of simply gathering it. For example, Pratt et al [2002] were forced to watch hours of video to count the number of recruitment events in the migration of a colony of the ant *Leptothorax albipennis*, a task well suited to computer automation. In fact, it has been stated that two observers are required to study the activities of one ant (one to call out the observations and the other to record them) [Gordon 1999]. Imagine the complications of social animal studies involving numerous subjects interacting with one another!

2.7.2 Primate Systems

In addition to work with social insect systems, higher-order biological systems have also provided a venue for the application of computer science automation techniques. One such recent study provides assistance to a behavioral neuroendocrinologist. In this research, the biologist evaluates the spatial memory in rhesus monkeys by measuring the paths the monkeys take as they explore an outdoor three dimensional arena over repeated trials [Khan et al 2004b]. Previously, the biologist was required to measure the path length indirectly (by timing the monkey) or through estimation. However, by using computer vision to track each monkey as it moves through the arena, exact path length can be easily determined.

Another study that will benefit similarly involves examining the behavior of a large number of individuals simultaneously, similar to Wallen [2005]. In this research, the biologist is interested in the social interactions of a group of monkeys. As with the social insects mentioned above, the introduction of interactions dramatically increases the time required to perform observations. Therefore, this is another area which could potentially benefit greatly from computer automation.

2.7.3 Human Systems

Some of the biggest impacts of automated tracking and behavior modeling fall into this category. Two of the first applications of multi-target tracking were air traffic control and battlefield surveillance [Reid 1979]. Since then, surveillance has become a major area of research, including both safety and security applications. For example, Coifman et al [1998] developed a feature-based tracking algorithm for tracking vehicles on a highway. The goal involves determining the flow of traffic (number of vehicles per hour), which could be used to identify accidents or other traffic incidents. Yan & Matarić [2002] analyze spatial features for recognizing the activities of multiple interacting humans to be applied to a video surveillance system or narrating agent. Outdoor surveillance of person-vehicle interactions such as pick-up and drop-off has also been studied [Ivanov et al 1999].

There are many other applications of tracking and behavior modeling of human systems. For example, behavior modeling is used in the study of human gaits [Bregler 1997], speech recognition [Jelinek 1998], and gesture analysis [Darrell et al 1996]. In the area of sports, Perš & Kovacic [2000] and Intille & Bobick [1995] use computer vision to track human soccer and football players, respectively, while Han & Veloso [1999] apply behavior recognition to robotic soccer to assist in controlling the players. Further, Pingali et al [1998] also use computer vision to track a tennis game in real-time for enhanced broadcasts.

Much of the work in this domain relies on video sensors, but some research has involved laser range finders as the primary sensors. For example, Prassler et al [1999] (discussed above) use lasers to track humans moving about an indoor environment, while

Zhao & Shibasaki [2004] track pedestrians in a wide open area (such as a shopping mall) with laser range finders placed at ankle level.

2.8 Discussion and Summary

This chapter presents a review of the important existing work which relates to the dissertation. Approaches which are built upon in this work are examined, as are alternate techniques for solving the problems addressed herein. The key points are:

- **Laser range finders** are growing in popularity as a sensor ideally suited for tracking targets over time. However, they cannot be reliably used distinguish specific individuals from each other.

- **Active RFID Tags** provide unique identification, but sensors which rely on the strength of RF signals to localize are very noisy. A great deal of research has gone into solving this problem in various domains with some success at coarse resolutions (1-6 feet).

- **Hidden Markov models and kernel regression** are two machine learning techniques that can be applied to behavior recognition tasks.

- **Many domains** which benefit from research towards automatically tracking and recognizing behaviors. These include biology, sports, robotics, and safety/security.

Other work differs from that presented here in a number of significant ways. First, most existing research on tracking involves only a single kind of sensor as input.

Those who use multiple types of sensors generally combine all data in order to improve localization (such as through the use of a Kalman filter). Instead, this work relies on each sensor modality to independently solve a separate part of the task, thus minimizing the effects of the sensors' failings. Additionally, this tracking algorithm is quick enough to function in real-time at a reasonable frame rate. Further, other researchers studying behavior recognition focus on single agents acting alone, whereas this research examines social behaviors among multiple, interacting agents. Finally, the problem domains that this dissertation concerns itself with (sports and biological systems) are relatively unstudied, especially with regards to the techniques and approaches taken herein.

CHAPTER 3

LASER-BASED TRACKING

This laser-based tracking approach focuses on automatically tracking the number and locations of multiple animals, objects or people (hereafter, "targets") in a dynamic environment, either indoors or outdoors, as they move rapidly through the environment over time. It is robust to uncertain and changing lighting conditions. This method accurately computes the tracks of a varying and unknown number of moving and interacting targets over time. These tracks can be generated in real-time or created from previously logged raw sensor data. The approach uses multiple laser range finders that record targets' positions. It removes "uninteresting objects" (i.e. the background) and accounts for individual targets in close proximity to one another. The ultimate result is a series of snapshots of the positions of targets as time unfolds. Individual targets in these snapshots are strung together, creating tracks representing an individual's location over time.

The key idea of this technique uses an iterative closest point (ICP) algorithm to determine the optimal placement of one or more models (or templates) representing the targets to be tracked. It exploits estimated locations of targets in previous frames to initialize model placement in the next frame. The data is processed in several phases, namely – data collection, registration, background subtraction and tracking. Figure 3.1 provides an overview, illustrating the flow of data from one phase to the next. First, in the data collection phase, the laser range finders record the targets in the area of interest.

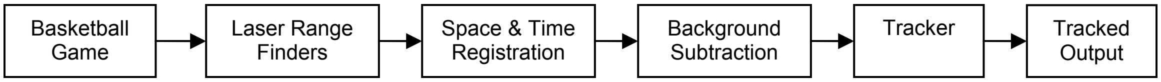


Figure 3.1: Overview of the tracking system.

This data can be processed immediately or logged to disk for later tracking. In the registration phase the data is passed to several modules, which register the data in space and time. The data is then run through the background subtraction module to remove extraneous laser hits not related to the targets. Finally, in the tracking phase the processed and formatted data is passed to the tracker, which computes tracks representing the location of each target, using a model-based, ICP tracking algorithm. The rest of this chapter details the approach taken in each phase and introduces the experiments and metrics in which it is tested, before ending with the results of these experiments and a brief summary.

3.1 Registration

A ladar captures data with respect to its own point of view, both spatially and temporally. This results in isolating each sensor's data from a comparison with the others. In order to utilize multiple sensors, output of each ladar is combined into one global "picture."

To accomplish this, ladars are aligned in both space and time, creating a global point of view. Synchronizing the sensors' measurements in time (and in space) ensures that all scans correspond to one another. Further, registering the measurements in space allows the increase in coverage provided by using multiple sensors.

Data is timestamped according to when it appeared in real time. Ladars record data continuously and independently of each other. In this approach, time is discretized in order to synchronize the data among the different ladars. First, a master log is created starting at the timestamp of the first scan and progressing in 26.67 ms increments (corresponding to the scan rate of 37.5 Hz), rounded to the nearest millisecond, to the timestamp of the last scan. Scans from each laser are matched up to the master log entry which minimizes the overall difference between scan times and master log times. This serves to correct for the buffering issue, which results in scans being given timestamps that generally increment by between 15 and 46 ms, despite the ladars generating the data at a vary precise rate. Further, this method corrects for the occasional laser scan which is lost due to corruption or communication buffer overflows.

Data is coordinated spatially and transformed into one global coordinate system by converting from polar into Cartesian coordinates. To do this, the location of each ladar in relation to the others is pre-computed. An initial "best guess" of global location and orientation from each ladar is used. The exact location and orientation of each ladar is fairly straightforward to calculate. A ladar is chosen as the primary ladar; its location and orientation is the ground truth with which the other ladar match up accordingly. Each ladar's data, in turn, is compared with the primary ladar's data in x/y space using the initial "best guess" placement. An error is generated by summing SQRT(d_i), where each d_i is the distance between each of the new ladar's laser hits and the nearest laser hit in the primary ladar's data. Small moves to the initial location and orientation of the new ladar are attempted, with the change that reduces the error the most accepted. This process is iterated until no move results in a lower error, and then is repeated with smaller and lastly even smaller moves. The final location and orientation of each laser is now best matched to the primary laser.

Interestingly, although d_i^2 is often used in calculating error, for this algorithm SQRT(d_i) works better. This is because several laser hits in a ladar's scan may not correspond to laser hits in the primary ladar's data (due to the perception of different objects from different points of view). It is expected that these laser hits would be far away from any other laser hits, since they truly do not appear in the other ladar's scan. However, by using d_i^2 to calculate error, these large distances would skew the desired result. Therefore, large distances are weighted less compared to small distances (which would represent laser hits that are more likely to correspond to each other).

This whole process can be likened to superimposing each subsequent ladar's data on top of the primary ladar's data. Rubber bands are attached from the subsequent ladar's hits to the nearest primary ladar's laser hit. The primary data is held fixed, while the subsequent data is "released," allowing it to slide about until equilibrium is reached. Because each rubber band prefers a state of lesser stretching, the "error" (length of each rubber band) is minimized. This process is repeated, connecting the bands to the new nearest laser hit, until no movement results.

3.2 Background Subtraction

In order to isolate data that correspond to the objects that are tracked, hits that represent the background are removed. Typically, the background is made up of stationary objects (e.g., the wall and chairs) and targets that are outside the desired area of monitoring.

The first step of background subtraction is designed to remove stationary (or mostly stationary) objects, and acts upon each ladar's data individually. Thus, it can be run independently of the registration steps described above. Further, the algorithm considers each angle of each ladar individually, determining at what distance (if any) a stationary object appears at that angle. This is done by finding the distance at which the most hits occur over time.

Because the data is recorded to the nearest centimeter, while the accuracy of the ladar is slightly lower, some fluctuation is likely. To account for this, “buckets” are used to count the number of occurrences within a small range of measurements. For example, all data with a distance of between 100 cm and 110 cm could be counted together if a bucket size of 10 cm was used. It is important that the bucket size be large enough to account for noise in the data, but not so large that desired targets to be tracked would be subtracted while close to stationary objects. A bucket size of 5 cm was experimentally determined to be ideal.

Once all of the data for each scanning angle is sorted into the correct bucket, the buckets are examined for likely stationary objects. Starting with the bucket nearest the ladar and working outward, the contents of each bucket is expressed as the percentage of all laser hits. The first bucket with a percentage above a threshold (experimentally determined to be 25%) is considered to contain a stationary object. If no such bucket is found, then there is assumed to be no stationary object to be subtracted at that scan angle, and nothing is done to that angle’s data.

If, on the other hand, a bucket is found to contain this high percentage of laser hits, any data it contains can be subtracted as a stationary object. Further, any laser hits in subsequent buckets, thus farther from the ladar, can also be subtracted. This is because nothing beyond a stationary object can be “seen” by the ladar, implying that further laser hits are the result of noise, and can thus be eliminated. Because of noise at the edge of each bucket, subtraction actually starts one bucket closer to the laser than the one with the necessary percentage of laser hits. This entire process is repeated for each scan angle of each ladar.

Once all of the stationary background is eliminated, and the data has been registered in space and time, it is desirable to convert the data into "frames," consisting of data from all ladar at a given time, in Cartesian coordinates. These frames consist of a full picture at a moment in time, and are analogous to, though quite different from, video frames. After this conversion, the rest of the background data, consisting of all laser hits outside the immediate area to be monitored, is eliminated (based simply on x- and y-coordinates). This subtracts all data far away from the area, relying on the initial background subtractions to remove the stationary objects near or inside the area (such as the ladar devices themselves, which are "visible" to each other, and any bordering walls).

3.3 Models (Templates)

The purpose of the tracker is to determine the location of each target within the data. This is done by attempting to fit an instance of a model to the data. Such a model consists of a number of coordinate points, oriented in such a way as to approximate the appearance of the actual targets to be tracked. For example, the model of a person being observed by laser range finders placed at chest level would consist of a number of points forming a hollow oval shape, as this is the way a person would appear in the laser data, as shown in Figure 3.2. Only instances in which the data adequately conforms to the model

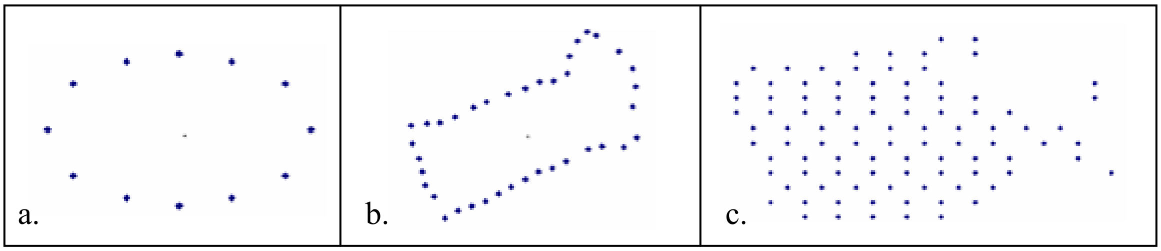


Figure 3.2: Several different models, not to scale. (a) A model of a person, as seen by a laser range finder. (b) A model of a person carrying a large rectangular box in front of them. (c) A 2-d model of a fish, as generated from video data.

are considered to be targets and tracked. In this way, noise (such as an incompletely subtracted background) can be prevented from impersonating an interesting target.

It is also possible to use multiple models. Using more than one model may be useful when it is necessary to track more than one type of target, such as several species of fish swimming in an aquarium. There could be one model for each shape and size of fish. Also, multiple models can be used when a given target can change shape. This may be caused by a change in perspective (e.g. a fish in two dimensions looks different head-on versus in profile) or when the targets can change states (such as a forklift which looks different when it is loaded than when it is not). By attempting to fit each model to the data, the tracker can determine which model best explains the data.

An instance of a model represents the location and orientation of a track. Generally, a track is considered a single point and could be considered to reside at the geometric center of a model. However, the actual pose of the model is maintained throughout tracking. This allows for the location of a specific part of a target to be known, in the case of an asymmetric model. Additionally, for such models, the actual orientation of the target can be determined.

3.4 Tracker

Once the data has been registered, background subtracted, and converted to Cartesian coordinates, it is tracked. A track represents the location of a single target over time. Determining the correct tracks is challenging for a variety of reasons. Sometimes the background is not fully removed or a target is (partially) occluded. Both of these situations result in difficulties identifying the "interesting targets" in a given frame. Further, the data association problem, the ability to correctly associate a given target with itself over time, is especially difficult when multiple targets are in proximity to each other or moving quickly.

The goal of the tracker is twofold. First, it must determine which groups of laser hits in a given frame correspond to one of the targets to be tracked (as opposed to non-subtracted background or noise). This is accomplished by fitting a model to each grouping of data points; the target's pose corresponds to the location and orientation of the model. Second, the tracker must recognize these groups of data points from frame to frame in order to build tracks representing the same target over time. This tracker accomplishes these goals in parallel, using the information about the clusters found in one frame to help find the corresponding cluster in the next. Because the tracks will later be paired up with the target which is responsible for the data, it is important that a single track only represent a single target – if the track "jumps" from one target to another, the track cannot be entirely correctly identified. On the other hand, it is also important that the tracker generate tracks which are as long as possible, in order to assist in the target assignment during a later step (see Chapter 4).

The tracker has two main elements. The first component, track generation, uses the pose of the models in previous frames(s) and iterates on a given frame to find all valid placements of models in the current frame, updating existing tracks then adding new instances of models to account for any remaining data. The second part, is the track splitter. It is responsible for splitting any tracks which are too close together to be accurately separable, preventing any potential track "jumps".

Track Generation

After registration and background subtraction, the tracker must identify the locations of each target within the remaining data. This can be thought of as a two step process. First, any existing tracks are updated to reflect their new location. Then, new tracks are looked for among any remaining data.

The first step in updating the existing tracks is to adjust the location and orientation of each track based on the previous velocity. For instance, the starting position of a track at t=2 would be found by calculating a vector between its locations at t=0 and t=1, then adjusting the t=1 position by that vector. The vector would include not only magnitude and direction of the location coordinates, but also the rotational changes of the model representing this track between t=0 and t=1. The benefit of this initial adjustment is that it allows for smaller distance requirements between the model points and the data point than would otherwise be possible – without this update step, a target is more likely to move too far away from its previous location, resulting in being identified as a different track. Smaller distance requirements are useful to help prevent a track from jumping from one target to another.

After the track location is updated based on velocity, all the data points within a certain distance of the center of the model are examined. This distance is dependent on the scale of the data and the size of the targets. For humans in an environment the size of a basketball court, an appropriate distance was experimentally determined to be 1 meter. All data points within range are potentially part of the target represented by the current track. While some component data points may fall outside this range, the likelihood is small and the exclusion of many distant points can greatly improve the speed of the algorithm.

Each model point is paired with the nearest data point (as shown in Figure 3.3). Iterative closest point (ICP) is used to determine the transform of the model points which minimizes the distance between each model-data point pairing. The model is adjusted accordingly, and then each point is again paired with the nearest data point. ICP again transforms the model points to better fit with the data points. This cycle is repeated until the pairings do not change after an ICP adjustment. Now that the model is at the final location, two tests are performed to determine if the track is considered to exist during this frame. First, the fit is calculated as the sum of the distances between each of the final pairs. If this (normalized) fit is outside of a threshold, then the data is determined to not adequately reflect the appearance of a target and the track is removed. Finally, the distance between each of these nearby data points and the nearest model point is calculated. All data points that are within a certain distance are added to a list. If the list

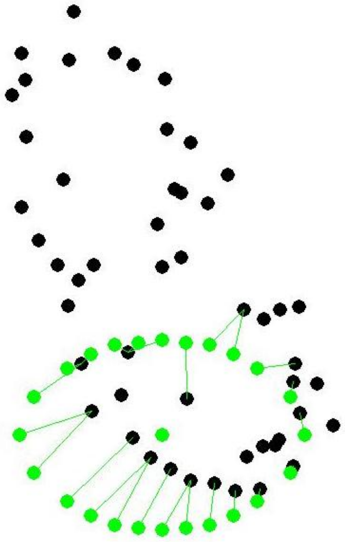


Figure 3.3: Fitting a model (green/grey dots) to data (black dots). In this step, each all model points are paired with the nearest data point.

is long enough (i.e. if there are enough data points very close to the model points), then the track kept; otherwise, is it removed. Both of these measures help prevent noisy data from generating extra tracks. Whether the track is ultimately kept or not, the data points making up the final list of very close points are removed from further consideration.

If the tracker is processing data with multiple models, all of these steps are repeated for each model. Once every model has been updated, the model with the best fit (as calculated above) is noted as the most likely model, the track is kept or not based on its parameters, and its list of nearby data points is removed. This entire process is reiterated for each existing track.

After all existing tracks are updated the remaining data points must be examined for new tracks, representing targets which were not tracked in the previous frame. First, a data point is chosen at random. An instance of the model (or models) is centered at this data point. From this point, the algorithm proceeds as with existing tracks, starting by pairing each model point with the nearest data point and using ICP to find the best transform. The only other difference between updating existing tracks and finding new ones is that new tracks require a larger number of very close data points in order to be kept – this is to allow known tracks to be partially occluded without being lost while still preventing small amounts of noise from being wrongly identified as tracks. The final results of four subsequent sample frames are illustrated in Figure 3.4.

The tracking algorithm can be described as follows:

Steps in tracking algorithm:

```
for each existing track
  call UpdateTrack(list of unused data points, current model location)
```

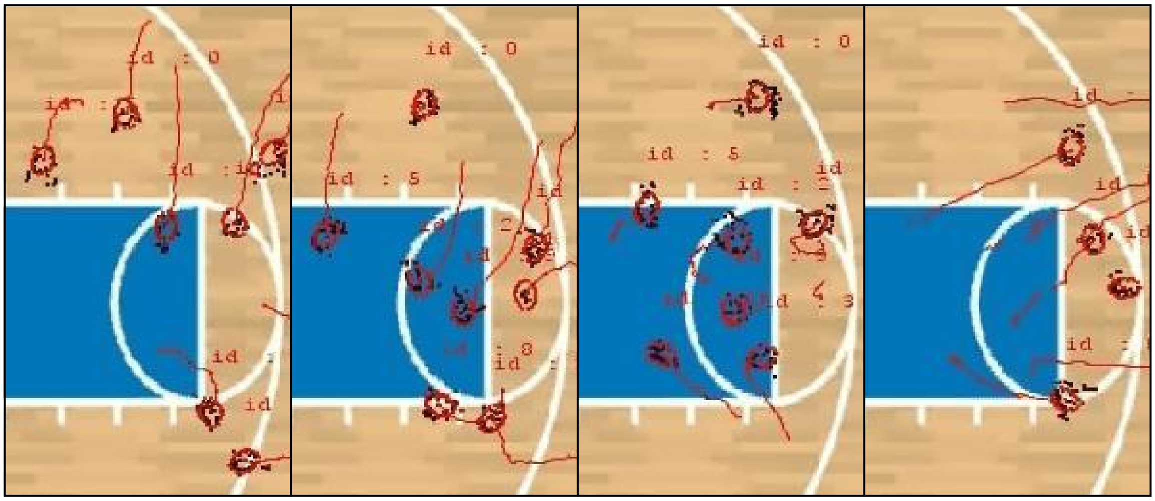


Figure 3.4: Results of processing 4 frames, each about 1 second apart. Black dots represent laser data. Red/grey dots are model instances placed at track locations. Trails show past trajectory. Note one spurious track in the 3rd image.

```
  remove all data points near the updated model points
  if #removed points < (minimum number of points / 4) || model-fit is too low
    remove this track
  while there are remaining data points
    call UpdateTrack(list of unused data points, first data point location)
    remove all data-points near the updated model points
    if #removed points > minimum number of points && model-fit is not too low
      create new track at this location


UpdateTrack(unused data points, current model location):
  while (model point, data point) pairing list changes
    call ICP to update model location
    update (model point, data point) pairing list based on new model location
  return(updated model location)
```

Track Splitter

One of the goals of the tracker is to ensure that a single track only represents a single target over its entire existence. This is because the tracks will later be associated with a target; if the track jumps from one target to another, then it will be impossible for the entire track to be correctly labeled. Therefore, it is crucial that track jumps be avoided. Unfortunately, there are some situations in which the underlying laser data of two nearby targets becomes ambiguous, resulting in uncertainty over which track belongs to which target. In these situations, the best the tracker can do is to split the two tracks into two "before ambiguity" and two "after ambiguity" tracks. This way, there is no chance of the tracks switching targets during the indistinctness. Additionally, during the uncertainty, the two tracks are replaced by a single track, located halfway between them. This denotes that the targets are so indistinct as to effectively merge into a single track. Therefore, the two tracks are split into a total of five distinct track segments.

The effectiveness of this technique is based on the distance at which two tracks must be in order to perform the necessary splitting. At one extreme, all potential track jumping can be eliminated by setting the split distance very high. However, this will cause frequent splits, resulting in much shorter tracks. Yet, another goal of the tracker is to generate tracks which are as long as possible, which will also help with track/target assignment. Therefore, a moderate split distance must be used, acting as a balance between track length and likelihood of track jumping. For humans, split distances of roughly 0.5 m (experimentally determined) are ideal with slightly lower values better when the targets move slowly and do not completely run into each other (track jumps are less likely in these situations, so track splits are less important).

3.5 Methods

Two sets of data are used to assess the tracking system's accuracy. Both datasets were gathered with 8 laser range finders placed around the perimeter of the area of interest, a basketball court. Each consists of a group of people moving around and interacting on the court in various ways. The first dataset includes 10 individuals playing a 5 on 5 pickup basketball game which lasts for approximately 16 minutes. In the second dataset, 25 people were asked to walk and run around, following a pre-described script outlining various social behaviors to perform; the duration is 9 minutes. The datasets each provided their own set of challenges. For example, while the basketball game has fewer targets (reducing occlusions), the targets generally move much faster and tend to interact in closer quarters than occur in the social behavior experiment. In addition to these test datasets, the best model and parameter values are determined using two training sets, consisting of a short (3 minute) section of the basketball game and a completely separate 9 minute dataset of the social behavior experiment.

Accuracy of the tracker is assessed in three ways: detection accuracy, average track length, and number of track jumps. The tracker's performance across these three metrics indicates how well it fulfills its stated goals. There are three main parameters which can be tweaked in order to adjust performance on one or more of these metrics. The first parameter, *maximum point distance*, is the maximum distance allowed between a data point and the nearest model point; it is used in the determination of which data points belong to which track. Next, the minimum number of points necessary for the creation of a new track is the *minimum points per track*. Finally, *split distance* is the distance inside of which two tracks are split. These parameters are dependent on the experimental set up (number and size of targets, typical distance from targets to ladars, and the number of ladars present), and should be tweaked as needed to optimize the tracker's performance, though in some cases, increases in one metric results in the decrease of another.

Finally, the system's ability to function at real-time on live data is examined. This test examines how well the tracker can perform when required to keep up with an incoming data stream. For example, if the tracker cannot function at the full rate that the data is being generated, then how is performance degraded by only processing as much data as possible?

Detection Accuracy

This metric is designed to assess the tracker's ability to detect the location of each target in each frame. It represents the fraction of total targets correctly found in each frame, and is expressed as the percent of "track-frames" found. A track-frame is defined as an instance of a single track in a single frame. Therefore, for example, a dataset containing of 5 frames, with 10 targets present in each frame, would consist of 5 * 10 = 50 track-frames. If the tracker only fails to detect one target in one frame, it would have a detection accuracy of 49/50 = 98%.

To determine which track-frames are correctly detected, the ground truth target locations are manually defined in each frame. Then, an attempt is made to match each ground truth track-frame to the nearest automatically detected track (in the same frame). If a match is found within 0.50 meter, then that ground truth track-frame is considered to have been correctly detected. It should be noted that a single detected track could match to multiple ground truth tracks if they are close enough together. This is allowed because of frames in which the track splitting module joined two nearby tracks – the single remaining track actually represents both targets during its entire existence. As such, it is possible to know when tracks represent two targets, and they could be marked accordingly.

Of the three parameters, the maximum point distance and minimum points per track have the largest effect on the detection accuracy. For example, decreasing the minimum points per track can result in the creation of multiple tracks per target, which will reduce the model fit and cause valid tracks to be eliminated. On the other hand, if the minimum points per track is set too high, then some targets may not be tracked at all (especially those farthest from the sensors or partially occluded). Likewise, adjustments to the maximum point distance can have similar effects.

Average Track Length

The second metric used to assess the quality of the tracks generated by the tracker is the average length of all detected tracks. This is important because many potential uses of the tracks rely on long tracks. For example, the system of determining track/target pairings described below uses RFID readings which are only broadcast every 2-2.5 seconds. As such, any tracks shorter than this are not guaranteed to be present for any RFID readings, while tracks somewhat longer receive only sparse readings. Therefore, it is important for the tracks to be as long as possible. The average track length is simply the sum of all detected track-frames divided by the number of tracks, expressed in seconds.

Although the removal of tracks shorter than 1 second will slightly increase the average track length (as compared to keeping them), the loss of these tracks will, in turn, lower the detection accuracy. Such effects are minor, but demonstrate one way in which the evaluation metrics are interconnected. It is important to optimize all of the metrics together, instead of only considering one at a time.

The main parameter which affects the average track length is the split distance. Decreasing the split distance increases the track length, but at the peril of increasing the number of track jumps (discussed below). Because adjusting the split distance affects both average track length and number of track jumps, unlike the primary detection accuracy parameters, changes to this parameter require examining both metrics to find the best value.

Track Jumps

The phenomenon of track jumping refers to instances of a single track segment representing multiple targets throughout its existence. This generally happens when two targets pass very close to one another, such that the track in question shifts from corresponding to the data points from one target to the data points of another target. Therefore, this metric counts the number of tracks which suffer from at least one track jump.

To detect instances of track jumping, the first step is to sum the distance between a data track and each ground truth track across all of the track's frames. The ground truth track with the lowest total distance is said to be the corresponding track. If this corresponding track has an average distance (total distance divided by the number of frames) of greater than 0.5 meters, then it is likely that a track jump occurred. Alternatively, if the distance between the data track and the corresponding track is greater than 2.0 meters in any individual frame, it is also likely that a track jump occurred. Each data track that suffers from either or both of these conditions is considered to have undergone a track jump, thus incrementing the number of track jumps in the dataset. The total number of track jumps reflects the number of tracks that have at least one jump – the metric does not determine the total number of times a given track jumps; once a track jumps once, the damage is done.

Similar to average track length, the parameter that has the largest affect on this metric is the split distance. As expected, the greater this distance, the less likely tracks are to jump from one target to another, because track jumps only occur when tracks are very close together. On the other hand, too large of a split distance will result in exceedingly short tracks. Therefore, a balance must be found.

Real-Time Tracking

Finally, the real-time performance of the system is examined. As data is read from the sensors, it is immediately background subtracted and registered (with previously obtained values), then given to the tracker for processing. The results (i.e. the locations of each track in this data) are returned and immediately passed on to whatever service will use the tracks. Currently, for this experiment, the tracks are simply logged for later analysis.

The module responsible for splitting nearby tracks is designed as a batch process which operates on entire tracks after they have been completely created. As such, it does not function in real-time mode. However, it could be re-implemented to work with tracks as they are being generated. Therefore, results of real-time tracking are examined both with and without running the track splitter. Additionally, the speed of the track splitter is considered, to determine the likely effect it will have if built directly into the tracking process.

In order to allow a comparison between live tracking and tracking pre-logged data, the live tracking is simulated by using the raw logged data described above. A module reads in this data at the rate that it would be read from the sensors. If the tracker is not ready for the next frame of data by the time it is read in, it would be discarded and the next frame made available. In this way, the tracker constantly receives the "current" data, regardless of how long tracking takes. Therefore, the tracker was not allowed to fall behind. On the other hand, if the tracker completes processing the current frame before the next frame is read in, the data is read in immediately. This way, if the tracker can process data faster than the sensors would provide it, its exact speed can be determined.

In addition to examining the rate at which the tracker can process data (expressed in frames per second), performance is evaluated similarly to the off-line version of the tracker. After all data is tracked and logged, the tracks are examined for number of track swaps, average track length, and the percent of track-frames detected. The first two metrics are the same as above, but the third is calculated slightly differently for this experiment. Because only a subset of frames are processed, and there is no specific temporal synchronization applied, it is difficult to compare these tracks with the hand-labeled, ground truth tracks for these datasets. Therefore, the percent of track frames found are estimated as the sum (across all frames) of the difference between the expected number of tracks (10 or 25) and the actual number of tracks. For instance, in two frames of basketball data, the expected number of tracks in each are 10; if there are 9 tracks detected in the first frame and 10 tracks detected in the second, then 19/20 or 95% of track-frames were detected. In this test, both live and off-line tracks are assessed with this detection estimation metric, which has been used by other trackers, such as Balch et al [2001].

3.6 Results

Models and Parameters

To generate the best possible results, both the model(s) and parameters must be varied. In the case of two parameters, maximum point distance and minimum points per track, the same best values apply to both training datasets. On the other hand, the differences between the two scenarios are such that the model and split distance which results in the best tracking results are different.

Figure 3.5 shows the models that are used to perform tracking in the basketball game and social behavior experiment, respectively. Note that the model for the social behavior experiment, in which people generally move slower, consists of a smaller oval. This is because the effects of rounding the ladar data to the nearest 26.67 ms is reduced when movement is slower, resulting in a tighter grouping of data points representing each target. Conversely, the consistent high speed of the basketball players cause the temporal offset to shift the data points from each laser noticeably (up to 18 cm for targets traveling at 15 mph). Additionally, the basketball player model includes more points because the players were typically much closer together (even colliding frequently) than the social

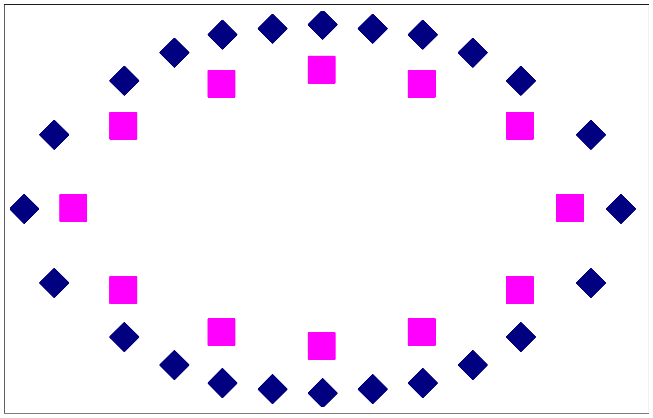


Figure 3.5: Models used in the experiments, diamonds for the basketball players and squares for the social behavior experiment participants. The models are to scale, with the larger model measuring 0.6m wide by 0.4m high.

behavior experiment participants, necessitating more model points to prevent a track from being pulled into the center of two targets.

Like the models, the best split distance is also different for each type of dataset, with the basketball data requiring a higher value (0.6 m, compared to 0.4 m). As previously stated, the basketball players were more prone to fast, close movements, resulting in less model-like data point distributions (due to the temporal offset). Thus, the tracks are more prone to jumping, requiring a higher split distance to combat the effect.

The parameters for maximum point distance and minimum points per track are affected less by the ways in which the targets move than they are by inherent constraints of the environment. Specifically, these parameters are most affected by the number of sensors used, the size of the targets being tracked, and their rough distance from the sensors. All of these factors affect the number of laser hits which will strike a target. The number of sensors and the target's distance from each also dictates how far apart the laser hits will occur, affecting the maximum distance between data points and model points. Therefore, in all experiments, 25 minimum points per track (with 25% as many required for existing tracks) and 0.2 m maximum point distance produce the best results.

Table 3.1 shows a summary of the tracking results. Included are both test data sets and the tracker's performance with regards to each metric. Below is an analysis of the results.

Table 3.1: Summary of results of the tracker on two datasets.

Dataset	Total Track-Frames	Avg. Track Length	Track Jumps	Detected Track-Frames
Basketball Game	366,196	39.81 seconds	5	360,443 (98.43%)
Social Behavior Experiment	496,810	339.57 seconds	2	496,307 (99.90%)

Detection Accuracy

The tracker achieved a detection accuracy of 98.43% of all track-frames in the basketball data. Most of the missing track-frames are due to either temporarily losing track of occluded players in the center of a multi-person "huddle" or the deletion of a number of short (less than 1 second) tracks which result from the middle segment created in track splitting. The tracker performed even better on the social behavior experiment data, achieving 99.10% of all track-frames detected. This dataset proved slightly easier, despite the increase in targets, due to the participants remaining more spread out than the basketball players.

These results compare favorably to vision-based tracking. For example [Balch et al 2001] achieved an 89% accuracy examining a similar metric (in which accuracy was a measure of the number of tracks detected in each frame, compared to the actual number of targets present) with a vision-based tracker applied to a number of ants in an arena.

Average Track Length

The average track length of the basketball players' tracks is 39.81 seconds, while tracks for the slower moving social behavior experiment participants are an order of magnitude longer at an average of 339.57 seconds. These results compare favorably to earlier versions of this tracker, which never surpassed an average track length of 10 seconds [Feldman et al 2007].

Track Jumping

Applying the track splitter after tracking reduced the number of track jumps in both datasets. Specifically, there are only 5 track jumps in the basketball game, or 2.07% out of a total of 242 tracks. The social behavior experiment also succeeds in this regard, with only 2 track jumps out of 39 tracks, or 5.13% of all tracks. It would be possible to eliminate some of these track jumps, but the associated reduction in average track length actually proves more detrimental to the track/target association phase (as described in Chapter 4) than the few existing track jumps. For example, by increasing the split distance until both of the social behavior experiment track jumps are eliminated results in average track length decreasing by nearly a factor of 10.

Real-Time Tracking

The tracker was evaluated when presented with data at a rate equal to or faster than would be gathered by the sensor (i.e. 37.5 frames per second). The basketball data can be tracked at 39.12 frames per second. That is, the tracker processes data even faster than it would be generated by the sensor. On the other hand, the social behavior experiment data is only tracked at 28.05 frames per second. The discrepancy is due to there being two and a half times as many targets in the latter dataset. There would be an even larger difference in the processing rate if not for the reduced number of data points in the model used by the social behavior experiment, as the running time is proportional to these two factors. Therefore, for datasets with more targets, a higher frame rate can be achieved by reducing the number of data points in the model(s).

Each dataset was evaluated both live and off-line, and with and without also using the track splitter as a post processing step. The track splitter (as currently implemented) only runs as a batch process, but is very quick, able to process over 700 frames per second, or about 1.5ms per frame. As such, even if it were made no more efficient for live use, it would only reduce the frame rate of the tracker by about 5%. This would have no effect on the basketball data (which would still have a frame rate above the sensors' rate) and only a decrease of 1-2 frames per second on the other dataset.

Table 3.2 shows the quality of the tracks generated in each configuration. For the basketball data, in which only a few frames of data are lost, the results are almost identical between the live and off-line tracking. Even though 25% of the frames are discarded in the social behavior experiment dataset, the tracker performance is almost as good, with the only major difference being a couple more track jumps. Therefore, the tracker can successfully track at least 25 targets live as the data is gathered with little degradation in track quality due to frame rate decreases.

On the other hand, most vision trackers cannot track in real time with a high level of accuracy. For example, Balch et al [2001] can locate ants at the rate of 24 frames per second, but requires additional time to perform the data association step necessary to create individual tracks over time. This tracker does not require such a step, as data association is performed in concert with the detection of tracks.

Table 3.2: Summary of results of the tracker on two datasets in real-time. Included are both real-time and off-line results, including with and without using the track splitter.

Dataset	Live?	Split?	Avg. Track Length	Track Jumps	Detected Track-Frames
Basketball Game	Yes	No	59.87 seconds	52	99.36%
Basketball Game	No	No	60.69 seconds	52	99.38%
Basketball Game	Yes	Yes	40.13 seconds	6	98.69%
Basketball Game	No	Yes	39.81 seconds	5	98.59%
Social Behavior Exp.	Yes	No	339.36 seconds	5	97.94%
Social Behavior Exp.	No	No	308.00 seconds	3	98.03%
Social Behavior Exp.	Yes	Yes	339.32 seconds	4	97.94%
Social Behavior Exp.	No	Yes	339.54 seconds	2	98.02%

3.7 Discussion and Summary

This chapter presents an algorithm used to produce tracks which fulfill the goals introduced in Section 3.4. Specifically, tracks average 40 seconds in the high-speed, high-impact basketball game and over 5 minutes in the slower moving (but more crowded) social behavior experiment. This means that, on average, a track is lost and re-initialized (or split) every 5-10 seconds. At that rate, a human labeler would have no problem assigning labels in (near) real-time, resulting in useful tracks, even in situations in which the RFID techniques presented in Chapter 4 cannot be used. Also, track jumps are rare, occurring only once every several minutes.

One important contribution of this work is the ability to track a varying (and unknown) number of targets moving in a single plane. The introduction or removal of targets in the middle of tracking does not add any complexity to the algorithm. Further, although the runtime is proportional to the number of targets, at least 25 targets can be efficiently tracked at a frame rate capable of producing high quality tracks from a real-time data stream. This is unlike most vision trackers which run slowly and/or require pre- or post-processing steps to remove the background or perform data association.

CHAPTER 4

RFID BASED TRACK/TARGET ASSOCIATION

Laser range finders provide excellent localization, but there is no way to associate tracks with the specific targets which the data represents using lasers alone. To account for this, a second sensor can be incorporated. Therefore, this chapter introduces a technique to use such a sensor to label the tracks generated in Chapter 3. Specifically, active RFID tags were chosen for the task of associating tracks with the targets they represent. RFID tags are a logical choice, as they provide completely unique signals (each has its own "serial number") and have a range comparable to the laser sensors. However, they also have two notable problems. First, they only send a signal every 2-2.5 seconds, reducing the number of data points they can provide, hence the reason for the tracker to strive for the creation of long tracks. This first problem can be further mitigated by placing multiple tags on each target. The second and more troublesome problem is the noisiness of the signal strength readings which are the only means of localizing the tags. It is this second problem that needs to be solved to make the RFID tags a useful addition to the system.

Ideally, the signal strength of a tag reading received at an antenna should be relatively deterministic based on the linear distance from the tag to the antenna. If this were the case, approximate tag locations could be determined by imagining concentric circles, centered at each antenna. The tag would be in the region formed by the overlap of the correct circles, based on the signal strength of the reading at each antenna. For example, Figure 4.1 shows an arena with two antennas. If the signal strength of 69 is received by antenna #4 and 82 is received by antenna #7, then the tag would be found in

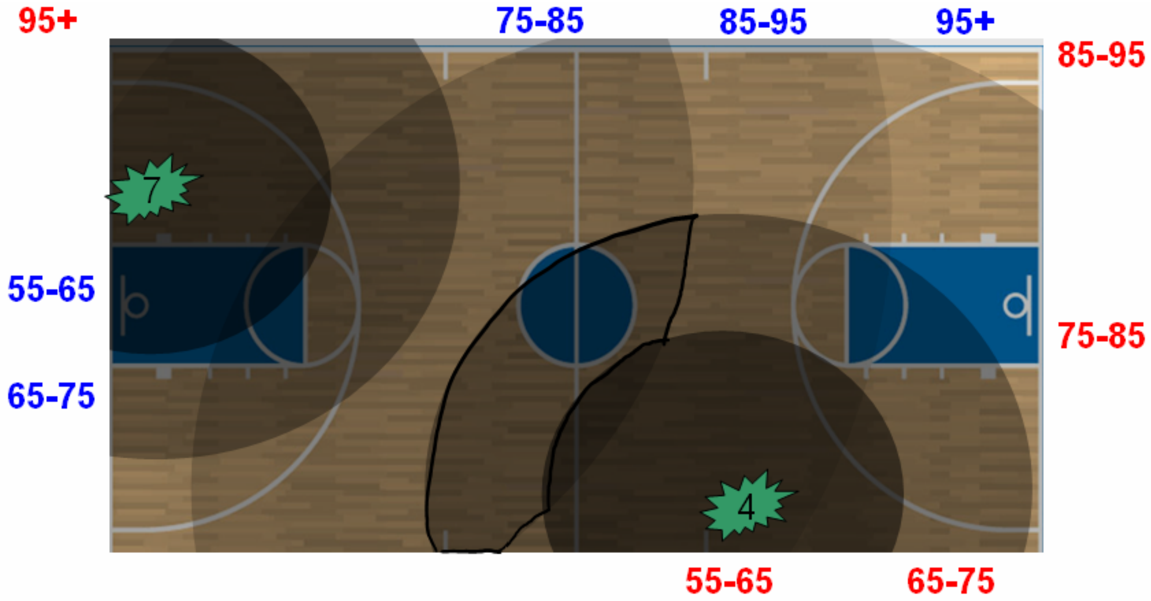


Figure 4.1: The region in which a tag is located could be determined by the signal strength received by one or more antennae. Circles and numbers correspond to antennae #4 and #7.

the region bounded by the black line. The addition of more antennae would reduce the size of that region. Unfortunately, the level of noise present in the signal strength readings result in very convoluted shapes emerging when signal strength is plotted against tag location. For example, Figure 4.2 shows the locations in which antenna #1 reads a signal strength of 76 in the training data. Therefore, RFID readings will not be used to generate approximate target locations which can then be used to augment other sensors in track generation.

The laser range finders are much more accurate at localization than any RFID system, so the tracks are generated solely from the laser data, with the RFID data only being used to label each existing track with the most likely target represented by the data. The technique works by building lookup tables of known tag locations versus recorded signal strengths at each of the antennae. Then, when new tag data is recorded, the lookup tables are consulted to determine how likely the tag (and thus the target) is to be in the

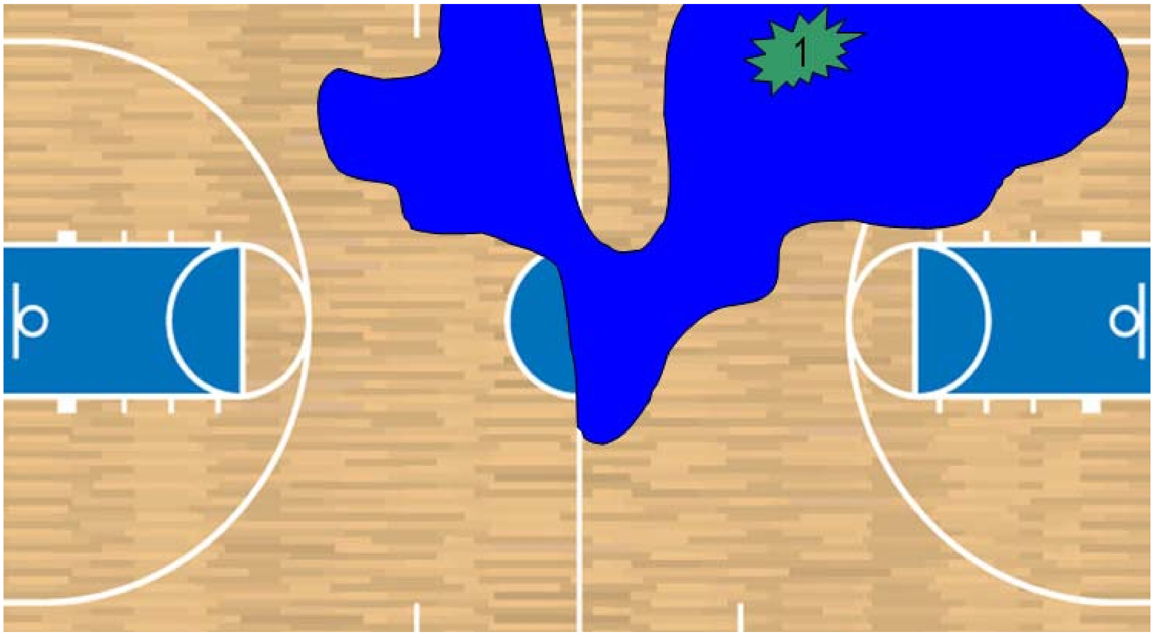


Figure 4.2: The region at which this antenna received readings with signal strength of 76 in the training data.

vicinity of each of the known laser-based tracks. This likelihood is updated over time, until a score is created for each track/target pairing. The final pairings are then assigned in an order based on the relative confidence of each track's scores.

4.1 Building the Lookup Tables

Instead of attempting to represent exact topologies with the RFID readings, coarse lookup tables are used to contain the training data. First, the area of interest is divided into a grid with cells measuring 2.6 meters by 2.6 meters (experimentally determined). For the rest of this process, instead of using actual x/y coordinate information to represent a location, the relevant grid cell will be used. Each lookup table consists of a single such grid, and will correspond to one signal strength/antenna pairing. Therefore, the number of lookup tables is the number of antennae times the number of possible discrete signal strength values. Then, to perform a single lookup, queries such as Lookup(antenna a, signal strength ssi, x-cell xc, y-cell yc) are used to retrieve the correct lookup table and then find the value in the correct cell of the table. This value corresponds to the likelihood of a receiving a reading of ssi when the tag is actually located in grid cell (xc, yc).

Initially, the value of every cell in every table is set to 0. Then, training data is used to increment the values of the correct cells. This training data consists of RFID readings (signal strength at each antenna) and tag locations. It is gathered when a single individual slowly walks around the environment to be tracked. Since this individual is the only target in the environment during training, laser range finders can determine the true location of the tag at every moment in time – the tag is simply at the location of the only track. Therefore, it is unnecessary to manually label any training data to complete this process. It is important that the training data cover the area of interest completely and slowly, so as to generate many readings spread out among all grid cells. Once the training data has been gathered, the lookup tables are incremented as follows. For a given training data point, the grid cell of the tag is determined and the value of this cell in the lookup table for each antenna/signal strength pairing found in the data is incremented to denote that there was one additional occurrence of this antenna receiving this signal strength while the tag was at this location. In this way, each training data point is responsible for incrementing one value per antenna it represents. After all of the training data has been processed, each lookup table is individually normalized by dividing by the sum of all the values of all the cells in that particular table. This results in each lookup table representing the probability distribution of tag locations for that antenna/signal strength pairing.

4.2 Scoring Each Track/Target Pairing

Once the lookup table values are set from the training data and normalized, they can be used to generate scores representing the likelihood that a given target/track pairing is correct. The score of a pairing is based solely on the track location during each of the RFID readings for the target's tags. Therefore, each pairing is scored independently of the other tracks and the other target information, although these other factors are considered when actually assigning the final pairing (see Section 4.3). The score represents how likely this track/target pairing is to be correct.

To determine the score of an individual track/target pairing, the lookup tables are consulted with regards to each of that target's RFID readings. For each reading, retrieve the lookup tables corresponding to each antenna/signal strength combination. The table cell of interest is the cell corresponding to the grid location of the track at this time. Increment the score by the sum of the values in the appropriate cell of each lookup table. Continue increasing the score in this way for each of this target's tags' readings. After all RFID readings have been used, normalize the score by dividing by the number of readings thus examined, to prevent scores from being skewed just because one or more readings were recorded for a given tag.

The above process generates the score for a single track/target pairing. It must then be repeated with respect to all pairings. This will result in a total number of (tracks * targets) scores. The final pairings can then be decided using these scores.

The algorithm can be described as follows:

Steps in track/target scoring algorithm:

```
 for each track
```

```
for each target
  for each of this target's RFID readings
    retrieve the lookup tables for each antenna/signal strength combination
    calculate the grid cell of the track's current position
    increment score by the sum of the correct cell in each retrieved lookup table
  normalize score by dividing by the number of RFID readings
```

4.3 Assigning Final Track Labels

The last step in determining the correct track/target pairings is to use the previously generated scores to find a labeling scheme which heuristically maximizes the sum of the selected scores. Because this refers to the physical locations of the targets, the labeling process can benefit from the fact that it is not possible for one target to be in multiple locations at the same time. Therefore, once a target label is applied to a track, no other coexisting track (even if only coexistent for a short time) can use that label. This makes the order of labeling very important, as each assigned label restricts the choices available to the other tracks.

To decide the order in which to label the tracks, a confidence is calculated for each one. This confidence is the product of the track length squared times the difference between the two highest available scores. The confidence represents a measure of the assurance that the highest scored label is correct. The track length is included because longer tracks benefit from existing during more RFID readings. By using the difference between the two highest scores, the algorithm is capturing the relative cost of using the second best label, similar to the process used in the Hungarian algorithm [Kuhn 1995].

For example, tracks that have a large difference between the highest and second highest scores are sacrificing more "likelihood of correctness" than tracks in which the two highest scores are more similar.

After all the confidences are calculated, the track with the highest is chosen for assignment. The target label that is assigned to this track is the one which has the highest score. As soon as this assignment is made, all unlabeled tracks are updated; any unlabeled, coexistent track has its score for the chosen label reduced to 0. Then, all the confidences are re-calculated, to take into account the new scores. Again, the track with the highest new confidence is labeled next, according to its highest scored target. This process is repeated until all the tracks are assigned a label. In some cases, it is possible that there is no valid assignment for a track (for instance, there may be an extra track which does not correspond to any of the targets wearing RFID tags) – if this occurs, a track with no available targets would be labeled as "UNLABELED." Table 4.1 gives a simple example of this process.

The algorithm functions as follows:

Steps in track/target assignment algorithm:

```
 while there are unlabeled tracks
   calculate confidence for each unlabeled track (length$^2$ * (best score – 2$^{nd}$
     best score))
   apply the best label to the track with the highest confidence
   if all labels have score of 0
     apply label "UNLABELED"
   else
```

```
    reset scores for this label to 0 for all coexisting tracks
```

Table 4.1: The upper left table shows the initial track/target scores and track confidences. In the upper right table, Track 2 has been labeled as Tag A, and the confidences have been recalculated after zeroing the scores for Tag A in Tracks 1 and 3. Then, the lower left table shows that Track 1 was labeled with Tag C, removing this option from Track 3. In the last table, the algorithm finishes by assigning Tag B to the final track, Track 3.

	Length	Tag A	Tag B	Tag C	Confidence
Track 1	1720	0.35	0.07	0.33	59,168
Track 2	907	0.27	0.15	0.12	98,717
Track 3	1251	0.40	0.44	0.38	62,600

	Length	Tag A	Tag B	Tag C	Confidence
Track 1	1720	0	0.07	0.33	769,184
Track 2	907	0.27	0.15	0.12	Tag A
Track 3	1251	0	0.44	0.38	93,900

	Length	Tag A	Tag B	Tag C	Confidence
Track 1	1720	0	0.07	0.33	Tag C
Track 2	907	0.27	0.15	0.12	Tag A
Track 3	1251	0	0.44	0	688,600

	Length	Tag A	Tag B	Tag C	Confidence
Track 1	1720	0	0.07	0.33	Tag C
Track 2	907	0.27	0.15	0.12	Tag A
Track 3	1251	0	0.44	0	Tag B

4.4 Methods

To evaluate the quality of track/target assignments generated by the system, two datasets are used. The origin of these datasets is the same two experiments previously described: a 16 minute, 5 on 5 basketball game, and a 9 minute social behavior experiment involving 25 participants. The actual tracks are those automatically generated by the tracker described above. In addition, the technique was tweaked using a separate 9 minute experiment and a 3 minute subset of the basketball game. Finally, the data used to create the lookup tables consists of a single target walking around the basketball court in a grid-like fashion. This data is about 46 minutes long and includes 2369 RFID readings. The trajectory is shown in Figure 4.3.

In the case of both the training data and the actual experimental data, the participants wore hats on which were affixed two RFID tags, as shown in Figure 4.4.

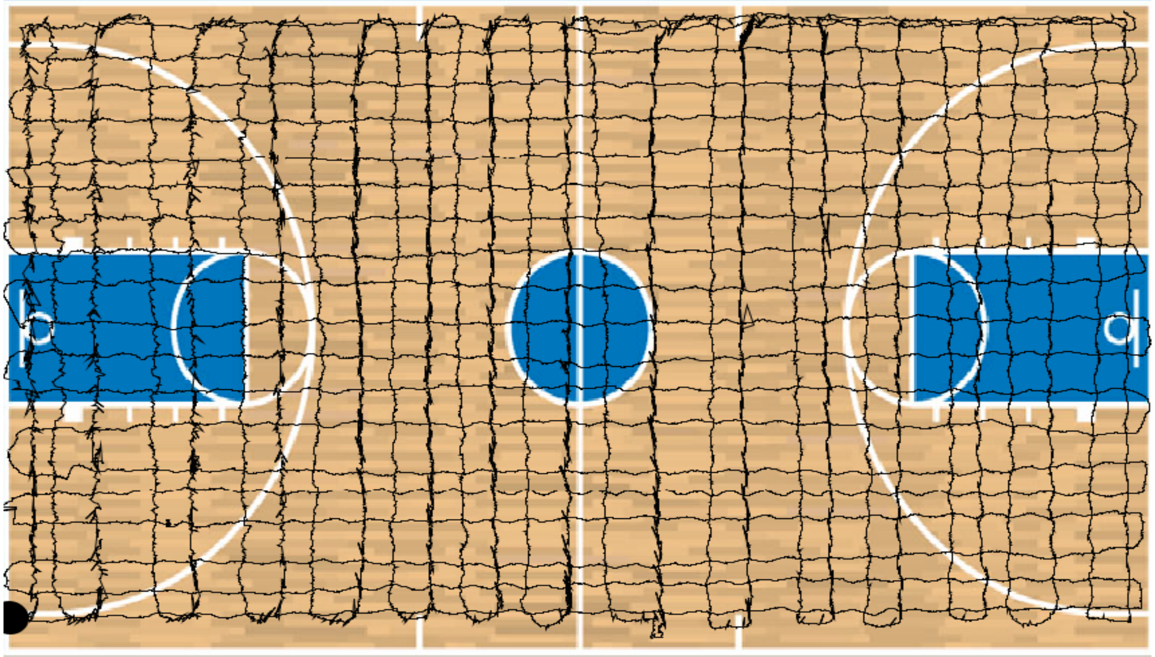


Figure 4.3: Black lines show the trajectory of the lookup table training data, as generated automatically by the laser range finders.

Using two tags effectively doubled the rate of readings, providing a richer data set. The tags were attached to the relatively steady surface provided by the top of the hat in order to help reduce the signal strength variability caused by changes in orientation. Although participants could be facing in any direction, the hat kept the tags generally fairly perpendicular to the ground, reducing their degrees of freedom. Each tag was color coded (and the entire experiment was videotaped) only to assist with human labeling of ground truth for evaluation.

Evaluation is similar to the detection accuracy metric described in the track methods above. However, instead of comparing the distance between a ground truth track and the nearest data track to see if it was detected, it is only compared to a data track with the same target label. If such a track is too far away (greater than 0.5 m), or if there are no tracks with the correct label in this frame, this track-frame is considered

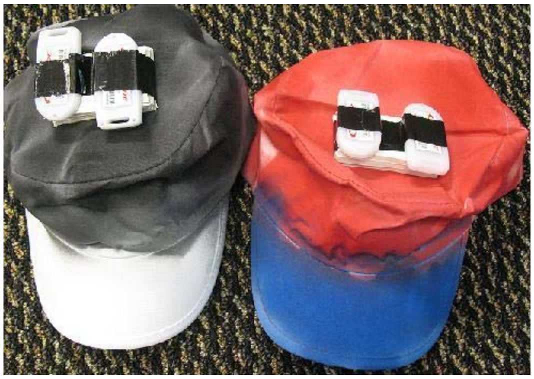


Figure 4.4: Two examples of the hats worn by the participants during experimental data collection.

incorrect. Therefore, the accuracy reflects the fraction of track-frames in which a target has an associated track which is correctly located and identified.

4.5 Results

Table 4.2 summarizes the results of the target association algorithm on the two test datasets. The original tracker results from above are also repeated. As the table shows, the percent of track-frames which are correctly identified in the basketball game is 95.9% of all track-frames, or 97.5% of the detected track-frames. Performance is also strong on the social behavior dataset, with 90.2% of all track-frames accurately detected and labeled.

Table 4.2: Summary of results of the tracker (with RFID identification) on two datasets.

Dataset	Total Track-Frames	Average Track Length	Track Jumps	Detected Track-Frames	Identified Track-Frames
Basketball Game	366,196	39.81 seconds	5	360,443 (98.43%)	351,262 (95.9%)
Social Behavior Experiment	496,810	339.57 seconds	2	496,307 (99.90%)	448,188 (90.2%)

Despite the social behavior experiment proving easier to track, the basketball game actually has a higher identification accuracy. An examination of the errors in identification of the social behavior experiment reveals that about 80% of the wrong track-frames are due to two tracks having reversed labels – unfortunately, these tracks both persisted for the entire dataset, resulting in 18 minutes of reversed track-frames. The people which these tracks represent spent the entire experiment in close proximity, roaming the arena side by side the whole time. Thus, they scored very similar to each other on all labels, with the wrong label just edging out the correct one. This is an example of long tracks actually being a hindrance, for, although the tracks never jumped targets, if they were split, even occasionally, they probably would have been labeled correctly at least some of the time. On the other hand, since these two participants largely traveled together, and interacted with the same other people, it is likely that the reversed tracks would not greatly hinder behavioral research. Without this single error, identification accuracy on this dataset would surpass 98%.

Most trackers which rely on RF signals for localization report accuracy in terms of the average distance between a target's reported location and its actual location. Taking this metric, Kantor & Singh [2002] achieve an average accuracy of 1.62 feet. Although a different metric, the system described here compares favorably with accuracy of at least 0.5 meters over 90% of the time.

4.6 Discussion and Summary

The tracks created in Chapter 3 represent high quality trajectories of the locations of each target at every moment in time. However, there is no way to differentiate specific targets from one another. Without this functionality, the applications which can use those tracks are limited. Therefore, a method of identifying the targets represented by each track is introduced in this chapter. Using active RFID tags in conjunction with the previously generated ladar-based tracks, allows the targets to be uniquely identified correctly over 90% of the time.

Most work involving localizing from RF signals has focused on localizing based solely on the RF signal, which provides a best case of localization of roughly 1-2 feet – good for some applications, but unsatisfactory for the types of social behavior research which will use these tracks. Alternately, the use of passive RFID tags or other short range beacons are not suited for the environments and targets studied here. Instead, the technique of using RF signals to label pre-generated tracks presented in this work is a novel approach compared to existing research.

CHAPTER 5

BEHAVIOR RECOGNITION WITHIN HONEY BEE COLONIES

The main research question involves an exploration of learning about social behaviors through observations. Once observations have been made, as discussed in Chapters 3 and 4, they can be used to model and recognize behaviors. This chapter describes a system that learns to label behavior automatically on the basis of a human expert's labeling of example data. As discussed previously, this will save the researcher time, which can be better used by the researcher to analyze the automatically labeled data.

The behaviors of interest are sequential activities that consist of several physical motions. For example, bees commonly perform waggle dances. These waggle dances consist of a sequence of motions: arcing to the right, waggling (consisting of walking in a generally straight line while oscillating left and right), arcing to the left, waggling, and so on [v. Frisch 1967]. In this work, the focus is on dancing, following, and active hive work as behavioral roles to be identified.

Specifically, the behaviors are defined as follows. A *follower* is a bee who follows a *dancer*, but does not perform the waggle segments, while a bee accomplishing *active hive work* is neither a dancer nor a follower, yet moves around with apparent purpose. **Behaviors** are distinguished from their constituent **motions**. *Arcing, waggling, moving straight,* and *loitering* are examples of motions, which are sequenced in various ways to produce behaviors. Accordingly, in order for a software system to recognize behaviors, it must also identify the motions that make them up. And conversely, knowing which behavior a bee is executing allows better identification of the constituent motions.

The system described here is designed to label a bee's motions and then identify, from motion sequences, the animal's behavior. There are several steps in the operation of this system. But before it can begin to operate, raw location data of each to be analyzed must be gathered. Therefore, first, marked bees in an observation hive are videotaped and tracking software extracts x- and y-coordinate information for each bee [Bruce et al 2000]. Then, the system begins by computing quantitative features of motion (such as velocity and heading change) from the raw location data. A kernel regression classifier identifies motions from these features (the classifier has been previously trained using data labeled by an expert) [Mitchell 1997]. The labels are:

- **ARCING_LEFT** (AL) – The bee is moving in a counter-clockwise direction
- **ARCING_RIGHT** (AR) – The bee is moving in a clockwise direction
- **STRAIGHT** (S) – The bee is moving steadily in a fairly straight line
- **WAGGLE** (W) – The bee is moving straight while oscillating left and right
- **LOITERING** (L) – The bee is moving very slowly in a non-specific direction
- **DEAD_TRACK** (D) – The bee is not moving at all

Finally, the motion sequences are evaluated using a hidden Markov model which identifies predicted labels of the data set (motions) and inferred behaviors. Hidden Markov models (HMMs), explained in Section 2.6.3, are convenient models of behavior that can also be used for recognition tasks. An HMM describes likely sequences of motion that correspond to specific behaviors. In this application, HMMs are used to increase accuracy by "smoothing" the labels across the data set.

There are a number of algorithms that operate on HMMs that can be leveraged. In this system, the output from the kernel regression classifier is used as input to the Viterbi algorithm over a fully connected HMM [Rabiner 1989]. In this way, incorrect classifications that are statistically unlikely can be discarded or corrected. For example, if there is a series of **ARCING_RIGHT** data points with a single **ARCING_LEFT** in the middle, it is likely that the single **ARCING_LEFT** is an error and should really be an **ARCING_RIGHT**, even though the features quantitatively indicate an **ARCING_LEFT**. The HMM technique will correct mistakes of this nature. HMMs can also be used to identify behavior. By creating an HMM for each of the possible behaviors, the correct behavior can be chosen by determining which HMM most closely fits the data.

The hypothesis is that this system can provide a means of labeling new data with reasonable accuracy. Note that since the overall goal of this recognizer is to identify behaviors automatically, it is not necessary to be able to label every data point precisely. If a majority of individual motions can be labeled properly, then it is possible to infer the correct behavior (dancer, follower, etc). Figure 5.1 shows an overview of the system.

5.1 Tracker

Tracking software is necessary to convert the bee videos into data that can be used by other software [Bruce et al 2000] [Khan et al 2003]. To collect the experimental data

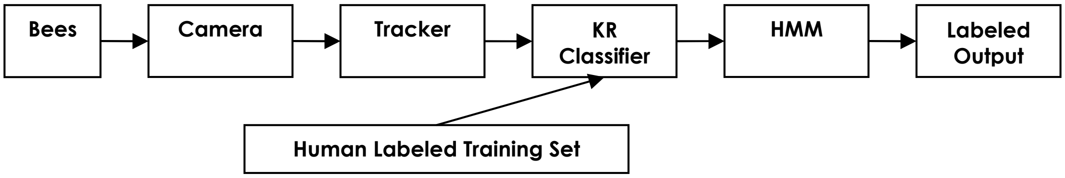
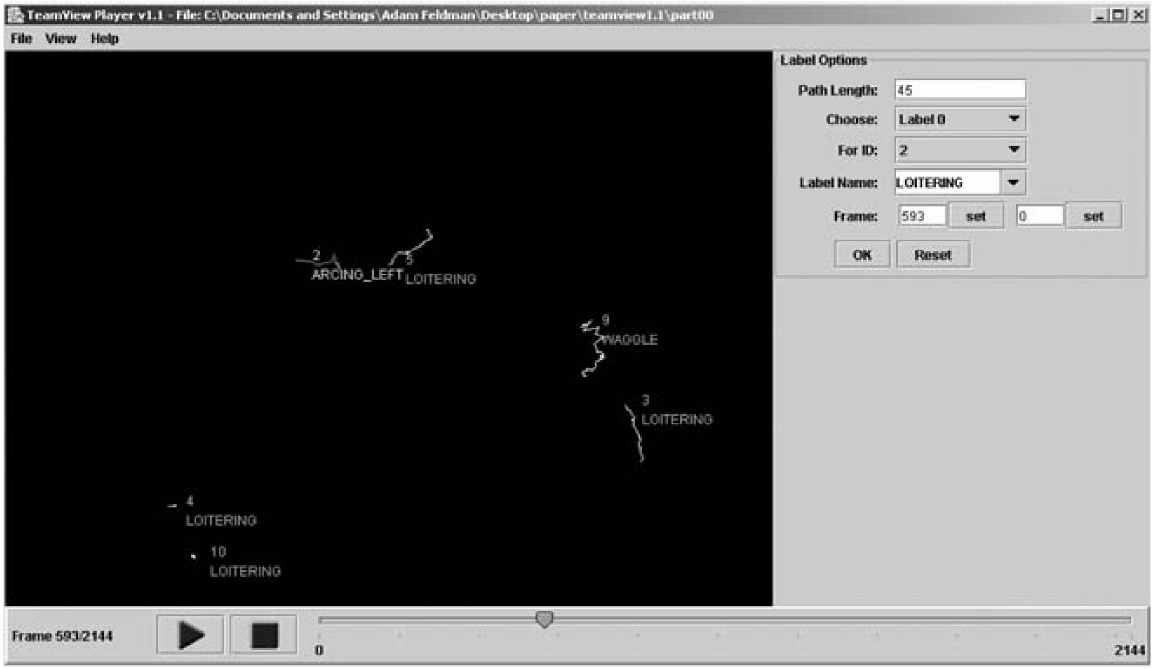


Figure 5.1: An overview of the system. After [Feldman & Balch 2004].

used in this system, some bees were removed from the hive and individually painted, by applying a drop of brightly colored paint (such as red or green) to each bee's back. A video camera was then trained on a section of the hive, and a recording was created. The tracker is then applied to the recording. For each frame of the video, the tracker is able to identify the location of each painted bee that is visible. Since the speed of the video is 30 frames per second, the data now consists of the coordinate information of each (visible) painted bee every 0.033 seconds. This is enough information to get a clear picture of the bee's movements.

5.2 TeamView

The TeamView software (shown in Figure 5.2) is used to visualize and hand label the data sets. The files that contain the x- and y- coordinate information (from the



TeamView Player v1.1 - File: C:\Documents and Settings\Adam Feldman\Desktop\paper\teamview1.1\part00

Label Options

Path Length: 45

Choose: Label 0

For ID: 2

Label Name: LOITERING

Frame: 593 set 0 set

Frame 593/2144

Figure 5.2: TeamView software. Labeling options appear to the right of the main viewing window, while playback controls are at the bottom. The displayed labels were previously created using this software. From [Feldman & Balch 2004b].

tracker) are loaded into TeamView. When the files are played, the main viewing window displays the position of each bee currently in the field. The lines behind each "bee" are a trail, showing where the bee has been over the last x frames (where x is definable by the user). The labeling options allow a user to mark a segment of the video and apply any label to a specific bee. In this way, it is possible to label the motions of each bee across the entire data set. Further, once data is labeled, the labels will be displayed next to the bee they are associated with. The advantage to using this software is the speed with which a human can label the data, as compared to more traditional pen and paper method of using a stopwatch and the original video.

5.3 Data Generation and Feature Extraction

The data used in this system begins as video of bees in the hive, prepared for analysis by the tracker, as discussed above. Once the coordinate information for each tracked bee is obtained from the tracker, numerical features of motion that are used to determine the bee's motion are extracted. All features are calculated for each tracked bee during every frame in which it is visible. Since all values are normalized, the units of measurement can be disregarded. Seven features that were extracted and examined for their usefulness (where t is the current frame in time):

- Instantaneous Speed (v0) – from time t-1 to t
- Speed over a Window (v1) – from t-3 to t+3
- Raw Heading (h0) – from t to t+1
- Heading Change over a Small Window (h1) – from t-1 to t+1
- Heading Change over a Large Window (h2) – from t-20 to t+20
- Speed times Heading (sh0) – multiply h1 and v0

- Average Speed times Heading (sh1) – average of sh0 values from t-5 to t+5

5.4 Kernel Regression Classification

Before kernel regression classification can be used, the appropriate features must be determined. From the information generated by the tracker, seven features are available. It is possible to use all seven of these features, however, it is beneficial to reduce this number if not all features are useful in classification. Reducing the number of features (and therefore the dimensionality of the feature space) will result in simpler and quicker computation, greatly reducing the working time of the system. Also, in some cases, more dimensions can make things worse – they are harmful to classification. This is because two points close to each other in a dimension that does not affect labeling would seem closer together in feature space than if that dimension were not included. For example, bee color has nothing to do with what motion a bee is performing, so it would not be a useful feature. Yet by including it, two bees of similar color which are performing different motions may appear (in feature space) to be more similar than two bees that are performing the same motion (and therefore warrant the same label) but are very different colors. It is obvious that bee color is not relevant, but this example illustrates how additional information, though correct, can be quite detrimental to results.

In order to determine which features are helpful and which are useless (or harmful) in determining the label of a data point, a sensitivity analysis is conducted. Every combination of the seven available features – from each one individually to all seven together – is tested by applying the kernel regression algorithm to a large training set. The combination of features that resulted in the highest accuracy (defined as the percent of the test points labeled correctly) are considered the most useful, and are the only features used in the rest of the experiments.

In the experiments, the training set is made up of 1000 points of each type of labeled motion. This ensures fair representation, despite frequency disparities among the labels (unlike some other methods of selecting the training set). The importance of this can be found in the infrequency of the most useful label – **WAGGLE**. This label is very telling due to its appearance only during a dance. However, **WAGGLE** points make up only 0.1% of the data. Therefore, choosing a random sampling of 6000 data points would result in few, if any, **WAGGLE** points being chosen.

As discussed above, kernel regression classification usually results in a single label being chosen for each point (the label with the highest score for that point). However, in order to provide the HMM with as much useful information as possible, instead of only recording the highest-scored label, this system actually records the (normalized) scores for all the labels. This information represents a sort of "confidence" level in the kernel regression classification. The advantage of this technique over traditional kernel regression methods is that when the classifier is wrong (because the correct answer has the second highest score, for example), the HMM can use the fact that the correct answer has a relatively high score, instead of simply being given the wrong information. This has the effect of helping to account for the large amount of noise in the data.

5.5 Hidden Markov Model

The kernel regression algorithm is very good at classifying data points based on features that are similar in value to those in the training set data. However, there are several reasons why the correct label does not directly reflect the features. For example, often while a bee is arcing right, it will jitter, causing the features to look like there are some frames of loitering or arcing left in the middle. In this case, the classifier will label these frames differently. It is desirable to "smooth" these places where the data isn't representative of what is really going on. Since the kernel regression classifier only considers each point individually, this time series information is lost. Thus, hidden Markov models (HMMs) are examined.

Although many HMMs use a specific topology, this system uses a fully connected HMM, as the system should learn this topology automatically. Instead, the HMM is used to statistically smooth the labels provided by the kernel regression classifier. Therefore, all of the states are connected, and use the training data to determine the probability of each transition (see Figure 5.3). It should be noted that this technique may result in certain transition probabilities dropping to zero, which causes the HMM to no longer be fully connected.

Once the HMM is specified, it will be used by the Viterbi algorithm to determine the most likely state sequence for a given observation sequence. It does this by using time series information to correct "glitches" which are statistically unlikely. For example, if there is a single **ARCING_LEFT** label in the midst of a series of **ARCING_RIGHT** labels, the Viterbi algorithm will decide that the **ARCING_LEFT** is an observation witnessed from the **ARCING_RIGHT** state since the low transition probabilities between **ARCING_LEFT** and **ARCING_RIGHT** make it very unlikely that the state changed twice here.

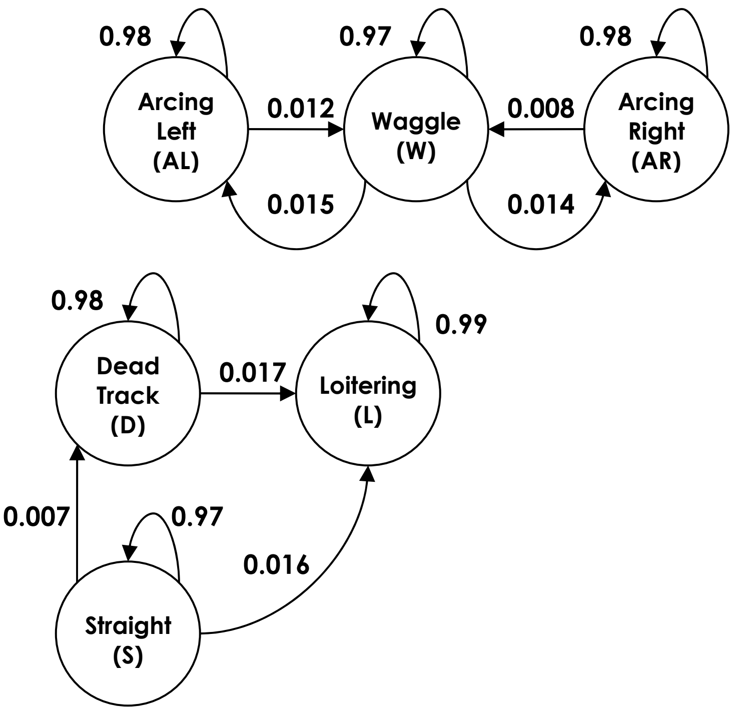


Figure 5.3: Possible HMM, after removing transitions with a probability less than 0.005. After [Feldman & Balch 2003].

The observation sequence given to the algorithm is actually the output from the kernel regression classifier. The form of this sequence is a series of continuous vectors, with one dimension for each possible label. It should be noted that since the observations are continuous (vectors between 0 and 1 in each dimension) instead of discrete, there is no observation table, per se. Instead, there are observation probability *functions*, which represent the probability of seeing a particular observation in a given state. These functions merely equal the value of a Gaussian at the observation. The mean of the Gaussian is dependent upon which state is being examined.

For example, the observations are made up of a 6-dimensional vector, with one dimension corresponding to each of the states (**ARCING_LEFT**, **ARCING_RIGHT**, **STRAIGHT**, **WAGGLE**, **LOITERING**, **DEAD_TRACK**), such as **o** = (u, v, w, x, y, z). u corresponds to the “leftness” of the point, while x represents its “waggle-ness”, etc. The observation function for the waggle state would be a Gaussian centered at (0, 0, 0, 1, 0, 0). Therefore, if observation **o** has a high x value, it will result in a higher probability of being an observation in the waggle state than if it had a low x value. Similarly, a high v value will move it closer to the mean of the **ARCING_RIGHT** state than a low v value, resulting in a higher probability of being an **ARCING_RIGHT** point.

5.5.1 Behavior Recognition

The tasks of motion identification and behavior recognition are usually treated separately, with recognition accuracy being dependent on the accuracy of the motion identifier. This system, however, completes these two tasks in parallel, allowing each to assist the other, by creating an HMM, as above, for each possible behavior. The behaviors considered are:

- **Dancer** – The bee is performing a series of waggle dances
- **Follower** – The bee is following a Dancer
- **Active** – The bee is neither a Dancer or Follower, yet moves around the hive with apparent purpose
- **Inactive** – The bee simply loiters about, not moving in a distinct direction

Each HMM is trained on a data set made up of only the corresponding behavior (as provided by a human expert labeler). Thus, the model for a dancer is different from the model for a follower. These HMMs are then connected via a null, start state, which allows movement to every state in every HMM. However, there is no movement back to the start state, nor between each smaller HMM (Figure 5.4).

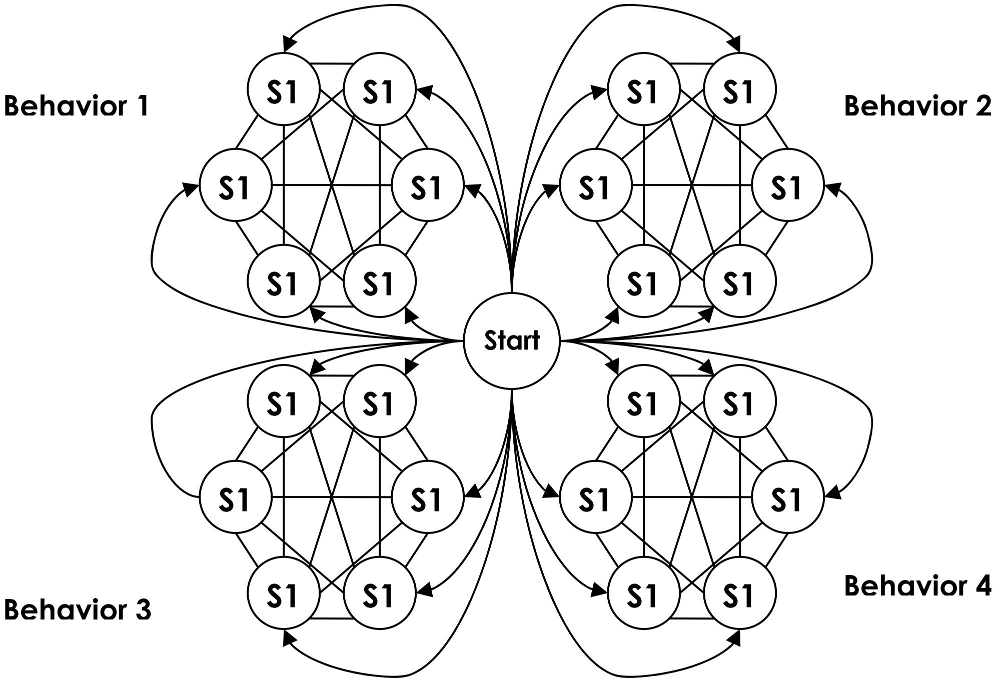


Figure 5.4: Behavioral HMM, which is made up of a start state and the four sub-models, one for each behavior. After [Feldman & Balch 2004a].

This technique allows the Viterbi algorithm to choose the best sequence of motions, by falling into the sub-set of the HMM which best models the data. Simultaneously, the algorithm can best choose the sub-set (and thus the behavior) because it is the one that most closely fits the observations.

5.6 Methods

To assess this classification system, an experimental data set consisting of fifteen minutes of video of honey bee behavior in an observation hive was collected. The tracker was used to extract the features, while TeamView was used for hand labeling. There were three human labelers, each labeling 5 minutes of the data. The data was then broken into a training set, consisting of the last third of the data, and a test set, consisting of the first two thirds. The test set is used only for accuracy validation after training the system.

First, the training set is prepared for use by the kernel regression classifier by having 1000 points of each label randomly extracted and placed in feature space. The remainder of the training set is then labeled, using the technique described above. The data is separated by (human determined) behaviors, and the labels, along with the manually determined "correct" labels, are then examined to find the transition table and the initial state probabilities of each sub-model. These are then combined to form the overall, behavioral HMM.

To establish the accuracy of the system, these 6000 points in feature space and HMM parameters are used to automatically label the test set, labeling both the motion of each data point and the behavior of each entire track (bee). In this phase of the experiment, the correct labels are not known by the system – instead they are only used to evaluate its accuracy.

5.7 Results

5.7.1 Feature Selection

Every combination of the seven available features is tested by applying the kernel regression algorithm to a large training set. This results in 127 possibilities (zero features is not an option). The combination of features that result in the highest accuracy (defined as the percent of the test points labeled correctly) is h2, v1, and sh1. Therefore, only these features are considered in the rest of the experiments.

It is interesting to note that accuracies using these three features plus combinations of other features range from 58.9% to 73.0%, while the accuracy of using only these three features is 73.1%. This demonstrates that having extra features can reduce accuracy.

5.7.2 Classification Results

Table 5.1 shows the fractional accuracy for each label type. The system achieves an overall accuracy of about 93%. Further, the overall accuracy increases by 17.9% by including the use of the HMM to "smooth" the results of the kernel regression classifier. Finally, the accuracy in determining the behavior is 79.8%. That is, roughly 80% of all tracks are automatically labeled with the same behavior as given by the human labeler.

Table 5.2 is a confusion matrix showing how each data point is (mis)labeled. For example, the W column indicates that 75% of the **WAGGLE** points are correctly labeled as **WAGGLE** points, while 9% of them were mislabeled as **ARCING_RIGHT** points.

Table 5.1: Fractional breakdown of accuracy, first with the kernel regression classifier, then with the addition of the HMM. Final column shows number of occurrences of each label in the test set.

Label	Accuracy (without HMM)	Accuracy (with HHM)	Total Occurrences in Test Set
ARCING_LEFT	0.71	0.84	2059
ARCING_RIGHT	0.65	0.83	2407
WAGGLE	0.49	0.75	1550
LOITERING	0.77	0.96	113285
DEAD_TRACK	0.91	0.90	5920
STRAIGHT	0.34	0.39	5343
Total	0.75	0.93	130564

Table 5.2: Fractional breakdown of system labels. Each row shows the percent of that row's label identified as each possible label by the system.

	System Label						
		AL	AR	W	L	D	S
Actual Label	AL	0.84	0.02	0.04	0.08	0.00	0.02
	AR	0.02	0.83	0.03	0.09	0.00	0.03
	W	0.10	0.09	0.75	0.01	0.00	0.05
	L	0.01	0.02	0.00	0.96	0.01	0.01
	D	0.00	0.00	0.00	0.09	0.90	0.00
	S	0.06	0.09	0.00	0.45	0.00	0.39

5.7.3 Discussion of Results

As hypothesized, the use of an HMM in conjunction with a kernel regression classifier provides higher accuracy than a kernel regression classifier alone. The HMM improves overall accuracy by almost 18% above the 75.1% accuracy of kernel regression alone. The two labels that correspond to the vast majority of the data (**LOITERING** and **DEAD_TRACK**) are very similar to one another, both in features and in appearance. Due to this fact, and some ambiguity among the human labelers, misclassifications between them are less important than other misclassifications. If these two labels were combined into one, the accuracy of the system would be approximately 94.1%.

Another label that caused many problems for the system was **STRAIGHT**. This label was included to make the system as general as possible. However, none of the common bee behaviors (dancing, following, active hive work) seem to rely on this label. Therefore, it would be possible to eliminate this label. Removing all points labeled **STRAIGHT** from consideration would increase the accuracy by about 2.5%, to 95.5% (or about 96.6% after combining **LOITERING** and **DEAD_TRACK**).

It should be noted that if the system merely labeled each point **LOITERING**, an accuracy of 86.8% would have been achieved. Although not much lower than the 93% result, this is accuracy based on a frame by frame comparison. However, since the ultimate goal is identifying the bee's behavior, it is not important that every frame be correctly identified, as long as each segment of like frames is recognized. For example, if the system says that a series of **WAGGLE** points starts and ends several frames before or after the "correct" labels indicate, it is of little importance, as the behavioral recognizer is still given a **WAGGLE** sequence of approximately the correct length.

The system achieves an accuracy of 79.8% in identifying the behaviors. It is possible that this is not a higher value because the four behaviors are so similar. This means that the transition probability table for each behavior is very similar to the transition probability tables of the other behaviors.

An even bigger factor which reduces the system's behavior recognition accuracy is the assumption that behaviors persist for the entire duration of a bee's presence. However, in reality, a bee will switch behaviors. For example, it will enter the hive and find a suitable place to begin dancing (Active Hive Bee), then it will dance for a time (Dancer), then it will move to a new location (Active) and begin dancing again (Dancer). By not letting a bee change behaviors, the models become diluted, and the all-important distinctiveness is lost.

5.8 Discussion and Summary

The system of modeling behaviors examined in this chapter achieves an accuracy at the motion level of approximately 93%. Further, a behavior accuracy of almost 80% has been realized, despite the inaccuracies introduced by the method of labeling behaviors. Thus, the system proves that its techniques are sound, and provides reasonable accuracies, with room for improvement by structuring the data slightly differently, as discussed above. By successfully mimicking the labels generated by a human labeler, this system is a step towards the ultimate goal of performing automatic behavior recognition, in the biological systems domain, without the need for a human labeler.

CHAPTER 6

INTERACTION DETECTION BETWEEN ANTS

Chapter 5 presented one approach to learning behaviors in a social biological system. This chapter introduces a method of detecting various interaction events between multiple nearby individuals. The technique was developed for myrmecologist Stephen Pratt to help him automatically detect interactions between ants in his research, as discussed in Chapter 2. Recall that the interactions to be detected are head to head, head to body, and body to head (from the point of view of the other ant in a head to body interaction). Instead of using the sensory perception method previously discussed, this system uses expertly labeled data to train a simple model. One of the goals of this research requires designing the system in such a way that a person not trained in computer science could understand how the results are derived.

The focus here is on using timestamped trajectories of the pose of each ant in order to detect and classify the various interactions. Therefore, any tracking technique that can handle multiple, interacting agents can be used to generate the trajectories. The experiments presented here use the tracker described by [Khan 2005].

6.1 Approach

Once trajectory data has been gathered, features must be extracted from which the model is built. These are the observable attributes of the trajectories which are used to model the interactions. Then, a portion of the data (the training set) is hand labeled by an expert, while the rest is put aside for validation (the test set). This training set is used to determine the thresholds of each feature for each type of interaction. The thresholds can be used to label new data (such as the test set). Once the labels are generated, they are updated in two post processing steps, to take advantage of the symmetry of interactions and to smooth over time.

Three features of a potential encounter are selected; all are easily determined from the data. The features are illustrated and described in Figure 6.1. The three features are calculated from the point of view of each ant (called the focal ant). As with the interactions, the features are also symmetrical, with one focal ant's theta being another ant's phi, and vice versa. Although the figure only shows two ants, it is not uncommon for three or more ants to be in close proximity to each other. To account for this, the feature data for each ant includes distance, theta, and phi for up to the three closest ants. Then, interaction detection is attempted separately on each of these pairings. The highest priority interaction, if any, is chosen as the label. The priority of interaction is based on the expert-perceived value of the interaction type, with the following order: head to head (HH), head to body (HB), body to head (BH), and no interaction (X). Therefore, if three ants are in close proximity and the focal ant is labeled as having an HH interaction with one ant and a BH interaction with another ant, it will be given the HH label even if the BH ant is closer.

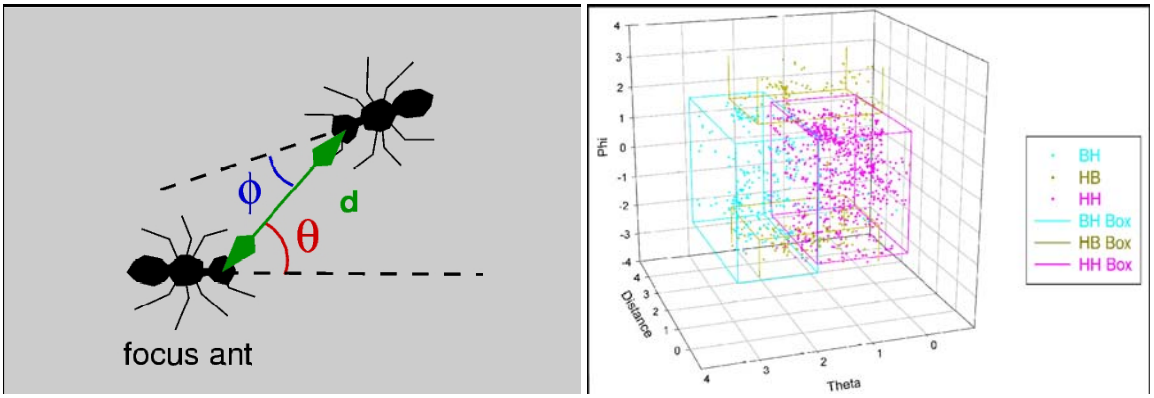


Figure 6.1: Left: Three features used to classify interactions between ants. Right: Training data is plotted in feature space. The boxes illustrate the thresholds used to identify each type of interaction. From [Balch et al 2005].

Once the features have been calculated, the thresholds can be determined. This is accomplished through a hill-climbing optimization. For each type of interaction (set of thresholds), the goal is to maximize the fraction of data points with that label inside the bounding box while minimizing the fraction of data points with different labels inside the box. Only data points which are labeled as an interaction are included in evaluation, causing the many data points representing a lack of interaction (X) to not count. Each interaction type is processed separately, generating a set of six thresholds (minimum and maximum values of each feature to be considered that interaction type). It is possible for overlap between bounding boxes; in this case, the interaction type with the highest priority will take precedence in the overlap area. Likewise, areas which are outside all three bounding boxes correspond to no interaction (X) taking place.

To find the optimal bounding box (set of six thresholds) for each interaction type, it is first initialized to be the smallest box which encloses all data points with that interaction type. The algorithm then adjusts the bounds of the box incrementally - in each iteration, the threshold change that results in the minimum error is accepted. The error function to be minimized is:

Error = # data points of current type outside the box * total # data points of other classes
+ # data points of other types inside the box * total # data points of current class
(the error is then normalized by the total number of data points)

To help reduce the effects of local minima, the entire process is repeated with three subsequently smaller steps. The resulting thresholds determined from a 5 minute labeled sequence are illustrated in Figure 6.1. These thresholds are used to label new data points, before two post processing steps are performed.

First, labels are updated to take advantage of the symmetry of interactions. Because each interaction involves two ants, any interaction should be specified in the labels of both ants. If the labels do not agree, then the possibility with the highest priority is used. For example, if ant 1 is labeled as having an HH interaction with ant 2, but ant 2 has anything other than an HH interaction with ant 1, its label is updated to be HH. The second post processing step is to temporally smooth the data. This is done by changing the label of frames that disagree with the previous and following frames.

6.2 Methods

Learning from the mistakes in evaluating the bee behavior identification system, this system is evaluated by checking for the detection of interaction events. Instead of comparing the system's labels to the ground truth on a frame by frame basis, which is not necessarily a useful measure of performance, events consisting of identical labels will be examined. In this way, every continuous block of a single label will be considered one event. Therefore, a single event consists entirely of frames with the same label, whether an interaction (such as HH) or not (X). An event is considered as having been detected if the correct label is given in at least one frame of the event.

The system is evaluated 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, and then depart. There are hundreds of interaction events in each video. The labeling of one video is the training data for the system, which then labeled the other video. The automatic labeling of the test data is compared to the human labeling of the same data in two ways; the percent of interaction events detected and the number of extraneous events detected.

6.3 Results

The automatic method correctly identifies 94% of the interactions. The order of training is then reversed: the system is trained on the second video, and then tested on the first. In that case the system correctly identified 87% of the interactions. Figure 6.2 shows an example frame indicating labels provided by the system.

This performance is a good start, having detected almost all of the interactions. However, the system reports as many as 43% too many events, many of which are false positives (detecting an interaction when there was not one).

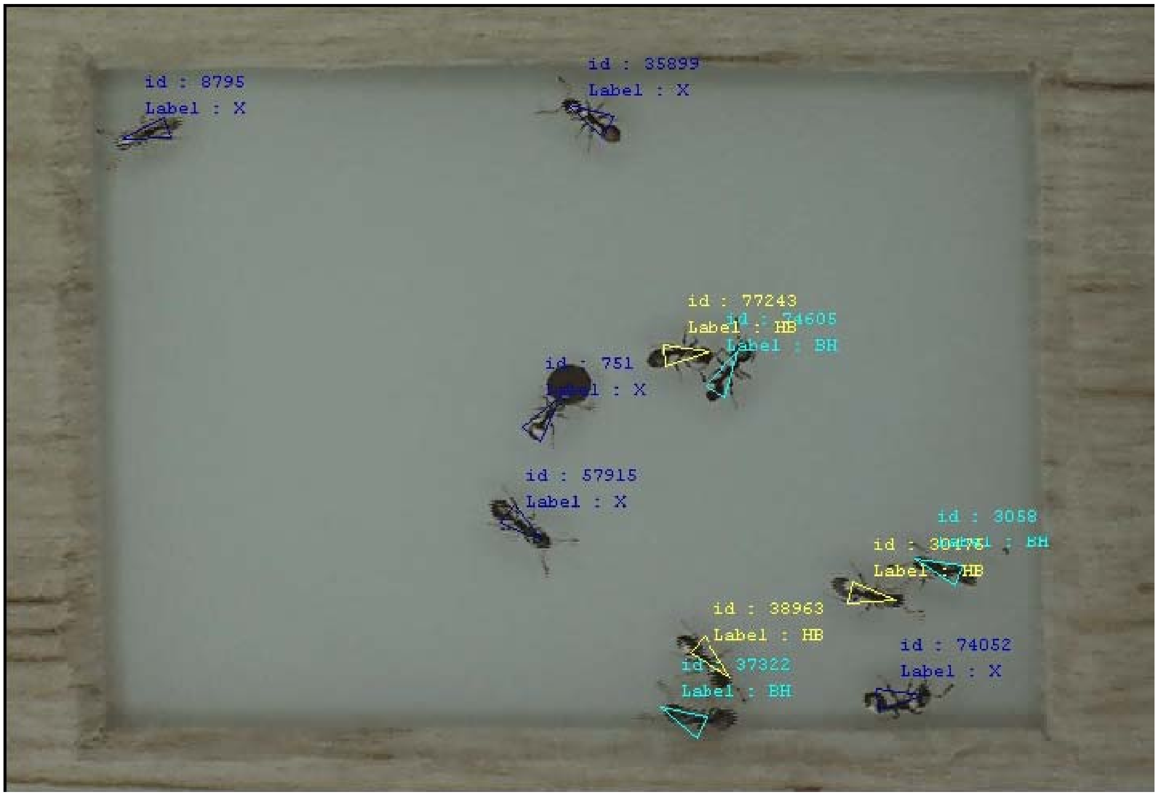


Figure 6.2: This image shows an example frame of video of Leptothorax albipennis labeled automatically by our human-trainable system. The colored triangles over the animals are coded for the different types of interaction that animal is experiencing (cyan: BH, yellow: HB, magenta: HH, blue: X). From [Balch et al 2005].

6.4 HMM Comparison and Integration

The system created to identify bee behaviors is tested with the ant data sets. First, the system is used as described in Section 6.1 through the extraction of the three features, which are then given to the kernel regression classifier, etc. (as described in Chapter 5). In this case, there are many fewer extraneous events detected, due to the sophisticated smoothing influence of HMMs that tends to suppress brief "noisy" detections. However, there is also a decrease in event detection accuracy, with only 71% and 77% of the interactions detected.

Finally, a hybrid system is attempted. For this attempt, the threshold technique is first used to generate labels, then the labels are used with an HMM to improve accuracy, as described above. This hybrid system achieves a better 84% interaction detection accuracy (on both data sets), while maintaining a relatively low number of extraneous interactions. The results of all three systems are summed up in Table 6.1.

Table 6.1: Interaction detection accuracy for each of the techniques attempted. Threshold is the initial system, KR/HMM is the system described in Chapter 5, and Hybrid is the Threshold system combined with an HMM. Dataset 1 has 191 events (98 interactions) and dataset 2 has 136 events (70 interactions). Total Extras refers to the number of events in the automatically labeled data which are completely wrong.

Method	Train Set	Test Set	Ints Detected	X's Detected	Total Events Detected	Total Events Labeled	Total Extras
Threshold	1	1	94.9%	69.9%	82.7%	408	164
Threshold	1	2	87.1%	68.2%	77.9%	294	129
Threshold	2	1	93.9%	70.1%	82.7%	388	165
Threshold	2	2	90.0%	65.2%	77.9%	244	114
KR/HMM	1	1	89.8%	57.0%	73.8%	185	43
KR/HMM	1	2	77.1%	57.6%	67.6%	129	33
KR/HMM	2	1	71.4%	72.0%	71.7%	184	50
KR/HMM	2	2	90.0%	68.2%	79.4%	129	20
Hybrid	1	1	84.7%	59.1%	72.3%	194	53
Hybrid	2	1	84.7%	62.4%	73.3%	176	40

6.5 Discussion and Summary

As with the other behavior recognition techniques presented herein (in contrast with most existing work), the main concern is with behaviors between multiple, socially interacting agents in biological domains. Biological researchers spend an inordinate amount of time gathering behavioral data from the systems they study. This task is especially arduous in social systems, in which the behaviors of many individuals must be simultaneously observed. These researchers would benefit from the introduction of tools automating any parts of this data collection process. For instance, the techniques presented in this chapter build upon earlier work to provide the frequency and types of interactions between Leptothorax albipennis ants searching for a new habitat to a myrmecologist. This way, he can gather much more data than would be possible through manual observation, allowing the testing of many more hypotheses about *why* these ants interact. Because this information is used in aggregate, 100% accuracy is not required to be useful.

CHAPTER 7

LEARNING PRIMATE SOCIAL FAMILY HIERARCHY

The previous two chapters describe algorithms which model, recognize, and detect behaviors and interactions within social insect colonies. The research in this chapter expands interaction detection to non-human primates (as simulated by human primates), and then seeks to use this information to learn information about the group as a whole. Among many biologists, one area of study involves determining the social structure within a colony of animals. This social structure, or the definite relationship between the individuals in the colony, must be established before many subsequent behavioral studies can be carried out. For instance, Drea & Wallen [1999] demonstrated that performance on learning tasks in subordinate-ranked rhesus monkeys varied dramatically based on the presence of hierarchical superiors. This "playing dumb" effect, also encountered in human societies, can skew the results of behavioral studies if not known and controlled for. It has also been shown that certain behaviors, such as grooming, are used to express kinship, but also occur as tools by subordinates to achieve the agonistic support of higher ranked individuals [Schino & Aureli 2008]. In order to ascertain the motivation of such behavior, the researchers need to know the hierarchical relationship of the individuals.

The social hierarchy dictates the manner in which individuals interact with one another. For instance, monkeys may only be aggressive to those of a lower ranking, who behave submissively in such situations. The social hierarchy among these animals, which are matrilineal, is based upon a monkey's lineage to the lead female. Therefore, learning the hierarchy of the individuals amounts to learning the hierarchy of the families, which can be grouped by ranking: the alpha family, the beta family, etc. Each individual of the beta family, for example, is considered to be more highly ranked than any member of the gamma or delta families.

Often, the family relationships are inherently known by the researchers. In such cases, the hierarchy can be learned by observing and recording a number of interactions between members of different groups. By examining trends among these interactions, relative rank between specific families can be determined. With enough observations, the definite rank of all families (and thus all individuals) can be found. The process is made more complicated by an incomplete adherence to the general ranking rules by certain individuals (e.g. a monkey may occasionally be aggressive to a superior).

In a colony with only 100 individuals (divided into a number of families) to be classified, hundreds or thousands of interactions must be observed to fully proscribe the family relationships and rankings of all individuals. As previously stated, this task is very time-consuming, greatly slowing the pace of behavioral research which relies on this information. Fortunately, once uniquely identified tracks of each animal's trajectory can be created, a great deal of interactions can be automatically found and used to learn the social structure.

The first step in automatically ranking the individuals of a colony is to detect potential interactions from the tracks. These interactions consist of two individuals behaving in a way which is generally known to occur only between members with a certain hierarchical relationship, such as superior/inferior, similar rank, or only members of the same family. Likely interactions can be found based on proximity over time. Once found, each interaction is classified as one type or another (e.g. grooming, aggressive, etc.). Individual monkeys' tendencies can then be used to assess relative hierarchy. Finally, the families are ranked according to overall interactions of all family members.

7.1 Detecting Interactions

Instead of attempting to create a label for each pair of agents consisting of the type of interaction occurring between them in every frame (including "none" when there is no interaction taking place), periods of likely interaction are detected based on track proximity. Then, features representing the potential interaction as a whole are generated. These features are then used to classify it (based on known examples) as a specific type of interaction or as not an interaction at all.

Some interactions can only occur between individuals within close spatial proximity to one another. For example, one monkey cannot groom another which is several feet away. However, one important type of interaction is aggressive, in which one agent chases another. During these interactions, the individuals are often never within one or more meters of each other. This is because it is rare for the aggressor to actually catch the agent being chased. To detect occurrences of such interactions by looking at the relative locations of each agent in each frame, the permissible distance between interacting agents would necessarily be several meters. This would result in finding a large number of interactions between individuals merely sitting or walking several meters apart.

Even if two individuals are never very close to each other in any point in time, such as during a chase, their trajectories are near one another (possibly even overlapping) within a short period of time. Therefore, by considering trajectory proximity – instead of physical proximity at an instant in time – the spatial threshold can be greatly decreased. This will allow so-called chase interactions to be detected, without resulting in detections of moderately distanced individuals without (nearly) crossing trajectories. Instead, only pairs of agents which are very close to one another within a short temporal offset are found to be interacting.

The detection process works as follows. For each frame of data, each pair of agents is examined. A line is drawn between the current location of each agent and its location one second in the past. If the minimum distance between these two lines is "small enough," then the two agents are said to be interacting in this frame. The minimum distance required to indicate an interaction is roughly the distance between two individuals engaging in one of the stationary interactions of interest. Also, the minimum distance required to continue an interaction is slightly higher than that required to start an interaction to prevent frequent toggles between interacting and non-interacting states between individuals near the threshold. Figure 7.1 includes examples when interactions would and would not be detected.

As each interaction is detected, important features which will be used to classify its type are recorded. The features will be used to classify the interaction based on hand-labeled training data using the kernel regression technique described above. The six features are:

- Interaction length – the number of frames from the start of the interaction until its end,
- Minimum total distance traveled – the total distance (in meters) traveled by the participant which moved less,

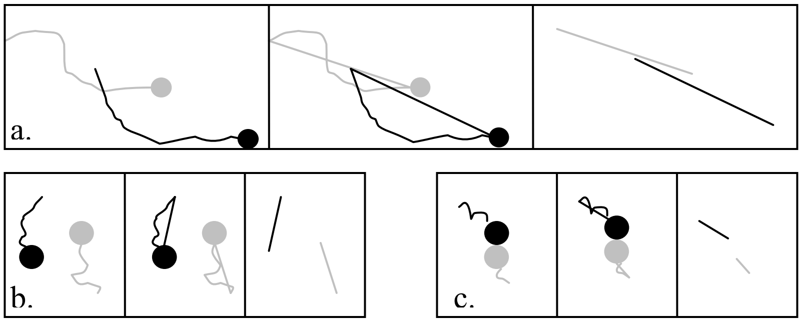


Figure 7.1: Three potentially detected interactions. In each, the circles represent the current track locations while the freeform lines are the trajectory over the last second. The second image shows a line drawn from the current location to the past location. In the third image, the minimum distance between the lines is used to determine if an interaction is taking place. (a) Even though the targets are always far apart during this chase, their lines are close. (b) These targets are closer than those in (a), but no interaction is detected because their lines remain far apart. (c) These interacting targets are detected because they (and their lines) are very close together.

- Maximum total distance traveled – the total distance (in meters) traveled by the participant which moved farther,
- Average distance – the distance (in meters) between the two participants, averaged across every frame of the interaction,
- Minimum relative heading – the minimum of theta and phi (after Figure 6.1) after each is averaged across every frame, and
- Maximum relative heading – the maximum of theta and phi.

The total distance traveled and relative heading features are categorized by maximum and minimum values because it is unknown at this stage which participant is fulfilling which role in each interaction.

To generate the training examples which populate the feature space used in classification, the interaction finding algorithm is used to detect interactions in the training dataset. These interactions are then given a ground truth label by looking for a hand-labeled interaction between the same two agents which at least partially overlaps in time. If such an interaction is found, the detected interaction is assigned the same label (aggressive, grooming, etc.); otherwise, the detected interaction is labeled as "non-interaction." In this way, the feature space can be populated by examples of each type of interaction, as well as examples of non-interactions which might be detected but should be differentiated from actual interactions. Each detected interaction can then be assigned a type based on its distance, in feature space, from each of these training examples.

7.2 Determining Family Hierarchy

Once the list of detected interactions has been labeled according to type, the relative social ranking can be decided. Only interactions which indicate the relative rank of participants (outside of a single family) are considered. For example, an aggressive interaction may indicate that one participant is of an arbitrarily higher ranking than the other while a grooming interaction only occurs between two individuals with a similar ranking (i.e. a member of the highest ranked family will not often groom a member of the lowest ranked family).

A "dominance factor" is calculated for each family. This factor represents the fraction of interactions involving this family's members in which it is the dominant party (e.g. the aggressor in a chase interaction). For each interaction occurring between members of two different families, the dominance factor of each is adjusted accordingly. Interactions which indicate a disparity between the participants' ranking will result in an increase of the dominant party's family and similar decrease in the submissive party's family. Likewise, interactions which indicate a similarity between participants (such as grooming interactions) will result in a partial equalizing of the involved families' dominance factors, by increasing the lower family and decreasing the higher. The amounts of these adjustments for each type of interaction can be learned from a body of training data or provided by an expert.

By adjusting the dominance factor of each family whose members are involved in every interaction, families which are involved in more "lower rank" behaviors (e.g. being submissive, grooming other lower ranked families, etc.) will have lower dominance factors than those families which engage primarily in "higher rank" behaviors, such as frequently being aggressive. In this way, the families become ordered relative to each other, even if every individual does not interact with every other individual. Also, by looking at the frequency of interaction types and participants across a large body of interactions, the impact of an occasional incorrectly identified interaction (whether due to an erroneous classification or a monkey behaving abnormally) is minimized.

7.3 Methods

There are many aspects that complicate attempts at tracking monkeys, including their small size, 3-d movements, and the many occlusions in their environment. Therefore, to test this method of determining family social hierarchy, the social behavior experiment datasets introduced in Chapter 3 are used to simulate actual animal behaviors. In these datasets, 25 individuals were given note cards detailing several behaviors in which to engage. Additionally, 20 people were assigned to one of four families (alpha, beta, gamma, delta), with 5 people per family. These 20 participants play the part of female monkeys, while the remaining 5 participants play the part of the male monkeys, which are outside the family groups of the females (but have their own hierarchy). All participants were instructed regarding which other families/individuals could be interacted with in each of the following ways:

- Aggressive: Individuals could chase others of belonging to lower ranked families. Males could aggress any female.
- Submissive: If aggressed by a member of a higher ranked family or any male (if a female), individuals must retreat.
- Proximity: Includes two individuals standing next to one another, within an arm's length. Only appropriate if participants are within one hierarchical rank of each other if female or of opposite sex.
- Grooming: One individual solicits grooming by bending at the waist. Groomer must be within one hierarchical rank if female or of opposite sex, and should stand close and scratch the other's back for several seconds.
- Mating: Any female follows any male closely for several seconds, and then bobs head. Male stands directly behind female.

These types of interactions were developed with a domain expert and designed to mimic the sorts of interactions which would be common among an actual monkey colony. Instead of attempting to detect and differentiate between all of these interaction types, some adjustments are made. For example, there are few instances of mating behaviors, and so this interaction type (which does not help differentiate families or rankings in this experiment) is ignored. Additionally, because the sensors used to perform tracking do not detect body part locations, grooming and proximity interactions cannot be distinguished and are combined as "affiliative" interactions; fortunately, both interaction types have the same participatory rules. Finally, because the small number of participants playing the part of male monkeys maintains their own social structure independent of the females, and since the males do not often interact with one another, only the social structure of the females is examined herein.

Aggressive interactions contain much richer information pertaining to determining social hierarchy than do affiliative interactions. This is because aggressive interactions occur only between two individuals with a specific relative hierarchy (one is higher than the other). On the other hand, the affiliative interactions could occur between individuals of the same family or individuals with slightly different ranking. Therefore, only aggressive interactions are considered in determining rank for this experiment, although the affiliative interactions can be useful in actually determining family membership in future work.

Instances of aggressive interactions were hand-labeled for the training data. All other detected interactions are considered to be examples of the affiliative interaction group (proximity or grooming). Some of these training examples are then used to classify interactions among the other datasets. There are a total of three 9 minute runs used in this assessment. Additionally, there are two versions of each dataset – one consisting of the actual ground truth tracks and identifications (run1g, run2g, run3g), while the second is made up of the tracks and identifications automatically generated (run1t, run2t, run3t), as detailed in Chapters 3 and 4. Two hand-labeled datasets are used for training and validation data – the ground truth tracks of the second dataset (run2g) and the automatically generated tracks of the third dataset (run3t). The detection and classification of interactions are assessed by using one hand-labeled dataset (run2g) to find and label the interactions in the other (run3t), then comparing to its hand-labels.

The hierarchical rankings are learned for each of these datasets. Performance is assessed as the number and magnitude of families placed at the wrong hierarchical ranking. For instance, if the alpha family is ranked lowest, the error for that family would be 3. Likewise, if the beta and gamma families are reversed, the pair would have an error of 2. The maximum error is 8 and a random assignment would produce an average error of 5. Figure 7.2 shows a graphical representation of perfect assignment – each family is assigned a shade, with the darker shades representing higher ranking.

Figure 7.2: Graphical depiction of correct social structure assignments.

7.4 Results

The run3t dataset is used to find and classify interactions in the run2g dataset. Of the 76 aggressive interactions manually identified, 53 are found and 40 correctly classified as aggressive. Table 7.1 shows the complete confusion matrix.

Table 7.1: Breakdown of system labels. Each row shows the number of that row's label identified as each possible label by the system.

	System Label		
Actual Label		Aggressive	Affiliative
	Aggressive	40	13
	Affiliative	98	316

There are a large number of affiliative interactions misclassified as aggressive. Recall that affiliative is a catch-all that includes any interaction which is not found to be aggressive. Many of the erroneous aggressive detections are due to one individual aggressing a whole group. The human labeler only indicates an aggressive detection involving the one individual who appears to be the target. Therefore, the interactions between the aggressor and the other members of the group are technically considered affiliative (since they are not specifically labeled as aggressive by the human labeler). However, the system found there to be aggressive interactions between the aggressor and several members of the group. While not strictly correct, this is not entirely wrong, as the other members of the group often withdraw, as does the specific target, resulting in the data visually appearing to contain multiple aggressions. Additionally, since participants often cluster along family lines, these erroneous detections actually provided information useful to learning the social structure. This, then, is a failing of the human labeler, not the automatic identifier.

Despite these inconsistencies involving interaction detection, determining hierarchy is successful. Figure 7.4 shows the results of automatically learning the family rankings in each of the test datasets made up of the ground truth tracks. That is, the hierarchies are automatically learned from tracks which were previously manually corrected and labeled. Of the three datasets, the correct hierarchy was learned in two; the third only reversed the two middle families. The errors, therefore are 2, 0, 0 – for an average of 0.67.

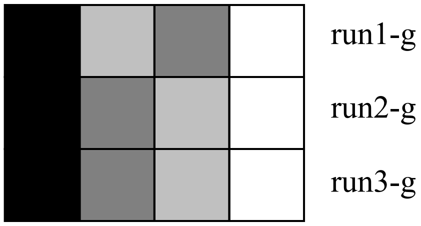


Figure 7.4: Graphical representation of the automatically learned rankings for the hand generated tracks in datasets run1, run2, and run3.

Figure 7.3 illustrates the results on the three datasets created from the automatically generated tracks and identifications. Although these datasets averaged an approximate 10% identification error (as detailed in Section 4.5), the hierarchies that are learned only contained one more error than in the datasets from the hand-labeled tracks (namely, the reversal of the alpha and beta families in run2). These errors are 2, 2, and 0, or an average of 1.33. This supports the hypothesis that occasional tracking errors have minimal impact in the higher level applications.

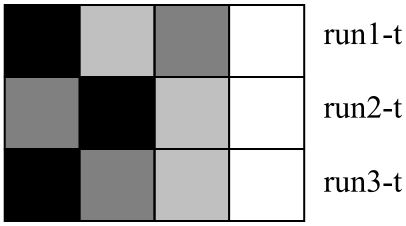


Figure 7.3: Graphical representation of the automatically learned rankings for the automatically generated tracks in datasets run1, run2, and run3.

Currently, biologists must study many hours of data in order to detect sufficient interactions to generate the social structure of a colony. However, each of the test datasets are only 9 minutes long. Because aggressive interactions are fairly scarce within these short datasets (as often in real life), the algorithm is tested against a combination of run1-t and run3-t (run2-t is left out because of its similarity to run2-g, the training dataset). Figure 7.5 shows that in this case, the perfect hierarchy (error of 0) is discovered. By demonstrating that more data helps overcome the noise of intermittent incorrect track labeling or false interaction detections, this result demonstrates that the algorithm has promise, even in the more complex real-life primate colony domain.

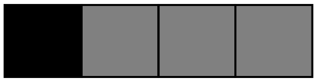


Figure 7.5: Graphical depiction of learned hierarchy for combined run1-t and run3-t.

7.5 Discussion and Summary

Some researchers have drawn inspiration from biological systems for assisting robots in learning to behave according to social relations [Matarić 1997]. But applying machine learning techniques towards automatically learning the social hierarchy of an actual colony of primates is original. Although the work outlined in this chapter does not reach that eventuality, the groundwork is laid through an experiment involving humans simulating the actual hierarchical behaviors of rhesus monkeys. This experiment demonstrates that the approach has the potential to learn actual social relationships in these animal colonies, as discussed in Chapter 8.

CHAPTER 8

SUMMARY AND CONTRIBUTIONS

There are many domains which currently or theoretically have a need to model and recognize behaviors from various types of observations. In some of these cases, such as in biological research, these models and detections are currently carried out by hand, consuming vast quantities of time. Others, such as security and sports applications, are not fully realized, as completing this process manually is not always feasible. This dissertation presents techniques designed to help automate the process in several fields, namely team sports (basketball) and ant, honey bee, and primate colonies.

Observations into these environments are made with a variety of sensors and logged to disk or processed in real-time. The trajectory of each individual is then automatically extracted, after the uninteresting data (i.e. the background) is removed. The target (or agent) which each trajectory represents is assigned a label to uniquely identify it, allowing the behaviors and interactions of specific individuals to be examined. From this trajectory information, models are created which represent the actions and behaviors of the individuals in each context. Finally, these models are used to recognize future instances. In the case of the human primate experiment, the detected interactions are used to reason about the social structure of the entire group.

The specific contributions of this work are:

- An algorithm for **tracking an unknown** and **changing number** of targets as they enter, move through and/or exit the observed planar arena using data from multiple sensors **in real-time** (Chapter 3).

- A method of integrating data from active RFID tags to produce labels which **uniquely identify each track**, without the loss in track precision endemic of noisy RF sensors (Chapter 4).
- Application of machine learning techniques to **model motions and behaviors** based on the trajectories of honey bees in a hive (Chapter 5).
- Methodology for **detecting frequency and types of social interactions** between pairs of Leptothorax albipennis ants exploring a potential nest site (Chapter 6).
- Experimental results of **learning social structure** automatically from raw sensor observations (Chapters 3, 4 and 7).

This chapter reviews the main contributions of the research and summarizes the results presented herein, before concluding with an examination of directions for future work that show high potential.

8.1 Detection-Based Tracking

Before any behavioral models can be learned, trajectories of each target must be generated as they move through the environment. It is not feasible for humans to manually create these tracks with accuracy at the scale necessary to be useful in real applications; even if possible, this would defeat the goal of saving humans time. Instead, the algorithms presented provide a robust mechanism for the automatic creation of trajectories.

The process consists of gathering data from a series of laser range finders. These datasets are registered to one another in time and space before the background is subtracted. The algorithm that uses this detection-based data relies on iterative closest point (ICP) to simultaneously locate the targets and perform data association in each frame. The targets cannot be identified specifically, but multiple models can be used to differentiate between targets of different types or in different states. Whenever two or more tracks become so close together that they cannot be clearly differentiated in the data, they are split into new tracks, preventing a single track from inadvertently representing more than one target.

The tracker is tested in experimentation with 8 laser range finders observing a basketball court. One experiment involves 10 people playing basketball and the other consists of 25 people walking and running around according to a script of common monkey behaviors. In both cases, over 98% of the track-frames are detected and the tracks averaged approximately 40 and 340 seconds, respectively. While there are a few track jumps, these only occurred every several minutes at most. Further, the tracker is tested in a real-time environment, and shown able to track 10 targets at over 37.5Hz and 25 targets at about 28Hz, a frame rate high enough to have minimal impact on the tracking results.

Compared to existing trackers, this work is differentiated in a number of ways. First, it is designed to track an unknown and potentially changing number of targets. This does not add complexity or slow the algorithm down. Therefore, unlike many other trackers, it will work in real-time, including the data association step. Further, most existing laser-based applications only make use of the data from one ladar, or consider each ladar independently. On the other hand, this work combines readings from multiple ladars in order to expand the field of view of the system and reduce the effects of occlusions. Finally, this tracker operates on detection-based data, regardless of what type of sensor the data initially came from.

8.2 Track/Target Association

Chapter 4 outlines a novel technique for labeling each track with the target it represents by employing active RFID tags. Adding id-sensors in addition to the ladars used for tracking allows the creation of uniquely identified tracks. Without these identifications, some basic group-wide aggregate behaviors can be studied. However, these identifications are necessary for the majority of behavior modeling and recognition tasks.

The environment is discretized and a series of lookup tables are generated from training data to build histograms of signal strength/location occurrences for each antenna. After normalizing, each table represents the probability that a tag is at each location when a given signal strength reading is received by a given reader. All track/tag pairings are scored based on the known location of the tracks at the time of each tag reading. Since two tracks which overlap in time cannot represent the same tag, the order of the final labeling is important; once a label is used, it is made unavailable to temporally overlapping tracks. The crucial labeling order is decided based on calculating confidences for each track. The most confident track is assigned to its highest scored tag, and then the scores and confidences of the remaining tracks are recalculated, repeating until all tracks are labeled.

The RFID-based identification algorithms are tested with the tracks created from the two experiments previously described – a 16 minute basketball game and 9 minutes of a social behavior experiment involving 25 people. Almost 96% of the ground truth track-frames are accurately labeled in the basketball game experiment. The social behavior experiment also achieves positive results, with over 90% of the track-frames correctly identified. Further, approximately 80% of the incorrect track-frames are due to the reversal of two individuals who spent the entire experiment in close proximity to one another.

Localizing based on RF signal strength is a well studied problem. However, this technique differs significantly from existing approaches. Most researchers attempt to localize directly from the RF data, whether alone or combined with data from another sensor (such as with a Kalman filter). On the other hand, this work capitalizes on the precise nature of the laser range finders to determine the exact track locations; the RFID tags are merely used to determine the identities of the already detected tracks. Alternately, short range sensors (included passive RFID and IR badges) are used as detectors to determine proximity to known locations. This requires instrumenting the full interior of the environment being observed, a constraint which is not always feasible in sporting venues or animal habitats. By placing long range active RFID antennas around the perimeter, this research has no such requirement.

8.3 Social Insect Behavior Modeling and Recognition

Machine learning approaches are applied to two very different real-world social insect behavior modeling problems in Chapters 5 and 6. In the first, the motional structures of honey bee behaviors in the hive (such as the so-called waggle dance) are modeled. These models are then used to recognize future instances of each behavior, in essence identifying the type of each bee based on which model fits its actions best. Chapter 6 is concerned with detecting and categorizing four types of interactions which frequently occur among the ants being observed. Both of these phenomena are studied by biologists would could benefit from being able to generate automatically generated data from their videos.

In both applications, the first step is to create tracks from the input video data and then generate several features from the raw trajectories of each tracked insect. The bee behavior modeler determines the motion of each bee during every frame of data using kernel regression. On the other hand, the ant interaction detector uses straightforward thresholding (learned from the training data) to determine instances of interactions. Both modelers then use hidden Markov models (trained automatically) to smooth the data, eliminating noisy readings and determining a more likely sequence of labels. Finally, the bee behavior recognizer chooses the bee's behavior by picking the HMM (from a total of one per behavior type) most likely to have generated the bee's motion series.

A total of 93% of the bees' motions are correctly labeled, resulting in approximately 80% of the bees labeled with the correct behaviors in the test data. While far from perfect, this partially achieves the goal of recognizing instances of bees performing a waggle dance. Likewise, 84.7% of all interactions between nearby ants are correctly detected and classified. Further testing is necessary to determine the precision of this process, but if a similar percent of interactions are recognized in all datasets, then these results provide exactly the information required to eliminate the need for human labeling.

The focus of most behavior recognition research is on identifying individual behaviors of sole agents, often acting in isolation from others. In contrast, the behaviors detected by this research are socially motivated. They rely on and are guided by the presence of others. Further, the behaviors detected in Chapter 6 specifically require multiple participants to occur. Although computer science has received much inspiration from various biological systems, this has been a one way flow. Instead, this research seeks to apply intelligent systems techniques to domains studied in biology, an area that has not received much attention from computer science researchers.

8.4 Social Primate Family Hierarchy Detector

Different types of interactions can mean different things in a group of social animals. For instance, friendly interactions often indicate familial behavior, while aggressive interactions are a sign of disparity between the social statuses of the participants. On the other hand, there are frequently interactions which occur for more than one reason. An example of this in rhesus monkey colonies is the grooming behavior, which may indicate friendship, but can also be used to gain protection from a superior. In such cases, it is useful to know the social structure of the colony being studied. Knowing the hierarchical structure is also important when studying a variety of other areas (such as intelligence and learning capabilities). Because some technical difficulties have delayed the testing of tracking on real monkey subjects, this research learns the social structure of a group of humans following the strict behavioral guidelines which govern monkey colony life.

First, individual interactions are detected whenever two people are located in spatial proximity over a short temporal window. Specific features are extracted for each interaction and used with a kernel regression classifier to determine which type of interaction is taking place. Once all the interactions are thus detected and classified, the aggressive interactions are used to learn the hierarchical ranking of each family, relative to the others.

The results of learning the social structure in this experiment are promising. The correct hierarchy is learned with only one pair of swapped families between the three test datasets involving hand labeled tracks. Further, when applied to the actual automatically generated and identified tracks, the algorithm performed similarly, this time reversing a single pair of families in each of two of the three test datasets. Finally, when the two non-training, automatically created datasets are combined, the system successfully learns the correct social ranking. Given the vast quantity of data available for a real-life learning attempt, this algorithm should perform well on such data.

As with the rest of the modeling and recognition work presented in this dissertation, learning the social structure of a monkey colony (whether or not through simulation with people) is based upon detecting and understanding behaviors involving social interactions. This is contrary to most behavior recognition research, which concerns itself with modeling isolated behaviors of solitary individuals. Additionally, this research contributes the application machine learning techniques to a new biological domain with practical implications. It also validates the tracking and identification techniques developed in Chapters 4 and 5 in a new, relevant domain.

8.5 Future Directions

One important extension of this work is to apply the tracking and identification algorithms in the actual environments in which they can provide useful information. Specifically, this includes real sporting events and actual non-human primates. To be able to gather such data, the main hurdle to overcome is to find placement of the RFID tags which is acceptable to all participants. Additionally, some way to deal with the 3-d nature of the monkeys' trajectories must be found. Testing in these real world situations will move us one step closer to tracking humans in an unconstrained environment for safety and security.

Another direction for future work is to apply these learning techniques to data from actual monkey colony interactions. In addition to determining the family rankings, a logical extension is to be able to learn the families themselves, as there are many situations in which they are unknown beforehand. Preliminary work to this end has already been started, with Figure 8.1 showing the families learned in one early "human experiment." While the algorithm currently requires a priori knowledge of the size of each family, in the future this can be made more generalized and applied to actual non-human data.

Finally, the author hopes that other researchers will find this work to be useful in new domains. While the research was designed with an eye towards sports and social animal domains, there are a large number of other areas which can benefit from the

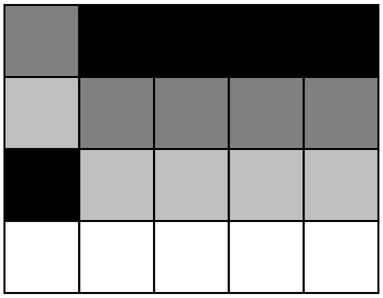


Figure 8.1: Graphical depiction of learned social structure assignments from combined run1t, run2t, run 3t. Each row is a family (as learned), with each square shaded according to the correct family. One alpha family member is misidentified as a gamma member, while one gamma is wrongly identified as a beta and one beta as an alpha, for a total of 3 incorrect out of 20.

methods introduced here, including other human systems and even robotics. Perhaps all of these areas can benefit from using observations to recognize various behaviors.

8.6 Conclusion

The research detailed in this dissertation seeks to use observations to automatically model and recognize the behaviors of a variety of multi-agent systems. Applications of this work currently or potentially include both human (sports, safety, and surveillance) and non-human (bees, ants, and monkeys) systems. Several machine learning techniques have been explored and adapted to create new algorithms for solving the tracking, identification, and modeling problems involved in these domains. Presented is a method of using multiple laser range finders and active RFID tags to track and identify targets, as well as several algorithms for learning and interpreting the behavior of those targets as they interact with one another.

REFERENCES

Arkin, E.M., Meijer, H., Mitchell, J.S.B., Rappaport, D., and Skiena, S. (1993). Decision Trees for Geometric Models. International Journal of Computational Geometry and Applications 8(3):343-364.

Azuma, R. (1993). Tracking Requirements for Augmented Reality. Communications of the ACM 36(7):50-51.

Bahl, P. and Padmanabhan, V.N. (2000). RADAR: An In-Building RF-Based User Location and Tracking System. In Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (Infocom).

Balch, T., Dellaert, F., Feldman, A., Guillory, A., Isbell, C., Khan, Z., Stein, A., and Wilde, H. (2005). How A.I. and Multi-Robot Systems Research Will Accelerate Our Understanding of Social Animal Behavior. In Proceedings of the IEEE 94(7):1445-1463.

Balch, T., Feldman, A., and Wilson, W. (2004). Assessment of an RFID System for Animal Tracking. Georgia Tech Technical Report GIT-CC-04-10.

Balch, T., Khan, Z., and Veloso, M. (2001). Automatically Tracking and Analyzing the Behavior of Live Insect Colonies. In Proceedings of the International Conference on Autonomous Agents.

Bengio, Y. and Frasconi, P. (1996). Input-Output HMM's for Sequence Processing. IEEE Transactions on Neural Networks 7(5):1231-1249.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). From Natural to Artificial Swarm Intelligence. New York, Oxford University Press.

Brashear, H., Starner, T., Lukowicz, P., and Junker, H. (2003). Using Multiple Sensors for Mobile Sign Language Recognition. In Proceedings of the IEEE International Symposium on Wearable Computers.

Bregler, C. (1997). Learning and Recognizing Human Dynamics in Video Sequences. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Bruce, J., Balch, T., and Veloso, M. (2000). Fast and Inexpensive Color Image Segmentation for Interactive Robots. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Cohen, I. and Medioni, G. (1999). Detecting and Tracking Moving Objects for Video Surveillance. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Coifman, B., Beymer, D., McLauchlan, P., and Malik, J. (1998). A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance. Transportation Research Part C: Emerging Technologies 6(4):271-288.

Couzin, I.D. and Franks, N.R. (2002). Self-Organized Lane Formation and Optimized Traffic Flow in Army Ants. In Proceedings of The Royal Society B: Biological Sciences 270:139-146.

Curtin, J., Kauffman, R.J., and Riggins, F.J. (2007). Making the 'MOST' Out of RFID Technology: A Research Agenda for the Study of the Adoption, Usage and Impact of RFID. Information Technology and Management 8(2) 87-100.

Darrell, T., Essa, I.A., and Pentland, A.P. (1996). Task-Specific Gesture Analysis in Real-Time Using Interpolated Views. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(12):1236-1242.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte Carlo Localization for Mobile Robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Dorigo, M. and Caro, G.D. (1999). The Ant Colony Optimization Meta-Heuristic. New Ideas in Optimization. McGraw-Hill.

Drea, C.M. and Wallen, K. (1999). Low-Status Monkeys 'Play Dumb' When Learning in Mixed Social Groups. In Proceedings of the National Academy of Sciences 96(22):12965-12969.

Egerstedt, M., Balch, T., Dellaert, F., Delmotte, F., and Khan, Z. (2005). What Are the Ants Doing? Vision-Based Tracking and Reconstruction of Control Programs. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Feldman, A., Adams, S., Hybinette, M., Balch, T. (2007). A Tracker for Multiple Dynamic Targets Using Multiple Sensors. Video Submission at the IEEE International Conference on Robotics and Automation (ICRA).

Feldman, A. and Balch, T. (2003). Automatic Identification of Bee Movement Using Human Trainable Models of Behavior. International Workshop on the Mathematics and Algorithms of Social Insects.

Feldman, A. and Balch, T. (2004a). Representing Honey Bee Behavior for Recognition Using Human Trainable Models. Adaptive Behavior 12(3-4):241-250.

Feldman, A. and Balch, T. (2004b). Modeling Honey Bee Behavior for Recognition Using Human Trainable Models. In Proceedings of Modeling Other Agents from Observations (MOO), an AAMAS'04 Workshop.

Ferris, B., Hähnel, D., and Fox, D. (2006). Gaussian Processes for Signal Strength-Based Location Estimation. In Proceedings of the Robotics: Science and Systems Conference (RSS).

Finkenzeller, K (2000). RFID Handbook: Radio-Frequency Identification Fundamentals and Applications. Wiley & Sons.

Fod, A., Howard, A., and Matarić, M. (2002). Laser-Based People Tracking. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA).

Fox, D., Thrun, S., Burgard, W., and Dellaert, F. (2001). Particle Filters for Mobile Robot Localization. Sequential Monte Carlo Methods in Practice. New York, Springer-Verlag.

Friedman, J., Bentley, J., and Finkel, R.A. (1976). An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Transactions on Mathematical Software 3(3):209-226.

Frisch, K.v. (1967). The Dance Language and Orientation of Bees. Cambridge, Harvard University Press.

Gonzales, J., Ollero, A., and Reina, A. (1994). Map Building for a Mobile Robot Equipped with a 2D Laser Range Finder. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA).

Gordon, D. (1999). Ants at Work. New York, The Free Press.

Gutmann, J.-S., Burgard, W., Fox, D., and Konolige, K. (1998). An Experimental Comparison of Localization Methods. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Gutmann, J.-S., Hatzack, W., Herrmann, I., Nebel, B., Rittinger, F., Topor, A., and Weigel, T. (2000). CS Freiburg Team: Playing Robotic Soccer Based on an Explicit World Model. AI Magazine 21:37-46.

Haeberlen, A., Flannery, E., Ladd, A.M., Rudys, A., Wallach, D.S., and Kavraki, L.E. (2004). Practical Robust Localization Over Large-Scale 802.11 Wireless Networks. In Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM).

Hähnel, D., Burgard, W., Fox, D., Fishkin, K., and Philipose, M. (2004). Mapping and Localization with RFID Technology. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA).

Han, K. and Veloso, M. (1998). Reactive Visual Control of Multiple Non-Holonomic Robotic Agents. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA).

Han, K. and Veloso, M. (1999). Automated Robot Behavior Recognition Applied to Robotic Soccer. In Proceedings of IJCAI-99 Workshop on Team Behaviors and Plan Recognition.

Harter, A., Hopper, A., Steggles, P., Ward, A., and Webster, P. (2001). The Anatomy of a Context-Aware Application. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM).

Hightower, J. and Borriello, G. (2001). Location Systems for Ubiquitous Computing. Computer 34(8):57-66.

Howard, A., Siddiqi, S., and Sukhatme, G. (2003). An Experimental Study of Localization Using Wireless Ethernet. In Proceedings of the International Conference on Field and Service Robotics.

Intille, S.S., and Bobick, A.F. (1995). Closed-World Tracking. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

Intille, S.S., and Bobick, A.F. (1999). A Framework for Recognizing Multi-Agent Action from Visual Evidence. In Proceedings of the National Conference on Artificial Intelligence (AAAI).

Ivanov, Y., Stauffer, C., Bobick, A., Grimson, W.E.L. (1999). Video Surveillance of Interactions. In Proceedings of the CVPR'99 Workshop on Visual Surveillance.

Jelinek, F. (1998). Statistical Methods for Speech Recognition. Cambridge, MIT Press.

Jung, B., and Sukhatme, G. (2001). Tracking Multiple Moving Targets Using a Camera and Laser Rangefinder. Institute for Robotics and Intelligent Systems (IRIS) Technical Report IRIS-01-397.

Jung, B., and Sukhatme, G (2002). Tracking Targets Using Multiple Robots: The Effect of Environment Occlusion. Autonomous Robots 13(3):191-205.

Kanazawa, K., Koller, D., and Russell, S.J. (1995). Stochastic Simulation Algorithms for Dynamic Probabilistic Networks. In Proceedings of the Conference on Uncertainty in AI (UAI).

Kantor, G. and Singh, S (2002). Preliminary Results in Range-Only Localization and Mapping. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA).

Khan, Z., Balch, T., and Dellaert, F. (2003). Efficient Particle Filter-Based Tracking of Multiple Interacting Targets Using an MRF-Based Motion Model. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Khan, Z., Balch, T., and Dellaert, F. (2004a). A Rao-Blackwellized Particle Filter for EigeTracking. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Khan, Z., Balch, T., and Dellaert, F. (2005) MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(11):1805-1918.

Khan, Z., Herman, R. A., Wallen, K., and Balch, T. (2004b). An Outdoor 3-d Visual Tracking System for the Study of Spatial Navigation and Memory in Rhesus Monkeys. Behavior Research Methods, Instruments & Computers 37(3):453-63.

Kornienko, L. and Kleeman, L. (2007). An Autonomous Human Body Parts Detector Using a Laser Range-Finder. Australasian Conference on Robotics and Automation.

Kuhn, H.W. (1955). The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly 2:83-87.

Ladd, A.M., Bekris, K.E., Rudys, A., Marceau, G., Kavraki, L.E., and Wallach, D.S. (2002). Robotics-Based Location Sensing Using Wireless Ethernet. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM).

Letchner, J., Fox, D., and LaMarca, A. (2005). Large-Scale Localization from Wireless Signal Strength. In Proceedings of the National Conference on Artificial Intelligence (AAAI).

Lorincz, K. and Welsh, M. (2007). MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. Personal and Ubiquitous Computing 11(6):489-503.

Lymberopoulos, D., Lindsey, Q., and Savvides, A. (2006). An Empirical Characterization of Radio Signal Strength Variability in 3-D IEEE 802.15.4 Networks Using Monopole Antennas. Embedded Networks and Applications Lab (ENALAB) Tech Report 050501.

Mallon, E., Pratt, S., and Franks, N. (2001). Individual and Collective Decision-Making During Nest Site Selection by the Ant Leptothorax Albipennis. Behavioral Ecology and Sociobiology 50(4):352-359.

Matarić, M. (1997). Learning Social Behavior. Journal of Robotics and Autonomous

Systems 20(2):191-204.

Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences. International Journal of Computer Vision 3(3):209-238.

Mitchel, T. (1997). Machine Learning. Boston, MIT Press & McGraw-Hill.

Mittal, A. and Davis, L. (2003). M_2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. International Journal of Computer Vision 51(3):189-203.

Oh, S.M., Rehg, J.M., Balch, T., and Dellaert, F. (2005). Data-Driven MCMC for Learning and Inference in Switching Linear Dynamic Systems. In Proceedings of the National Conference on Artificial Intelligence (AAAI).

Oh, S.M., Rehg, J.M., Balch, T., and Dellaert, F. (2008). Learning and Inferring Motion Patterns Using Parametric Segmental Switching Linear Dynamic Systems. International Journal of Computer Vision 77(1-3):103-124.

Panangadan, A.M., Matarić, M., and Sukhatme, G. (2004). Detecting Anomalous Human Interactions Using Laser Range-Finders. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Perš, J. and Kovacic, S. (2000). Computer Vision System for Tracking Players in Sports Games. In Proceedings of the International Workshop on Image and Signal Processing and Analysis (IWISPA).

Philipose, M., Fishkin, K., Fox, D., Kautz, H., Patterson, D., and Perkowitz, M. (2003). Guide: Towards Understanding Daily Life via Auto-Identification and Statistical Analysis. In Proceedings of the International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications (Ubi-health).

Pingali, G.S., Jean, Y., and Carlbom, I. (1998). Real Time Tracking for Enhanced Tennis Broadcasts. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Prassler, E., Scholz, J., and Elfes, A. (1999). Tracking People in a Railway Station During Rush-Hour. Computer Vision Systems. Berlin, Springer.

Pratt, S.C. (2005). Quorum Sensing by Encounter Rates in the Ant Temnothorax Curvispinosus. Behavioral Ecology 16: 488-496.

Pratt, S.C., Mallon, E.B., Sumpter, D.J.T., and Franks, N.R. (2002). Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant Leptothorax albipennis. Behavioral Ecology and Sociobiology 52(2):117-127.

Rabiner, L.R. (1988). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In Proceedings of the IEEE 77(2):257-286.

Reid, D.B. (1978). An Algorithm for Tracking Multiple Targets. IEEE Transactions on Automatic Control 24(6):843-854.

Rosales, R. and Sclaroff, S. (1998). Improved Tracking of Multiple Humans with Trajectory Prediction and Occlusion Modeling. In Proceedings of the CVPR Workshop on the Interpretation of Visual Motion.

Rosin, P.L. and Ellis, T. (1995). Image Difference Threshold Strategies and Shadow Detection. In Proceedings of the British Conference on Machine Vision.

Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., and Sievänen, J. (2002). A Probabilistic Approach to WLAN User Location Estimation. International Journal of Wireless Information Networks 9(3):155-164.

Schino, G. and Aureli, F. (2008). Grooming Reciprocation Among Female Primates: A Meta-Analysis. Biology Letters 4(1):9-11.

Schleidt, W., Yakalis, M., Donnelly, M., and McGarry, J. (1984) A Proposal for a Standard Ethogram, Exemplified by an Ethogram of the Bluebreasted Queil (Coturnix Chinensis). Zeitschrift für Tierpsychologie 64(3-4):193-220.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2003a). People Tracking with a Mobile Robot Using Sample-Based Joint Probabilistic Data Association Filters. The International Journal of Robotics Research 22(2):99-116.

Schulz, D., Fox, D., and Hightower, J. (2003b). People Tracking with Anonymous and ID-Sensors Using Rao-Blackwellised Particle Filters. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).

Seeley, T. (1995). The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies. Cambridge, Harvard University Press.

Shepard, D. (1968). A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. In Proceedings of the ACM National Conference.

Singh, S., Kantor, G., and Strelow, D. (2002). Recent Results in Extensions to Simultaneous Localization and Mapping. In Proceedings of the International Symposium on Experimental Robotics.

Schneider, M. (2003). Radio Frequency Identification (RFID) Technology and its Applications in the Commercial Construction Industry. Master's Thesis, Civil Engineering Department, University of Kentucky, Lexington.

Smola, A.J. and Schölkopf, B. (1997). On a Kernel-Based Method for Pattern Recognition, Regression, Approximation, and Operator Inversion. Algorithmica 22(1-2):211-231.

Starner, T., Weaver, J., Pentland, A. (1998). 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.

Stillman, S., Tanawongsuwan, R., and Essa, I. (1998). A System for Tracking and Recognizing Multiple People with Multiple Cameras. Georgia Tech Technical Report #GIT-GVU-98-25.

Stroupe, A. and Balch, T. (2003). Value-Based Observations with Robot Teams (VBORT) for Dynamic Targets. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Surmann, H., Lingemann, K., Nüchter, A., and Hertzberg, J. (2001). A 3D Laser Range Finder for Autonomous Mobile Robots. In Proceedings of the International Symposium on Robotics.

Swain, M. and Ballard, D. (1991). Color Indexing. International Journal of Computer Vision 7(1):11-32.

Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: Principles and Practice of Background Maintenance. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

Veloso, M., Stone, P., and Han, K. (1998). The CMUnited-97 Robotic Soccer Team: Perception and Multiagent Control. In Proceedings of the International Conference on Autonomous Agents.

Vogt, H. (2002). Multiple Object Identification with Passive RFID Tags. IEEE International Conference on Systems, Man and Cybernetics.

Wallen, K. (2005). Hormonal Influences on Sexually Differentiated Behavior in Nonhuman Primates. Frontiers in Neuroendocrinology 26(1):7-26.

Wang, S., Chen, W., Ong, C., and Chuang, Y. (2006). RFID Applications in Hospitals: A Case Study on a Demonstration RFID Project in a Taiwan Hospital. In Proceedings of the Hawaii International Conference on System Sciences (HICSS).

Want, R., Hopper, A., Falcão, V., and Gibbons, J. (1992). The Active Badge Location System. ACM Transactions on Information Systems 40(1):91-102.

Welch, G. and Bishop, G. (2004). An Introduction to the Kalman Filter. Chapel Hill, NC, UNC.

Westeyn, T., Vadas, K., Bian, X., Starner, T., and Abowd, G.D. (2005). Recognizing Mimicked Autistic Self-Stimulatory Behaviors Using HMMs. In Proceedings of the IEEE International Symposium on Wearable Computers (ISWC).

Yan, H. and Matarić, M.J. (2002). General Spatial Features for Analysis of Multi-Robot and Human Activities from Raw Position Data. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Ye, C. and Borenstein, J. (2002). Characterization of a 2-D Laser Scanner for Mobile Robot Obstacle Negotiation. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

Youssef, M., Agrawala, A., and Shankar, A. (2003). WLAN Location Determination via Clustering and Probability Distributions. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications.

Zhao, H. and Shibasaki, R. (2004). A Novel System for Tracking Pedestrians Using Multiple Single-Row Laser Range Scanners. IEEE Transactions on Systems, Man and Cybernetics, Part A 35(2):283-291.

Zhao, T. and Nevatia, R. (2003). Bayesian Human Segmentation in Crowded Situations. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).