Automatic Identification of Bee Movement Using Human Trainable Models of Behavior

Adam Feldman[1], **Tucker Balch**[2]

1. Georgia Institute of Technology, Atlanta, Georgia 30332, USA. *Corresponding author*: storm@cc.gatech.edu
2. Georgia Institute of Technology, Atlanta, Georgia 30332, USA.

Abstract

Identifying and recording subject movements is a critical, but time-consuming step in animal behavior research. The task is especially onerous in studies involving social insects because of the number of animals that must be observed simultaneously. To address this, we present a system that can automatically analyze animal movements, and label them, on the basis of examples provided by a human expert. Further, in conjunction with identifying movements, our system also recognizes the behaviors made up of these movements. Thus, with only a small training set of hand labeled data, the system automatically completes the entire behavioral labeling process. For our experiments, activity in an observation hive is recorded on video, that video is converted into location information for each animal by a vision-based tracker, and then numerical features such as velocity and heading change are extracted. The features are used in turn to label the sequence of movements for each observed animal. Our approach uses a combination of k-nearest neighbor (KNN) classification and hidden Markov model (HMM) techniques. The system was evaluated on several hundred honey bee trajectories extracted from a 15 minute video of activity in an observation hive. Additionally, simulated data and models were used to test the validity of the behavioral recognition techniques.

Keywords: Behavior Recognition, Bee Movements, Hidden Markov Models, Biological Inspiration.

1. Introduction

A honey bee colony is a "superorganism" – a system composed of thousands of simple individuals that exhibits apparently intelligent behavior. As such, honey bees are popular subjects of study for behavioral neurobiologists and behavioral ecologists who seek to understand how system level behavior emerges from the activities of thousands of interacting individual animals. Currently, when a researcher studies honey bee colony behavior, animals in an observation hive are videotaped and the resulting tape is viewed and hand-labeled [Seeley 1995]. Typically this requires the observer to watch the video many times, and is a rather time-consuming process. If honey bee behaviors could be recognized and identified automatically, research in this area could be greatly accelerated.

Our objective is to develop a system that can learn to label behavior automatically on the basis of a human expert's labeling of example data. This could save the researcher time, which could be better used by the researcher evaluating the automatically labeled data.

The behaviors of interest are sequential activities that consist of several physical motions. For example, bees commonly perform waggle dances (see Figure 1). 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 we have focused on dancing, following, and active hive work as behavioral roles to be identified.

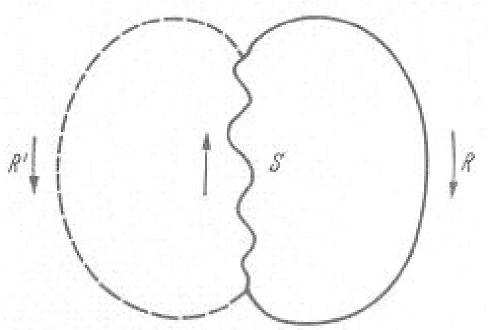


Figure 1: Pattern of waggle dance: S is waggling segment, R and R' are return runs (alternating between left and right). (Note: we are seeking permission to use this image).

We define these behaviors 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. We distinguish **behaviors** from constituent **motions** they are composed of. *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, if we know which behavior a bee is executing, we can better identify the constituent motions.

The system described in this paper 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 our system. First, bees in an observation hive are videotaped. Then, a tracker extracts x- and y-coordinate information for each bee [Bruce et al 2000]. From raw location data, quantitative features of motion (such as velocity and heading change) are computed. A *k*-nearest neighbor (KNN) 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 detail below, 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 our 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 we can leverage in this work. In our system, the output from the KNN classifier is used as input to the Viterbi algorithm over a fully connected HMM. 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 designate an **ARCING_LEFT**. The HMM technique will correct mistakes of this nature. The HMM is also used to determine the 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.

Our hypothesis is that this system will 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 will be possible to infer the correct behavior (dancer, follower, etc).

2. Background and Related Work

2.1 *k*-Nearest Neighbor

The *k*-nearest neighbor (kNN) classifier is a classification technique that classifies data points based on their location in an n-dimensional feature space (where n is the number of features). For training, and evaluation, a data set to be classified is broken into two parts – a training set and a test set. The training set is manually labeled, and is used to "train" the system to be able to classify the rest of the data (the test set). Values are first normalized so that every dimension of the feature space is uniform (such as from 0 to 1).

Classification works by evaluating each test set point in the populated feature space and finding the *k* nearest neighbor points (geometrically). Each possible label for the test point is scored according to distance to each of the *k* points. The score is incremented by a value proportional to the inverse squared distance from the test point. After scoring, whichever label has the highest score is the label that is given to the test point. In this way, test set points are classified based on the labels of the points that they are near in the feature space [Mitchell 1997].

This technique works well, but has a limitation when applied to our application. *k*-nearest neighbor considers each data point individually, without considering the sequence of data points as a whole. Therefore, for our application, we also employ hidden Markov models to take advantage of this additional contextual information.

2.2 Hidden Markov Models

If we assume an observed agent acts according to a Markov model, we can employ HMM-based approaches to identify its behavior. Hidden Markov models (HMMs) can be used as models of sequenced behavior. They consist of a series of states, observations and transitions. The states represent the topography of the model, with the model being in one state at any given time. We assume that the animals we are observing act according to a Markov model, where each state in the model corresponds to a motion, and sequences of motions are behaviors. As we observe the animal however, we make certain observation errors (thus it is a Hidden model). The observations correspond to the output of the system being modeled. For each state, there is some probability for each possible observation occurring. Additionally, HMMs require a probability table for the initial state. This table is the probability of beginning any sequence in each state. An HMM can be thought of as a graph where each state is a node and each transition with non-zero probability is a link. An HMM's topology reflects the topology of the behavior it models (e.g. the top diagram in Figure 4 models a waggle dance).

Once the parameters of an HMM are provided (or learned), it can be used to answer the following question: "Given an observation sequence, what is the most likely state sequence that created it?" We use the Viterbi algorithm to do this [Rabiner 1989]. The Viterbi algorithm takes as input a specified HMM (states, observations, and probability distributions) and an observation sequence in order to compute the most likely state sequence.

2.3 Related Work

Traditionally, hidden Markov models have been used in speech recognition tasks. However, they can also be used in gesture recognition tasks. Unfortunately, most available HMM toolkits are geared for speech recognition, and require adapting for general gesture recognition. In light of this, the Georgia Tech Gesture Toolkit **GT^{2}k** [Westeyn et al 2003] was created. The GTK was designed as an all-purpose gesture recognition toolkit, and supports such projects as American Sign Language recognition [Brashear et al 2003].

Another type of behavior recognition was studied by Kwun Han and Manuela Veloso [Han and Veloso 1999]. They examined identifying the behavior of autonomous robots, as applied to robotic soccer. Their framework uses hidden Markov models to recognize the behaviors of the robotic agents.

Couzin and Franks [Couzin and Franks 2003] have used video tracking techniques to identify certain movements in ants in order to understand their behavior. Our work is distinct in that our system can learn from suggestions given by an expert.

3. Approach

Our system is composed of several components. Figure 2 provides an overview, illustrating the flow of data from one component to the next. First, a video camera records bees in the observation hive. This video is passed to a tracker, which extracts coordinate information to be used by the human labeler (creating the training set) and then by the kNN classifier. The output of the kNN classifier is used as an observation sequence by the Viterbi algorithm (with an HMM) to generate the most likely state sequence. This final sequence is the labels determined by the system.

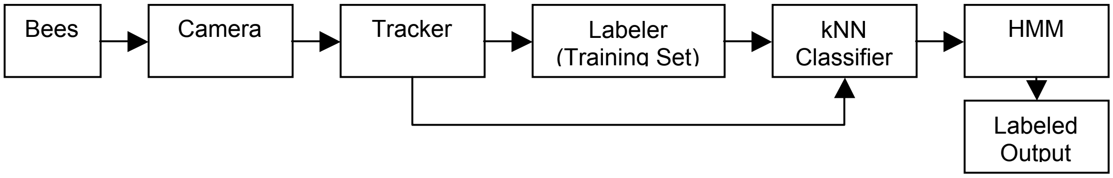


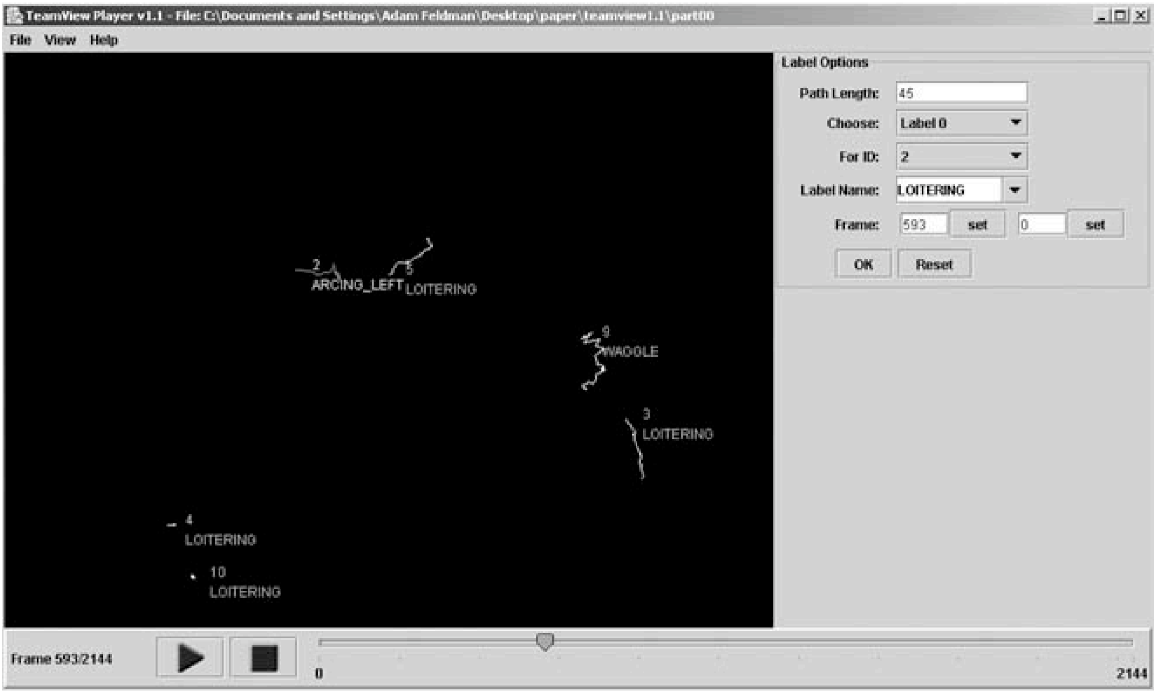
Figure 2: Overview of our system.

3.1 Tracker

Tracking software is necessary to convert the bee videos into data that can be used by other software [Bruce et al 2000 and Khan et al 2003]. In our experiments, 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, we now have 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.

3.2 TeamView

The TeamView software (Figure 3) is used to visualize and hand label the data sets. The files that contain the x- and y- coordinate information (from the 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.



Path Length: 45
Choose: Label 0
For ID: 2
Label Name: LOITERING
Frame: 593 0
Frame 593/2144

Figure 3: 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.

3.3 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

3.4 *k*-Nearest Neighbor Classification

Before kNN 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 who 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.

In order to determine which features are helpful and which are useless (or harmful) in determining the label of a data point, we conducted a sensitivity analysis. Every combination of the seven available features – from each one individually to all seven together – was tested by applying the kNN 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) were considered the most useful, and are the only features used in the rest of the experiments.

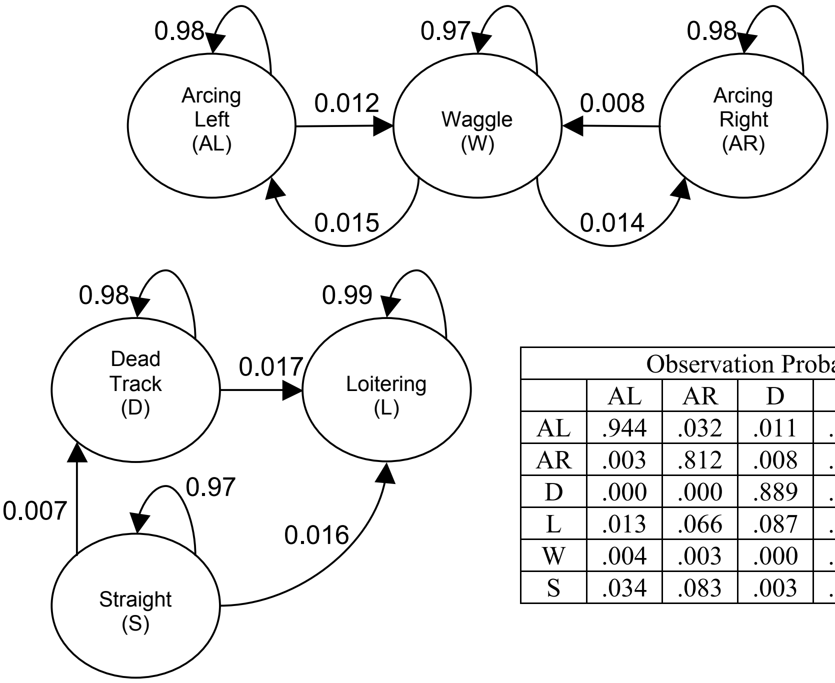
In our experiments, the training set is made up of 1000 points of each type of labeled motion. This ensures fair representation, despite frequency disparities among the labels (unlike some other methods of selecting the training set). The importance of this can be found in the infrequency of our most 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, kNN 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 kNN classification. The advantage of this technique over traditional kNN 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.

3.5 Hidden Markov Model

The kNN 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. What we want is to "smooth" these places where the data isn't representative of what is really going on. Since the kNN classifier only considers each point individually, this time series information is lost. Thus, we turn to Hidden Markov Models.

Although many HMMs use a specific topology, we used a fully connected HMM, as we would like our system to learn this topology automatically. Instead, we want to use the HMM to statistically smooth the labels we have already determined with the kNN classifier. Therefore, we connect all of the states, and use the training data to determine the probability of each transition (see Figure 4). It should be noted that this technique may result in certain transition probabilities to drop to zero, causing the HMM to no longer be fully connected.



Observation Probabilities						
	AL	AR	D	L	W	S
AL	.944	.032	.011	.095	.004	.031
AR	.003	.812	.008	.095	.003	.044
D	.000	.000	.889	.087	.000	.002
L	.013	.066	.087	.648	.000	.029
W	.004	.003	.000	.000	.987	.004
S	.034	.083	.003	.071	.003	.888

Figure 4: Possible HMM, after removing transitions with a probability less than 0.005, and observation probability table determined in the experiment discussed below. Cell (x, y) of the observation table shows the probability of observing x while being in state y.

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.

The observation sequence given to the algorithm is actually the output from the kNN classifier. For each iteration (corresponding to one time step in the dataset), the HMM multiplies the values in the observation probabilities table by the scores from the kNN (and then normalizes) to create the new table that it will use for that iteration. In this way, the HMM is able to take advantage of the added information that the kNN records.

3.5.1 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. Our system, however, completes these two tasks in parallel, allowing each to assist the other. This is done 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 (See Figure 5).

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.

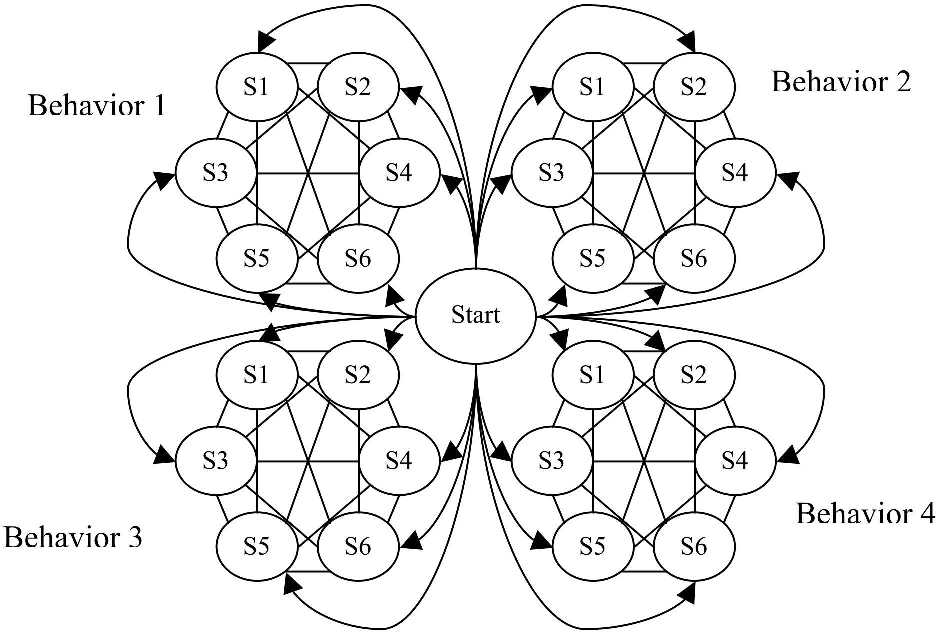


Figure 5: Behavioral HMM, which is made up of a start state and the four sub-models, one for each behavior.

3.6 Methods

To test this classification system, we began with a data set consisting of fifteen minutes of video of honey bee behavior in an observation hive. The tracker was used to extract the features, while TeamView was used for hand labeling. There were three human labelers, each labeling five minutes (1/3) of the data. The data was then broken into a training set, consisting of the last one third of the data, and a test set, consisting of the first two thirds. The test set was put aside for accuracy validation after training the system.

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

To establish the accuracy of the system, these 6000 points in feature space and HMM parameters were 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.

Finally, to test the techniques employed by this system, two additional experiments were conducted. First, artificial data was generated following the behavioral models created from the actual bee data. By generating this data, we were able to control the amount of noise present in the features. Data was generated by defining a locus (between -1 and 1 along each dimension) for each of the six class of points, then adding a random amount of Gaussian noise to each dimension (feature). The noise level parameter, n, was equal to 3 standard deviations, such that the vast majority of noise added was between –n and n.

Second, we created artificial "behavioral models" which did not correspond directly to any behaviors, but could be used to demonstrate the effectiveness of the technique in cases when the behavioral models were very distinct from one another. These models were manually created to be very different from each other. Using these models we generated artificial data (as described above), varying the noise level to test effectiveness. For simplicity, these models contain only five states. The transition tables for each of the four simulated behaviors are in Table 1.

Table 1: Transition tables of the four simulated behavioral models.

	State 1	State 2	State 3	State 4	State 5
State 1	0.50	0.50	0.00	0.00	0.00
State 2	0.00	0.50	0.50	0.00	0.00
State 3	0.00	0.00	0.50	0.50	0.00
State 4	0.00	0.00	0.00	0.50	0.50
State 5	0.50	0.00	0.00	0.00	0.50

	State 1	State 2	State 3	State 4	State 5
State 1	0.50	0.50	0.00	0.00	0.00
State 2	0.00	0.50	0.00	0.00	0.50
State 3	0.00	0.00	0.50	0.50	0.00
State 4	0.33	0.00	0.33	0.33	0.00
State 5	0.00	0.00	0.50	0.00	0.50

	State 1	State 2	State 3	State 4	State 5
State 1	0.00	0.50	0.00	0.00	0.50
State 2	0.00	0.50	0.50	0.00	0.00
State 3	0.00	0.00	0.50	0.50	0.00
State 4	0.50	0.00	0.00	0.50	0.00
State 5	0.00	0.00	0.50	0.00	0.50

	State 1	State 2	State 3	State 4	State 5
State 1	0.50	0.00	0.00	0.00	0.50
State 2	0.00	0.50	0.00	0.00	0.50
State 3	0.00	0.00	0.50	0.50	0.00
State 4	0.00	0.33	0.00	0.33	0.33
State 5	0.50	0.00	0.00	0.00	0.50

Table 2: Fractional breakdown of accuracy, first with the kNN 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 HMM)	Total Occurrences in test set
ARCING_LEFT	0.791	0.866	2059
ARCING_RIGHT	0.688	0.715	2407
DEAD_TRACK	0.901	0.665	5920
LOITERING	0.744	0.812	113285
WAGGLE	0.591	0.511	1550
STRAIGHT	0.371	0.461	5343
Total	0.733	0.787	130564

4. Results

4.1 Feature Selection

Every combination of the seven available features was tested by applying the *k*-nearest neighbor algorithm to a large training set. This resulted in 127 possibilities (zero features was not an option). The combination of features that resulted in the highest accuracy (defined as the percent of the test points labeled correctly) is h2, v1, and sh1. Therefore we consider only these features in the rest of the experiments.

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

4.2 Classification Results

Table 2 shows the fractional accuracy of the system for each label type. As indicated, the system achieved an overall accuracy of about 78.7%. Further, the overall accuracy increased by over 5.5% by including the use of the HMM to "smooth" the results of the kNN classifier. Finally, the accuracy in determining the behavior was 45.9%.

5. Discussion

As we hypothesized, the use of an HMM in conjunction with a kNN classifier provides higher accuracy than a kNN classifier alone. The HMM improved overall accuracy by 5.5%, above the 73.3% accuracy of only the kNN. 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 92%.

Another label that caused many problems for the system was **STRAIGHT**. This label was included because we wanted 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 81.2% (or about 95% after combining **LOITERING** and **DEAD_TRACK**).

Table 3 shows the performance of the system on simulated data, generated by following the models created from the actual bee data. The system performed reasonably well with low noise levels, yet accuracy dropped off quickly with rising noise levels.

We hypothesize that these errors arise because the four behaviors are so like one another. This means that the transition probability table for each behavior is very similar to the transition probability tables of the other behaviors. To test this hypothesis, we created artificial models with parameters we could vary to see which affected performance.

Table 4 shows the system's accuracy using simulated, distinct behavior models. In this case, the system achieved a high accuracy, reaching almost 90%, even with high noise levels.

Table 3: Accuracy of simulated data (using real behavioral models)

Noise Level	Motion Accuracy	Behavior Accuracy
0.00	85.0%	85.1%
0.05	85.5%	85.8%
0.10	72.5%	73.8%
0.15	52.5%	54.6%
0.20	56.5%	55.3%
0.30	46.5%	51.1%

Table 4: Accuracy of simulated data (using synthetic behavioral models)

Noise Level	Motion Accuracy	Behavior Accuracy
0.30	73%	100%
0.40	63%	90%
0.50	50%	89%

The artificial models and data show that the technique is sound and that the system can perform as desired. By using this artificial data, we can see that the root of the problem lies in the fact that the models for the different behaviors are too similar. This is illustrated in the accuracy with this simulated data – even with high levels of noise, the behavior recognition is around 90% accurate. Further, the system correctly identifies most behaviors (which is the ultimate purpose of the system) even when motion accuracy is at only 50%.

The main failing of the system can be traced to trying to differentiate between models that are too similar. We attribute this not to the behaviors being similar, but to poor training of the models. First, our experiments made use of relatively small training sets – future work will involve many more examples of each type of behavior, allowing for more accurate model generation. Second, and even more important, we assumed 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. Therefore, future work will focus on allowing a bee to change behavior throughout its existence in the data set.

Using artificial models simulated the above goals of making the behavioral models more distinct. Once this is accomplished, the system achieves reasonable performance. Thus, it successfully demonstrates the effectiveness of its techniques.

Acknowledgements

We would like to thank Zia Khan and Frank Dellaert for the software used to track the bees; and Kevin Gorham, Stephen Ingram, and Edgard Nascimento for TeamView and for hand-labeling our data. We are also grateful to Tom Seeley for his advice on observing bees, and his help in gathering data.

This project was funded under NSF Award IIS-0219850.

References

Brashear, H., T. Starner, P. Luckowicz, and H. Junker. Using multiple sensors for mobile sign language recognition. In *Proceedings of IEEE International Symposium on Wearable Computing*, page In Press, October 2003.

James Bruce, Tucker Balch, and Manuela Veloso. Fast and inexpensive color image segmentation for interactive robots. In Proceedings of IROS-2000, Japan, October 2000.

Couzin, I.D. & Franks, N.R. (2003) Self-organized lane formation and optimized traffic flow in army ants. Proc. R. Soc. Lond. B. 270, 139-146

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

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

Khan, Z., T. Balch, F. Dellaert. Efficient Particle Filter-Based Tracking of Multiple Interacting Targets Using an MRF-based Motion Model. To appear in Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03), 2003.

Mitchell, T. Machine Learning. Boston, Massachusetts: MIT Press & McGraw-Hill, 1997.

Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc. IEEE, 77(2): 257-286, 1989.

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

Westeyn, T., H. Brashear, A. Atrash, and T. Starner. Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. ICMI'03, November 5-7, 2003, Vancouver, British Columbia, Canada.