

What Are the Ants Doing? Vision-Based Tracking and Reconstruction of Control Programs

M. Egerstedt^{*}, T. Balch[†], F. Dellaert[†], F. Delmotte^{*}, and Z. Khan[†]

^{*}{magnus, florent}@ece.gatech.edu [†]{tucker, dellaert, zkhan}@cc.gatech.edu
School of Electrical and Computer Engineering The BORG Lab – College of Computing
Georgia Institute of Technology Georgia Institute of Technology
Atlanta, GA 30332, U.S.A. Atlanta, GA 30332, U.S.A.

Abstract—In this paper, we study the problem of going from a real-world, multi-agent system to the generation of control programs in an automatic fashion. In particular, a computer vision system is presented, capable of simultaneously tracking multiple agents, such as social insects. Moreover, the data obtained from this system is fed into a mode-reconstruction module that generates low-complexity control programs, i.e. strings of symbolic descriptions of control-interrupt pairs, consistent with the empirical data. The result is a mechanism for going from the real system to an executable implementation that can be used for controlling multiple mobile robots.

I. INTRODUCTION

Control and coordination of multiple mobile robots is a problem that has received a significant amount of attention during the last decade. Different types of solution strategies have emerged, either based on purely theoretical considerations [4], [12], [13], [26], or by drawing inspiration from biologically available systems, such as ants, fish, birds, or slime mold, just to name a few [1], [21], [23].

In this paper we take the biologically inspired point-of-view, but attack the problem from a control-theoretic rather than biological vantage point. Hence, we will not assume any underlying description of what the ants are doing, but rather let the control programs be generated directly from the data. This has the obvious advantage that such a system would be transferable between different biological systems and no extensive, initial biological research investment is needed. Our proposed solution will rely on two distinctly different technologies, namely vision-based tracking and automatic control generation. We will thus, in the first part of this paper (Section 2), develop novel computer vision algorithms and demonstrate their usefulness to the problem of tracking and examining encounter rates between ants in a colony of *Aphaenogaster cockerelli*.

On the control side, we continue the development begun in [3], [10], where the programs are viewed as having an information theoretic content. In other words, they can be coded more or less effectively. Within this context, one can ask questions concerning minimum complexity programs, given a particular control task. But, in order to effectively code symbols, drawn from a finite alphabet, one must be

able to establish a probability distribution over the alphabet. If such a distribution is available then Shannon's celebrated source coding theorem tells us that the minimal expected code length l satisfies $\mathcal{H}(\mathcal{M}) \leq l \leq \mathcal{H}(\mathcal{M}) + 1$, where \mathcal{M} is the set of possible modes, and where \mathcal{H} is the entropy. The main problem that we will study in the second part of this paper (Section 3) is thus how to produce an empirical probability distribution over the set of modes, given a string of input-output data, which is equivalent to establishing what modes generated the data string in the first place. We will conclude the paper in Section 4 by applying the control generation procedure to the data obtained in the first part of the paper.

II. TRACKING MULTIPLE TARGETS

We let the experimental data be gathered by video camera. Hence, our system must therefore identify and track multiple interacting targets (ants) in sequences of images. The objective is to obtain a record of the trajectories of the animals over time, and to maintain correct, unique identification of each target throughout.

The classical multi-target tracking literature approaches the problem by performing a target detection step followed by a track association step in each video image (frame). The track association step solves the problem of converting the detected positions of animals in each image into multiple individual trajectories. The multiple hypothesis tracker and the joint probabilistic data association filter (JPDAF) [6], [15] are the most influential algorithms in this class. These multi-target tracking algorithms have been used extensively. Some examples are the use of nearest neighbor tracking in [11], the multiple hypothesis tracker in [9], and the JPDAF in [22]. Recently, a particle filter version of the JPDAF has been proposed in [25].

The tracker that we will use is based on a novel multi-target particle-filter tracker based on Markov chain Monte Carlo sampling. (See [18], [19] for details.) The general operation of the tracker is illustrated in Figure 1. Each particle represents one hypothesis regarding a target's location and orientation. The hypothesis is a rectangular region approximately the same size as the ant targets. In the example, each target is tracked by 5 particles. In actual experiments we typically use the equivalent of one thousand particles per target.

The research of Magnus Egerstedt and Florent Delmotte was supported under NSF EHS Award 0207411 and NSF ECS CAREER Award 0237971. The research of Tucker Balch, Frank Dellaert and Zia Khan was supported under NSF ITR Award 0219850.

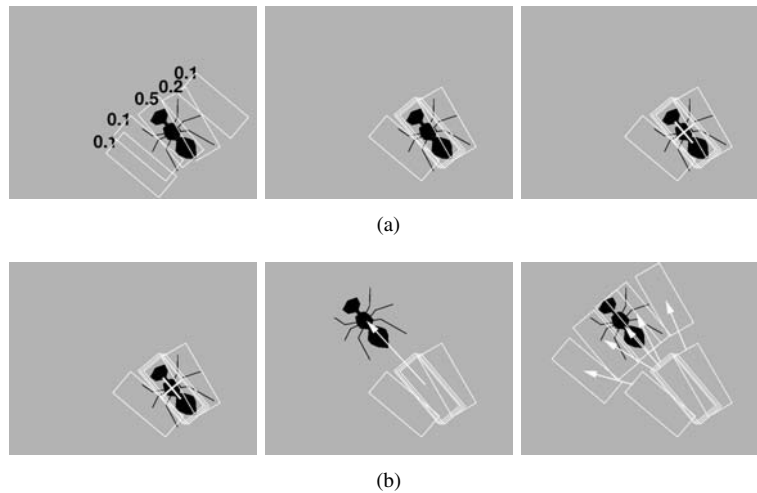


Fig. 1. Particle filter tracking. (a) A set of particles (white rectangles), are scored according to how well the underlying pixels match the appearance model (left). Particles are resampled (middle) according to the normalized weights determined in the previous step. Finally, the estimated location of the target is computed as the mean of the resampled particles. (b) Motion model: The previous image and particles (left). A new image frame is loaded (center). Each particle is advanced according to a stochastic motion model (right). The samples are now ready to be scored and resampled as above.



Fig. 2. The appearance model used in tracking ants. This is an actual image drawn from the video data.

We assume that we start with particles distributed around the target to be tracked. After initialization, the principal steps in the tracking algorithm include:

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

The algorithm just described is suitable for tracking an individual ant, but it would likely fail in the presence of many targets. Several additional extensions are necessary for multi-target tracking: First, each particle is extended to include the poses of all the targets (i.e. they are *joint* particles). Second, in the “scoring” phase of the algorithm, particles are penalized if they represent hypotheses that we know are unlikely because they violate known constraints on ant movement (e.g. ants seldomly walk on top of each

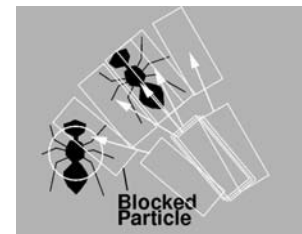


Fig. 3. Blocking. Particles that overlap the location of other tracked targets are penalized.

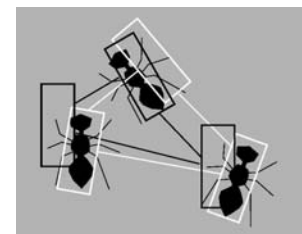


Fig. 4. Joint particles. When tracking multiple animals, we use a joint particle filter where each particle describes the pose of all tracked animals. In this figure there are two particles – one indicated with white lines, the other with black lines.

other). These extensions are reported in detail in [18], [19], and after trajectory logs have been gathered they are checked against the original video. In general, our tracker has a very low error rate (about one of 5000 video frames contains a tracking error).

The next step in our automated process is to examine the trajectory logs to find encounters. Typically, ant behaviorists describe an encounter as occurring when two ants approach each other, then experience brief or extended antennal contact. Our computer vision algorithms, however, cannot resolve the motions of ant antennae at the scale we observe the animals. If we zoomed in so close that

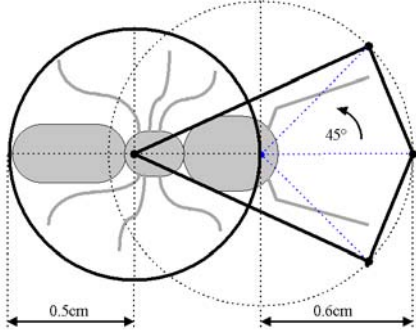


Fig. 5. Sensory field is formed by simple geometric inputs to our modeling software.

we could track their antennae, we would only be able to observe a small portion of the arena at once. So we make the simplifying assumption that interactions occur when the sensory fields of two ants overlap.

In order to do this we approximate an ant's antennal and body sensory fields with a polygon and circle, respectively (Figure 5). An encounter is inferred when one of these regions for one ant overlaps a sensory region of another ant. Our model is adapted from the model introduced for army ant simulation studies, introduced in [8].

The following are details of our sensory model for *Aphaenogaster cockerelli*. We estimate the front of the head to be a point half a body length away from the center point along the centerline. From the head, the antennae project to the left and right at 45 degrees. We project an additional point, one antenna length away, directly along the ant's centerline in front of the head. The inferred "antennal field of view" is a polygon enclosed by the center point and these other three points. We assume a body length of 1cm and antenna length of 0.6cm. We estimate an ant's "body" sensory field to include a circle centered on the ant with a radius of 0.5cm.

We assume that any object within the head sensory field will be detected and touched by the animal's antennae, and any object within the range of the body can be detected by sense organs on the legs or body of the ant. By considering separate sensory fields for the body and the antennae, we are able to classify each encounter into one of four different types: head-to-head, head-to-body, body-to-body, and body-to-head. A head-to-body encounter is one where the subject ant's head contacts the body of another. In such a case the other ant is simultaneously experiencing a body-to-head encounter.

To determine when an interaction occurs, we look for overlaps between the polygonal and circular regions described above. Computationally, determining an overlap consists of checking for intersections between line segments and circles. To gain efficiency we do not test for interactions between ants that are too far away for their fields to overlap. Furthermore, we require that two ants' orientations must differ by at least 5 degrees before an interaction can count as "head-to-head." Without this

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

III. CONTROL PROGRAM GENERATION

The idea now is to use the data obtained in the previous section in order to generate strings of control laws that are consistent with the empirical data. Such strings correspond to abstract descriptions of multi-modal control programs, and we say that such strings constitute words in a *Motion Description Language* (MDL) [7], [14], [17], [20].

Each string in a MDL corresponds to a control program that can be operated on by a given controlled dynamical system. Slightly different versions of MDLs have been proposed, but they all share the common feature that the individual atoms, concatenated together to form the control program, can be characterized by control-interrupt pairs. In other words, given a dynamical system

$$\begin{aligned} \dot{x} &= f(x, u), \quad x \in \mathbb{R}^N, \quad u \in U \\ y &= h(x), \quad y \in Y, \end{aligned} \quad (1)$$

together with a control program $(k_1, \xi_1), \dots, (k_z, \xi_z)$, where $k_i : Y \rightarrow U$ and $\xi_i : Y \rightarrow \{0, 1\}$, the system operates on this program as $\dot{x} = f(x, k_1(h(x)))$ until $\xi_1(h(x)) = 1$. At this point the next pair is read and $\dot{x} = f(x, k_2(h(x)))$ until $\xi_2(h(x)) = 1$, and so on. (Note that the interrupts can also be time-triggered, which can be incorporated by a simple augmentation of the state space.)

If we assume that the input and output spaces (U and Y respectively) in Equation (1) are finite, which can be justified by the fact that all physical sensors and actuators have a finite range and resolution, the set of all possible modes $\Sigma_{total} = U^Y \times \{0, 1\}^Y$ is finite as well. We can moreover adopt the point of view that a data point is measured only when the output or input change values, i.e. when a new output or input value is encountered. This corresponds to a so called *Lebesgue sampling*, in the sense of [2]. Under this sampling policy, we can define a mapping $\delta : \mathbb{R}^N \times U \rightarrow \mathbb{R}^N$ as $x_{p+1} = \delta(x_p, k(h(x_p)))$, given the control law $k : Y \rightarrow U$, with a new time update occurring whenever a new output or input value is encountered. For such a system, given the input string $(k_1, \xi_1), \dots, (k_z, \xi_z) \in \Sigma^*$ where $\Sigma \subseteq \Sigma_{total}$, and Σ^* denotes the set of all finite length words over Σ (see for example [16]), then the evolution is given by

$$\begin{cases} x(q+1) = \delta(x(q), k_{l(q)}(y(q))), & y(q) = h(x(q)) \\ l(q+1) = l(q) + \xi_{l(q)}(y(q)). \end{cases} \quad (2)$$

Given a mode sequence of control-interrupt pairs $\sigma \in \Sigma^*$, we are interested in how many bits we need in order to specify σ . If no probability distribution over Σ is available, this number is given by the *description length*, as defined in [24]:

$$\mathcal{D}(\sigma, \Sigma) = |\sigma| \log_2(\text{card}(\Sigma)),$$

where $|\sigma|$ denotes the length of σ , i.e. the total number of modes in the string. This measure gives us the number of bits required for describing the sequence in the "worst"

case, i.e. when all the modes in Σ are equally likely. However, if we can establish a probability distribution p over Σ , the use of optimal codes can, in light of Equation (I), reduce the number of bits needed, which leads us to the following definition:

Definition: (Specification Complexity) Given a finite alphabet Σ and a probability distribution p over Σ . We say that a word $\sigma \in \Sigma^*$ has specification complexity

$$S(\sigma, \Sigma) = |\sigma| \mathcal{H}(\Sigma),$$

where \mathcal{H} is the entropy of the distribution.

Now, consider the problem of establishing a probability distribution over $\Sigma \subseteq U^Y \times \{0, 1\}^Y$ by recovering modes (and hence also empirical probability distributions) from empirical data. For example, supposing that the mode string $\sigma = \sigma_1 \sigma_2 \sigma_1 \sigma_3$ was obtained, then we can let $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, and the corresponding probabilities become $p(\sigma_1) = 1/2$, $p(\sigma_2) = 1/4$, $p(\sigma_3) = 1/4$. In such a case where we let Σ be built up entirely from the modes in the sequence σ , the empirical specification complexity depends solely on σ :

$$S^e(\sigma) = |\sigma| \mathcal{H}^e(\sigma) = - \sum_{i=1}^{M(\sigma)} \lambda_i(\sigma) \log_2 \frac{\lambda_i(\sigma)}{|\sigma|}, \quad (3)$$

where $M(\sigma)$ is the number of distinct modes in σ , $\lambda_i(\sigma)$ is the number of occurrences of mode σ_i in σ , and where we use superscript e to stress the fact that the probability distribution is obtained from empirical data.

Based on these initial considerations, the main problem, from which this work draws its motivation, is as follows:

Problem: (Minimum Specification Complexity) Given an input-output string

$$S = (y(1), u(1)), (y(2), u(2)), \dots, (y(n), u(n)) \in (Y \times U)^n,$$

find the minimum specification complexity mode string $\sigma \in \Sigma_{total}^*$ that is consistent with the data. In other words, find σ that solves

$$P(\Sigma_{total}, \mathbf{y}, \mathbf{u}) : \begin{cases} \min_{\sigma \in \Sigma_{total}^*} S^e(\sigma) \\ \text{subject to } \forall q \in \{1, \dots, n\} \\ \begin{cases} \sigma_{l(q)} = (k_{l(q)}, \xi_{l(q)}) \in \Sigma_{total} \\ k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q), \end{cases} \end{cases}$$

where the last two constraints ensure consistency of σ with the data S , and where $\mathbf{y} = (y(1), \dots, y(n))$, $\mathbf{u} = (u(1), \dots, u(n))$ give the empirical data string.

Note that this is slightly different than the formulation in Equation (2) since we now use $\sigma_{l(q)}$ to denote a particular member in $U^Y \times \{0, 1\}^Y$ instead of the $l(q)$ -th element in σ . Unfortunately, this problem turns out to be very hard to address directly. However, the easily established property

$$0 \leq \mathcal{H}^e(\sigma) \leq \log_2(M(\sigma)), \quad \forall \sigma \in \Sigma_{total}^*$$

allows us to focus our efforts on a more tractable problem. Here, the last inequality is reached when all the $M(\sigma)$ distinct modes of σ are equally likely.

As a consequence, we have $S^e(\sigma) \leq |\sigma| \log_2(M(\sigma))$ and thus it seems like a worth-while endeavor, if we want to find low-complexity mode sequences, to try to minimize either the length of the mode sequence $|\sigma|$ or the number of distinct modes $M(\sigma)$:

Problem: (Minimum Number of Modes) Given an input-output string

$$S = (y(1), u(1)), (y(2), u(2)), \dots, (y(n), u(n)) \in (Y \times U)^n,$$

find σ that solves

$$P_1(\Sigma_{total}, \mathbf{y}, \mathbf{u}) : \begin{cases} \min_{\sigma \in \Sigma_{total}^*} |\sigma| \\ \text{subject to } \forall q \in \{1, \dots, n\} \\ \begin{cases} \sigma_{l(q)} = (k_{l(q)}, \xi_{l(q)}) \in \Sigma_{total} \\ k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q). \end{cases} \end{cases}$$

Problem: (Minimum Distinct Modes) Given an input-output string

$$S = (y(1), u(1)), (y(2), u(2)), \dots, (y(n), u(n)) \in (Y \times U)^n,$$

find σ that solves

$$P_2(\Sigma_{total}, \mathbf{y}, \mathbf{u}) : \begin{cases} \min_{\sigma \in \Sigma_{total}^*} M(\sigma) \\ \text{subject to } \forall q \in \{1, \dots, n\} \\ \begin{cases} \sigma_{l(q)} = (k_{l(q)}, \xi_{l(q)}) \in \Sigma_{total} \\ k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q). \end{cases} \end{cases}$$

IV. WHAT ARE THE ANTS DOING?

Both the *Minimum Number of Modes Problem* and the *Minimum Distinct Modes Problem* have been solved [3], [10] and the first solution relies on dynamic programming, while the second solution relies on the initial construction of mode sequences where the interrupts always trigger. These sequences are then modified in such a way as to be maximally permissive in terms of their interrupts, while maintaining consistency with the data. Due to space limitations, we do not give these solutions explicitly, and the reader is referred to [3], [10] for the technical details. Instead we apply these methods to the problem of constructing mode sequences from the ant data, and in particular we consider an example where ten ants (*Aphaenogaster cockerelli*) are placed in a tank with a camera mounted on top, as seen in Figure 6. A 52 second movie is shot from which the Cartesian coordinates, x and y , and the orientation, θ , of every ant is calculated every 33ms using the previously discussed vision-based tracking software.

From this experimental data, an input-output string is constructed for each ant as follows: At each sample time τ , the input $u(\tau)$ is given by $(u_1(\tau), u_2(\tau))$ where $u_1(\tau)$ is the quantized angular velocity and $u_2(\tau)$ is the quantized translational velocity of the ant at time τ . Moreover, the output $y(\tau)$ is given by $(y_1(\tau), y_2(\tau), y_3(\tau))$ where $y_1(\tau)$ is the quantized angle to the closest obstacle, $y_2(\tau)$ is the quantized distance to the closest obstacle, and $y_3(\tau)$ is the quantized angle to the closest goal. Here, an *obstacle* is either a point on the tank wall or an *already visited* ant

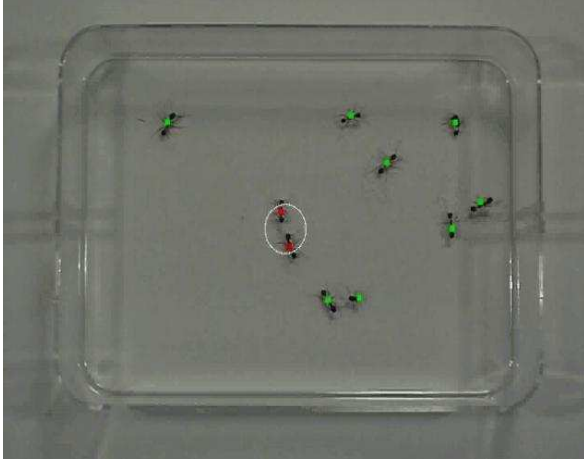


Fig. 6. Ten ants are moving around in a tank. The circle around two ants means that they are "docking", or exchanging information.

within the visual scope of the ant, and a *goal* is an ant that has not been visited recently. Figure 7 gives a good illustration of these notions of visual scope, goals and obstacles.

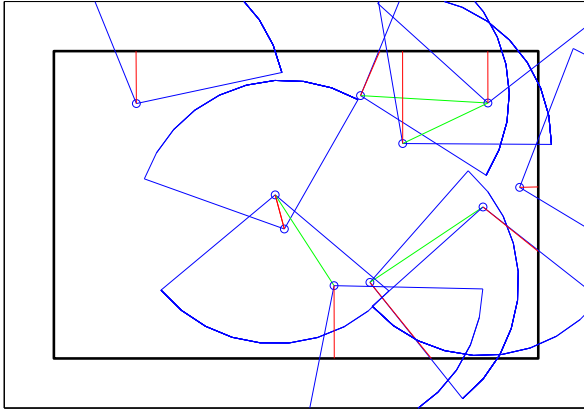


Fig. 7. This figure shows the conical visual scope as well as the closest obstacles (dotted) and goals (dashed) for each individual ant.

In this example, we choose to quantize $u_1(\tau), u_2(\tau), y_1(\tau), y_2(\tau)$ and $y_3(\tau)$ using 8 possible values for each. Thus $u(\tau)$ and $y(\tau)$ can respectively take 64 and 512 different values. For each ant, a mode sequence σ_1 with the shortest length as well as a minimum distinct mode sequence, σ_2 , have been computed from the input-output string of length $n = 106$. Results including string length, number of distinct modes, entropy and specification complexity of these two sequences for each of the ten ants are given in Table 1. In the table, results marked with a star are optimal. For σ_1 , it is the length $|\sigma|$ that is minimized and for σ_2 , it is the number of distinct modes $M(\sigma)$.

The minimum length sequence σ_1 has been constructed using a dynamic programming algorithm (see [3]), in which every element of the mode sequence is a new mode.

TABLE I

ant#	$ \sigma $		$M(\sigma)$		$\mathcal{H}^e(\sigma)$		$\mathcal{S}^e(\sigma)$	
	σ_1	σ_2	σ_1	σ_2	σ_1	σ_2	σ_1	σ_2
1	21*	57	21	5*	4.4	1.4*	92	82*
2	34*	66	34	5*	5.1	1.5*	172	99*
3	25*	68	25	6*	4.6	2.0*	116*	139
4	33*	64	33	6*	5.0	1.8*	166	116*
5	20*	65	20	6*	4.3	1.9*	86*	121
6	26*	73	26	6*	4.7	1.8*	122*	133
7	33*	71	33	6*	5.0	2.0*	166	145*
8	19*	74	19	7*	4.2	2.2*	80*	166
9	25*	71	25	10*	4.6	2.4*	116*	169
10	23*	60	23	4*	4.5	1.7*	104	102*

Consequently, $|\sigma_1| = M(\sigma_1)$. Moreover, the entropy of σ_1 is exactly equal to $\log_2(|\sigma_1|)$ as every mode is used only once in the sequence. The entropy of σ_2 is always smaller because the number of distinct modes is minimized and the modes are not equally recurrent in σ_2 .

Finally, the specification complexity is smaller with σ_1 for five of the ten ants, and smaller with σ_2 for the five others. On the average, there is a little advantage for σ_2 , with a total of 1152 bits compared to 1220 bits for σ_1 . An efficient way to ensure a low complexity coding would be to estimate both sequences for each ant and pick the one with lowest specification complexity. In our example, the total number of bits needed to encode the ten mode sequences using this coding strategy is 1064 bits.

It should be noted, however, that even though we have been able to recover mode strings, these strings can not be directly used as executable control programs without some modifications. Since the input-output string is generated from empirical data, measurement errors will undoubtedly be possible. Moreover, the dynamic system on which the control program is to be run (e.g. we have implemented mode strings obtained from the ant data on mobile robots) may not correspond exactly to the system that generated the data. Hence, a given input string might not result in the same output string on the original system and on the system on which the mode sequence is run.

For example, consider the case where we recovered the mode (k, ξ) and where the available empirical data only allows us to define the domain of k and ξ as a proper subset of the total output space Y , denoted here by Y_k or Y_ξ . (From the construction of the modes, these two subsets are always identical, i.e. $Y_k = Y_\xi$.) But, while executing this mode, it is conceivable that a measurement $y \notin Y_k$ is encountered, at which point some choices must be made. We here outline some possible ways in which this situation may be remedied:

- If $y_p \in Y_k$ and $\xi(y_p) = 0$, but the next measurement $y_{p+1} \notin Y_k$, we can replace $k(y_{p+1})$ with $k(y_p) \in Y_k$ as well as let $\xi(y_{p+1}) = 0$. As would be expected, this approach sometimes produces undesirable behaviors, such as robots moving around indefinitely in a circular motion.
- If $y_p \notin Y_k$, but $y_p \in Y_{\tilde{k}}$ for some other mode pair

$(\tilde{k}, \tilde{\xi})$ in the recovered mode sequence, we can let $k(y_p)$ be given by the most recurrent input symbol $\tilde{u} \in U$ such that $\tilde{k}(y_p) = \tilde{u}$. This method works as long as y_p belongs to the domain of at least one mode in the sequence. If this is not the case, additional choices must be made.

- If y_p does not belong to the domain of any of the modes in the sequence, we can introduce a norm on Y , and pick \tilde{y} instead of y_p , where \tilde{y} minimizes $\|y_p - \tilde{y}\|_Y$ subject to the constraint that \tilde{y} belongs to the domain for at least one mode in the sequence.

Note that all of these choices are heuristic in the sense that there is no fundamental reason for choosing one over the other. Rather they should be thought of as tools for going from recovered mode strings to executable control programs in an automatic fashion.

As for the question "What are the ants doing?", the proposed method provides the answer in terms of mode strings rather than qualitative descriptions of ant behaviors. In fact, the driving motivation behind the proposed research is to come up with executable programs, i.e. strings of control laws and interrupt conditions that can be operated on by robotic devices to produce "ant-like" behaviors rather than to produce insights into the lives of ants. However, a closer inspection of the recovered mode strings reveals that the most frequently occurring modes are qualitatively generating behaviors like "go straight slowly if no obstacles or goals are visible" or "go fast, turning left/right when an obstacle is to the right/left and/or the goal is to the left/right". These qualitative descriptions indicate that the hypothesis of labeling recently encountered ants as "obstacles" and other ants as "goals" is consistent with the ant behavior. Moreover, this fact is supported by the results reported in [5], where the head-to-head encounter rates were investigated as functions of the ant density.

V. CONCLUSIONS

In this paper, we presented a computer vision based software system that is able to track multiple social insects simultaneously, and our software was able to automatically infer aspects of their behavior. We demonstrated how the track information could be used to infer interactions between animals. We used the obtained ant data to generate low-complexity control programs consistent with the data. Such programs are built up from strings of control-interrupt pairs and can be thought of as strings of executable control code that can be applied in a number of robotic multi-agent scenarios.

REFERENCES

- [1] R.C. Arkin. *Behavior Based Robotics*. The MIT Press, Cambridge, MA, 1998.
- [2] K.J. Åström and B.M. Bernhardsson. Comparison of Riemann and Lebesgue Sampling for First Order Stochastic Systems. In *IEEE Conference on Decision and Control*, pp. 2011–2016, Las Vegas, NV, Dec. 2002.
- [3] A. Austin and M. Egerstedt. Mode Reconstruction for Source Coding and Multi-Modal Control. *Hybrid Systems: Computation and Control*, Springer-Verlag, Prague, The Czech Republic, Apr. 2003.
- [4] T. Balch and R.C. Arkin. Behavior-Based Formation Control for Multi-Robot Teams. *IEEE Transactions on Robotics and Automation*, Vol. 14, pp. 926–939, Dec. 1998.
- [5] T. Balch, H. Wilde, F. Dellaert, and Z. Khan. Automatic Detection and Analysis of Encounter Rates in the Desert Ant *Aphaenogaster Cockerelli* Using Computer Vision. Technical Report, *College of Computing*, Georgia Institute of Technology, Atlanta, GA, July 2004.
- [6] Y. Bar-Shalom, T.E. Fortmann, and M. Scheffe. Joint probabilistic data association for multiple targets in clutter. *Proc. Conf. on Information Sciences and Systems*, 1980.
- [7] R.W. Brockett. On the Computer Control of Movement. In the *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, pp. 534–540, New York, April 1988.
- [8] I. "Cousin and N. Franks. Self-organized lane formation and optimized traffic flow in army ants. *Proceedings of the Royal Society of London B*, 270:139–146, 2002.
- [9] I.J. Cox and J.J. Leonard. Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *AI*, Vol. 66, No. 2, pp. 311–344, April 1994.
- [10] F. Delmotte and M. Egerstedt. Reconstruction of Low-Complexity Control Programs from Data. *IEEE Conference on Decision and Control*, Atlantis, Bahamas, Dec. 2004.
- [11] R. Deriche and O.D. Faugeras. Tracking Line Segments. *IVC*, Vol. 8, pp. 261–270, 1990.
- [12] J. Desai, J. Ostrowski, and V. Kumar. Control of Formations for Multiple Robots. *IEEE Conference on Robotics and Automation*, Leuven, Belgium, May 1998.
- [13] M. Egerstedt and X. Hu. Formation Constrained Multi-Agent Control. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 6, pp. 947–951, 2001.
- [14] M. Egerstedt. Motion Description Languages for Multi-Modal Control in Robotics. In *Control Problems in Robotics, Springer Tracts in Advanced Robotics*, (A. Bicchi, H. Cristensen and D. Prattichizzo Eds.), Springer-Verlag, pp. 75–90, Las Vegas, NV, Dec. 2002.
- [15] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, Vol. 8, July 1983.
- [16] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, 2nd Ed., Addison-Wesley, New York, 2001.
- [17] D. Hristu and S. Andersson. Directed Graphs and Motion Description Languages for Robot Navigation and Control. *Proceedings of the IEEE Conference on Robotics and Automation*, May. 2002.
- [18] Z. Khan, T. Balch, and F. Dellaert. Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, 2003.
- [19] Z. Khan, T. Balch, and F. Dellaert. An MCMC-based Particle Filter for Tracking Multiple Interacting Targets. *European Conference on Computer Vision (ECCV 04)*, 2004.
- [20] V. Manikonda, P.S. Krishnaprasad, and J. Hendler. Languages, Behaviors, Hybrid Architectures and Motion Control. In *Mathematical Control Theory*, Eds. Willems and Baillieul, pp. 199–226, Springer-Verlag, 1998.
- [21] M. Mataric, M. Nilsson, and K. Simsarian. Cooperative Multi-Robot Box-Pushing. *Proc. IROS*, Pittsburgh, PA, 1995.
- [22] C. Rasmussen and G.D. Hager. Probabilistic Data Association Methods for Tracking Complex Visual Objects. *IEEE Trans. Pattern Anal. Machine Intell.*, Vo. 23, No. 6, pp. 560–576, June 2001.
- [23] J. Reif and H. Wang. Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots. *Robotics and Autonomous Systems*, Vol. 27, No. 3, 1999.
- [24] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, World Scientific Series in Computer Science, Vol. 15, River Edge, NJ, 1989.
- [25] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association. *Proceedings ICRA*, 2001.
- [26] H. Tanner, A. Jadbabaie, and G.J. Pappas. Coordination of Multiple Autonomous Vehicles. *IEEE Mediterranean Conference on Control and Automation*, Rhodes, Greece, June 2003.