Data-Driven Generation of Low-Complexity Control Programs[*]

Florent Delmotte, Magnus Egerstedt, and Adam Austin
{florent,magnus,austin}@ece.gatech.edu
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

In this paper we study the problem of generating control programs, i.e. strings of symbolic descriptions of control-interrupt pairs (or modes) from input-output data. In particular, we take the point of view that such control programs have an information theoretic content and thus that they can be more or less effectively coded. As a result, we focus our attention on the problem of producing low-complexity programs by recovering the shortest mode strings as well as the strings that contain the smallest number of distinct modes. An example is provided where the data is obtained by tracking ten roaming ants in a tank.

1 Introduction

As the complexity of many control systems increases, due both to the system complexity (e.g. manufacturing systems, [7]) and the complexity of the environment in which the system is embedded (e.g. autonomous robots [1, 19]), multi-modal control has emerged as a useful design tool. The main idea is to define different modes of operation, e.g. with respect to a particular task, operating point, or data source. These modes are then combined according to some discrete switching logic and one attempt to formalize this notion is through the concept of a *Motion Description Language* (MDL) [6, 11, 17, 20].

Each string in a MDL corresponds to a control program that can be operated on by the control system. Slightly different versions of MDLs have been proposed, but they all share the common feature that the individual atoms, concatenated together

[*]This work was supported by NSF through the programs EHS NSF-01-161 (grant # 0207411) and ECS NSF-CAREER award (grant # 0237971).

to form the control program, can be characterized by control-interrupt pairs. In other words, given a dynamical system

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

together with a control program $(k_1,\xi_1),\ldots,(k_z,\xi_z)$, where $k_i : Y \to U$ and $\xi_i : Y \to \{0,1\}$, 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[1].

Now, a number of results have been derived for such (and similar) systems, driven by strings of symbolic inputs, i.e. when the control and interrupt sets are finite. For example, in [5], the set of reachable states was characterized, while [12] investigates optimal control aspects of such systems. In [10, 17, 20], the connection between MDLs and robotics was investigated. However, in this paper we continue the development begun in [3, 8], where the control 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 [22] tells us that the minimal expected code length l satisfies

$$\mathcal{H}(\mathcal{A}) \leq l \leq \mathcal{H}(\mathcal{A}) + 1, \tag{2}$$

where $\mathcal{A}$ is the alphabet, and where the *entropy* is given by

$$\mathcal{H}(\mathcal{A}) = \sum_{a \in \mathcal{A}} p(a) \log_2 \frac{1}{p(a)}, \tag{3}$$

where $p(a)$ is the probability of drawing the symbol a from $\mathcal{A}$. The main problem that we will study is how to produce an empirical probability distribution over the set of modes, given a string of input-output data. Such a probability distribution would be useful when coding control procedures since a more common mode should be coded using fewer bits than an uncommon one. The ability to code control programs effectively moreover has a number of potential applications, from teleoperated robotics, control over communication constrained networks, to minimum attention control.

The outline of this paper is as follows: In Section 2 we introduce motion description languages, and in Section 3, we define the problem at hand and show how this can be addressed within the MDL framework. In Sections 4 and 5 we show how to find the shortest modes sequences as well as the mode sequences that contain the smallest number of distinct modes respectively. In Section 6, we give an example where the control programs are obtained from data generated by 10 roaming ants.

[1]Note that the interrupts can also be time-triggered, which can be incorporated by a simple augmentation of the state space.

2 Motion Description Languages

The primary objects of study in this paper are so called *motion description languages* (MDLs). Given a finite set, or alphabet, $\mathcal{A}$, by $\mathcal{A}^\star$ we understand the set of all strings of finite length over $\mathcal{A}$, with the binary operation of concatenation defined on $\mathcal{A}^\star$. Relative to this operation, $\mathcal{A}^\star$ is a semigroup, and if we include the empty string in $\mathcal{A}^\star$ it becomes a monoid, i.e. a semigroup with an identity, and a *formal language* is a subset of the free monoid over a finite alphabet. (See for example [14] for an introduction to this subject.) The concept of a *motion alphabet* has been proposed recently in the literature as a finite set of symbols representing different control actions that, when applied to a specific system, define segments of motion [6, 9, 13, 16, 20]. A MDL is thus given by a set of strings that represent such idealized motions, i.e. a MDL is a subset of the free monoid over a given motion alphabet. Particular choices of MDLs become meaningful only when the language is defined relative to the physical device that is to be controlled, as seen from Equation (1).

Now, in order to make matters somewhat concrete, we illustrate the use of MDLs with a navigation example, found in [10]. What makes the control of mobile robots particularly challenging is the fact that the robots operate in unknown, or partially unknown environments. Any attempt to model such a system must take this fact into account. We achieve this by letting the robot make certain observations about the environment, and we let the robot dynamics be given by

$$\begin{array}{l} \dot{x} = u, \;\; x, u \in \mathbb{R}^2 \\ y_1 = o_d(x), \;\; y_2 = c_f(x), \end{array}$$

where o_d is an odometric (possibly quantized) position estimate of x, and c_f is the (possibly quantized) contact force from the environment. The contact force could either be generated by tactile sensors in contact with the obstacle or by range sensors such as sonars, lasers, or IR-sensors.

Relative to this robot it is now possible to define a MDL for executing motions that drive the robot toward a given goal, located at x_F, when the robot is not in contact with an obstacle. On the other hand, when the robot is in contact with an obstacle, it seems reasonable to follow the contour of that obstacle in a clock-wise or counter clock-wise fashion, as suggested in [15]. We let the MDL be given by the set $\sigma_{GA} \cdot (\sigma_{OA} \cdot \sigma_{GA})^\star$, where GA and OA denote "goal-attraction" and "obstacle-avoidance" respectively, and where $a^\star = \{\emptyset, a, aa, aaa, ...\}$. The individual modes $\sigma_{GA} = (k_{GA}, \xi_{GA})$ and $\sigma_{OA} = (k_{OA}, \xi_{OA})$ are furthermore given by

$$\left\{ \begin{array}{l} k_{GA}(y_1, y_2) = \kappa(x_F - y_1) \\ \xi_{GA}(y_1, y_2) = \left\{ \begin{array}{l} 0 \text{ if } \langle y_2, x_F - y_1 \rangle \geq 0 \\ 1 \text{ otherwise} \end{array} \right. \end{array} \right.$$

$$\left\{ \begin{array}{l} k_{OA}(y_1, y_2) = cR(-\pi/2) y_2 \\ \xi_{OA}(y_1, y_2) = \left\{ \begin{array}{l} 0 \text{ if } \langle y_2, x_F - y_1 \rangle < 0 \text{ or } \angle(x_F - y_1, y_2) < 0 \\ 1 \text{ otherwise.} \end{array} \right. \end{array} \right.$$

The idea here is that the goal is located at x_F, and when the robot is not in contact with an obstacle, x_F is taken as a set-point in a proportional feedback law, provided

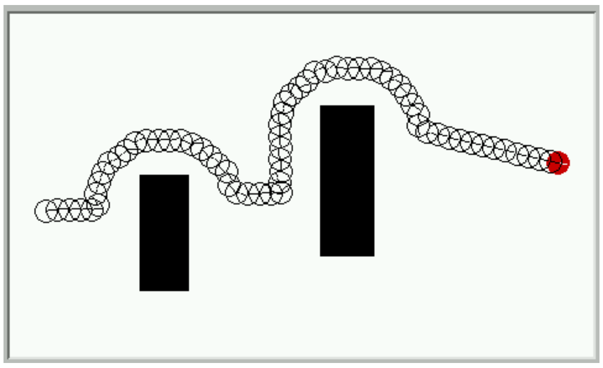

Figure 1: A multi-modal input string is used for negotiating two rectangular obstacles. Depicted is a simulation of a Nomadic Scout in the Nomadic `Nserver` environment.

by the mapping $k_{GA}(y_1, y_2) = \kappa(x_F - y_1)$, with $\kappa > 0$. When the robot is in contact with an obstacle, no set-point is needed, and $k(y_1, y_2)$ is simply given by $cR(-\pi/2)y_2$, where $c > 0$, $R(\theta)$ is a rotation matrix, and the choice of $\theta = -\pi/2$ corresponds to a clockwise negotiation of the obstacle. Note that in this example, the interrupts trigger as new obstacles are encountered, and in the definition of the interrupts, $\angle(\gamma, \delta)$ denotes the angle between the vectors γ and δ. An example of using this multi-modal control sequence is shown in Figure 1.

3 Specification Complexity

If we now 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 $\delta : \mathbb{R}^N \times U \to \mathbb{R}^N$ as $x_{p+1} = \delta(x_p, k(h(x_p)))$, given the control law $k : Y \to U$, with a new time update occurring whenever a new output or input value is encountered. For such a system, given the input string $(k_1, \xi_1), \ldots, (k_z, \xi_z) \in \Sigma^*$ where $\Sigma \subseteq \Sigma_{total}$, the evolution is given by

$$\begin{cases} x(q+1) = \delta(x(q), k_{l(q)}(y(q))), \quad y(q) = h(x(q)) \\ l(q+1) = l(q) + \xi_{l(q)}(y(q)). \end{cases} \tag{4}$$

Now, given a mode sequence of control-interrupt pairs $\sigma \in \Sigma^\star$, 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 [21]:

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

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 (2), 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*

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

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:

$$\mathcal{S}^e(\sigma) = |\sigma| \mathcal{H}^e(\sigma) = -\sum_{i=1}^{M(\sigma)} \lambda_i(\sigma) \log_2 \frac{\lambda_i(\sigma)}{|\sigma|}, \tag{5}$$

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*

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

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

$$\mathrm{P}(\Sigma_{total}, \mathbf{y}, \mathbf{u}) : \left\{ \begin{array}{l} \min_{\sigma \in \Sigma^*_{total}} \mathcal{S}^e(\sigma) \\ \text{subject to } \forall q \in \{1, \ldots, n\} \\ \left\{ \begin{array}{l} \sigma_{l(q)} = (k_{l(q)}, \xi_{l(q)}) \in \Sigma_{total} \\ k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q), \end{array} \right. \end{array} \right.$$

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

Note that this is slightly different than the formulation in Equation (4) 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

$$0 \le \mathcal{H}^e(\sigma) \le \log_2(M(\sigma)),\ \forall \sigma \in \Sigma^\star_{total}$$

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 $\mathcal{S}^e(\sigma) \leq |\sigma| \log_2(M(\sigma))$ and thus it seems like a worthwhile 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 $M(\sigma)$, which is the main pursuit in this paper:

Problem (Minimum Number of Modes):
Given an input-output string $S = (y(1), u(1)), (y(2), u(2)), \ldots, (y(n), u(n)) \in (Y \times U)^n$, find σ that solves

$$\mathrm{P}_1(\Sigma_{total}, \mathbf{y}, \mathbf{u}) : \begin{cases} \min_{\sigma \in \Sigma^*_{total}} |\sigma| \\ \text{subject to } \forall q \in \{1, \ldots, n\} \\ \begin{cases} \sigma_{l(q)} = (k_{l(q)}, \xi_{l(q)}) \in \Sigma_{total} \\ k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q). \end{cases} \end{cases}$$

Problem (Minimum Distinct Modes):
Given an input-output string $S = (y(1), u(1)), (y(2), u(2)), \ldots, (y(n), u(n)) \in (Y \times U)^n$, find σ that solves

$$\mathrm{P}_2(\Sigma_{total}, \mathbf{y}, \mathbf{u}) : \begin{cases} \min_{\sigma \in \Sigma^*_{total}} M(\sigma) \\ \text{subject to } \forall q \in \{1, \ldots, n\} \\ \begin{cases} \sigma_{l(q)} = (k_{l(q)}, \xi_{l(q)}) \in \Sigma_{total} \\ k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q). \end{cases} \end{cases}$$

4 Shortest Mode Sequences

In this section we are interested in solving Problem P_1, i.e. in generating the *shortest* mode string consistent with the data. We say that $\sigma \in sol(P_1)$ if σ solves P_1. The reason for the inclusion is that we can not hope for a unique solution. We furthermore let *length*(P_1) denote the unique string length of the solutions to P_1.

Lemma 1.1 *For any output-input string $(\mathbf{y}, \mathbf{u}) \in (Y \times U)^n$ it holds that $\emptyset \neq sol(\mathrm{P}_1)$ as well as length$(\mathrm{P}_1) \leq q$.*

Proof: We can always find a string of modes, containing q elements, that is consistent with the data by simply using modes that interrupt on every y-value, which will be exploited further in Section . In other words, $\sigma = \sigma_1 \cdots \sigma_n$ is consistent with the data if

$$\begin{cases} \sigma_i = (k_i, \xi_i) \\ k_i(y(i)) = u(i) \\ \xi_i(y) = 1, \ \forall y \in Y. \end{cases}$$

■

Now, we can reduce the complexity of the problem by restricting the interrupt functions to functions with a special structure, and we define $\hat{\Xi}$ as

$$\hat{\Xi} = \{\xi : Y \to \{0,1\} \mid \xi(y) = 1 \text{ for exactly one } y \in Y\} \subset \{0,1\}^Y,$$

i.e. $\hat{\Xi}$ is the set of interrupts that trigger for exactly one output value. By using this restricted set of interrupts, we can define the new problem $\hat{P}_1$ as the problem of solving P_1 while the interrupts are restricted to $\hat{\Xi}$.

Lemma 1.2 *length*$(\hat{\mathrm{P}}_1) =$ *length*(P_1).

Proof: We directly note that $\mathit{length}(P_1) \leq \mathit{length}(\hat{P}_1)$. Now, assume that $\sigma = \sigma_1 \cdots \sigma_M \in \mathit{sol}(P_1)$, where $\sigma_i = (k_i, \xi_i),\ i = 1, \ldots, M$. If we let $\mathit{trig}(i, \sigma, \mathbf{y}) \in Y$ denote the element in the output string that σ_i interrupts on, i.e. the $y \in \mathbf{y}$ that triggers a transition from σ_i to σ_{i+1} when $\xi_i(y) = 1$. We can then form $\hat{\sigma} = \hat{\sigma}_1 \cdots \hat{\sigma}_M \in (U^Y \times \hat{\Xi})^\star$, with

$$\begin{cases} \hat{\sigma}_i = (\hat{k}_i, \hat{\xi}_i),\ i = 1, \ldots, M \\ \hat{k}_i(y) = k_i(y),\ i = 1, \ldots, M \\ \hat{\xi}_i(y) = \begin{cases} 1 & \text{if } y = \mathit{trig}(i, \sigma, \mathbf{y}) \\ 0 & \text{otherwise,} \end{cases} \quad i = 1, \ldots, M. \end{cases}$$

It is clear that $\hat{\sigma}$ is consistent with the data, and hence $\hat{\sigma} \in \mathit{sol}(P_1)$. ■

Corollary 1.1 $\emptyset \neq$ *sol*$(\hat{\mathrm{P}}_1) \subset$ *sol*(P_1).

As a direct consequence of Corollary 1.1 we can focus our attention on the solutions to the reduced problem $\hat{P}_1$ directly. Without loss of generality we, in the next paragraphs, will use this observation as a basis for searching for strings of pairs $(k, \xi) \in U^Y \times \hat{\Xi}$.

Remark 1.1 *It should be noted that Lemma 1.2 would no longer hold if we were interested in finding the mode string that contained the least number of distinct modes, which will be the topic of Section , instead of the shortest strings, e.g.* $\sigma_1 \cdot \sigma_2 \cdot \sigma_1 \cdot \sigma_2$ *would be preferred over* $\sigma_1 \cdot \sigma_2 \cdot \sigma_3$. *This might arguably be a more natural way of selecting the modes, but in that case, it is potentially beneficial to use a mode that interrupts on multiple output values, and not only on one distinct y-value, which would contradict Lemma 1.2.*

By restricting the search to finding solutions in $\hat{P}_1$ we note that the behavior components are directly given by the output-input strings, i.e. that $k_{l_k}(y(k)) = u(k)$ is given by the output-input string $(y(1), u(1)), \ldots, (y(n), u(n))$. All that remains is thus to construct the correct interrupts. Since the interrupts are members of $\hat{\Xi}$, and the output-input string has length q, we can formulate the problem as a *dynamic programming* problem, where the *cost-to-go* satisfies *Bellman's equation*:

$$\mathcal{V}_k(\xi) = \min_{\xi' \in \hat{\Xi}} \{\mathcal{C}_k(\xi, \xi') + \mathcal{V}_{k-1}(\xi')\},$$

where $\mathcal{C}_k(\xi, \xi')$ is the transition cost associated with using ξ as interrupt at time k and letting ξ' be the interrupt at time $k-1$, $k = 2, \ldots, q$. It is clear that $\mathcal{C}_k(\xi, \xi') \in \{0, 1, \infty\}$ since a mode switch corresponds to increasing the number of modes by one, no mode switch corresponds to keeping the cost constant, and an infinite cost is incurred if the absence of a mode switch leads to inconsistencies with respect to the data.

The cost-to-go $\mathcal{V}_k(\xi)$ thus specifies the minimum number of modes that are needed for producing a mode sequence consistent with the data
$(y(1), u(1)), \ldots, (y(k), u(k))$ when the interrupt at time k is given by $\xi \in \hat{\Xi}$. In other words, we solve the mode recovery problem, i.e. solve P_1, by computing

$$\left\{ \begin{array}{l} \min_{\xi \in \hat{\Xi}} \mathcal{V}_q(\xi) \\ \mathcal{V}_1(\xi) = 1, \ \forall \xi \in \hat{\Xi}. \end{array} \right.$$

Now, in order to be able to define $\mathcal{C}_k(\xi, \xi')$ it is vitally important that we have a characterization of when the lack of mode switches produce inconsistencies with respect to the data. To this end, we will introduce a set $\mathcal{M}$ that contains the feedback mappings associated with the current mode. The idea now is to capture inconsistencies by comparing the mapping $k(y(k)) = u(k)$ to the feedback mappings present in the current mode, i.e. to the members of $\mathcal{M}$. To make this observation concrete, we first note that we can construct a bijective mapping $\Pi : \hat{\Xi} \to Y$ by letting

$$\xi(\Pi(\xi')) = \left\{ \begin{array}{ll} 0 & \text{if } \xi \neq \xi' \\ 1 & \text{if } \xi = \xi'. \end{array} \right.$$

If we, at time k, use interrupt ξ then we note that a mode switch occurs if and only if $\Pi(\xi) = y(k)$. In that case we should "reset" the description of the current mode. If we let $\mathcal{M}_k(\xi) \subset (Y \times U)^\star$, $k \in \{1, \ldots, q\}$, $\xi \in \hat{\Xi}$, we can update $\mathcal{M}$ as

$$\mathcal{M}_k(\xi) = \left\{ \begin{array}{ll} \{(y(k), u(k))\} & \text{if } \Pi(\xi) = y(k) \\ \{(y(k), u(k))\} \bigcup \mathcal{M}_{k-1}(\xi) & \text{otherwise.} \end{array} \right.$$

The reason why we use the same ξ as argument to $\mathcal{M}_k$ and $\mathcal{M}_{k-1}$ above is that when $\Pi(\xi) \neq y(k)$ an interrupt is not triggered and the same interrupt is used at time k and time $k-1$.

Now, what remains to be done is to define the transition costs and we first define a mapping η from $Y \times (Y \times U)^\star$ to $U \times \{\epsilon\}$, where ϵ is any symbol not in U. The idea is to use $\eta(y, \mathcal{M}_k(\xi))$ to produce the $u \in U$ that the current mode maps y to, i.e. to let

$$\eta(y, \mathcal{M}_k(\xi)) = \left\{ \begin{array}{ll} u & \text{if } (y, u) \in \mathcal{M}_k(\xi) \\ \epsilon & \text{otherwise.} \end{array} \right.$$

The transition cost $\mathcal{C}$ can thus be given by

$$\mathcal{C}_k(\xi,\xi') = \begin{cases} 1 & \text{if } \Pi(\xi) = y(k) \\ \infty & \text{if } \Pi(\xi) \neq y(k) \wedge \Big(\xi \neq \xi' \vee \eta(y(k), \mathcal{M}_{k-1}(\xi)) \notin \{\epsilon, u(k)\}\Big) \\ 0 & \text{otherwise.} \end{cases}$$

From Lemma 1.1 we know that P_1 has a finite solution, i.e. that the dynamic programming problem is guaranteed to produce the optimal solution [4] since, by construction, $\mathcal{C}$ and $\mathcal{M}$ are designed in such a way that by solving Bellman's equation we recover the shortest string of modes consistent with the data.

Proposition 1.1 *If*

$$\begin{array}{l} \xi(n) = \mathrm{argmin}_{\xi\in\hat{\Xi}}\{\mathcal{V}_n(\xi)\} \\ \xi(k-1) = \mathrm{argmin}_{\xi'\in\hat{\Xi}}\{\mathcal{C}_k(\xi(k),\xi') + \mathcal{V}_{k-1}(\xi')\},\ \ k = n,\dots,2, \end{array}$$

then $length(\mathrm{P}_1) = \mathcal{V}_n(\xi(n)) = card\Big(\{k \in \{2,\dots,n\} \mid \Pi(\xi(k)) = y(k)\}\Big) + 1.$

Furthermore, we can bound the computational effort involved in solving Bellman's equation in a straightforward manner:

Proposition 1.2 *Given the output-input string* $(y(1),u(1)),\dots,(y(n),u(n))$*. The number of operations needed for solving the Bellman equation is bounded above by* $\mathcal{O}(n\cdot card(Y)^3)$*.*

Proof: When solving the dynamic programming problem a total number of $card(\hat{\Xi}) = card(Y)$ possible interrupts must be investigated at each step $k = 1,\dots,n$, i.e. a total number of $n\cdot card(Y)$ nodes must be investigated in the dynamic programming graph. For each node the transition cost $\mathcal{C}_k(\xi,\xi')$ must be computed for all $\xi' \in \hat{\Xi}$, which is obtained by searching through $\mathcal{M}_{k-1}(\xi)$ for inconsistencies. But, $\mathcal{M}$ can at most contain $card(Y)$ consistent output-input pairs, i.e. the total number of computations needed for obtaining $\mathcal{C}_k(\xi,\xi')$ is bounded by $\mathcal{O}(card(Y))$, and the proof follows. ■

Example

As an illustrative example, consider the problem of recovering the mode string when the data $(y(1),u(1)),\dots,(y(6),u(6))$ is given by

y	0	0	1	2	0	1
u	2	0	0	2	1	0

We note that $Y = U = \{0,1,2\}$ and we let $\xi_i \in \hat{\Xi}$ denote the interrupt such that $\xi_i(i) = 1,\ i = 0,1,2$. We also switch the order of the data string in the dynamic programming algorithm for notational convenience. Hence $(y(1),u(1)) = (1,0),\ (y(2),u(2)) = (0,1)$, and so on.
Step 1:

$\mathcal{V}_1(\xi_i) = 1$ and $\mathcal{M}_1(\xi_i) = \{(y(1), u(1))\} = \{(1,0)\},\ \forall i \in \{0,1,2\}$.

Step 2:

Since $\Pi(\xi_0) = y(2) = 0$ we get that $\mathcal{C}_2(\xi_0, \xi) = 1,\ \forall \xi \in \hat{\Xi}$, and hence $\mathcal{M}_2(\xi_0) = \{(y(2), u(2))\} = \{(0,1)\}$. Furthermore, $\eta(y(2), \mathcal{M}_1(\xi)) = \epsilon, \forall \xi \in \hat{\Xi}$ since $y(2) = 0$ is not present in any output-input pairs in $\mathcal{M}_1$, and hence, for $i = 1, 2$, $\mathcal{C}_2(\xi_i, \xi) = 0,\ \forall \xi \in \hat{\Xi}$, which gives us that

$$\mathcal{V}_2(\xi_0) = 2,\ \mathcal{V}_2(\xi_1) = 1,\ \mathcal{V}_2(\xi_2) = 1,$$

as shown in Figure 2. Furthermore, for $i = 1, 2$, $\mathcal{M}_2(\xi_i) = \{(y(1), u(1)), (y(2), u(2))\}$ which is equal to $\{(1,0), (0,1)\}$.

This procedure can be repeated until Step 6, at which point $\min_{\xi \in \hat{\Xi}} \{\mathcal{V}_6(\xi)\} = 3$, as shown in Figure 2. The optimal mode string is thus given by the mode triple $(k_1, \xi_1), (k_2, \xi_2), (k_3, \xi_3)$ where the subscript denotes the order in the string, and where

$$\begin{array}{l}
\left\{\ k_1(0) = 2,\ \xi_1(0) = 1\right. \\
\left\{\begin{array}{l} k_2(0) = 0,\ k_2(1) = 0 \\ \ \xi_2(0) = 0,\ \xi_2(1) = 1 \end{array}\right. \\
\left\{\begin{array}{l} k_3(0) = 1,\ k_3(1) = 0,\ k_3(2) = 2 \\ \xi_3(0) = 0\ \xi_3(1) = 1,\ \xi_3(2) = 0. \end{array}\right.
\end{array}$$

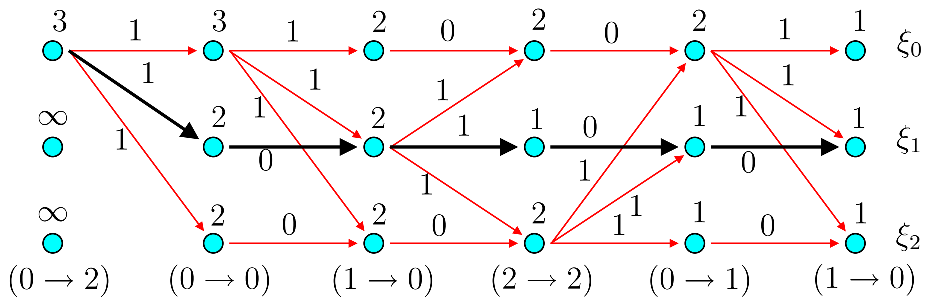

Figure 2: Depicted is the dynamic programming graph associated with the example. The number over each node gives the cost-to-go, and the arc-number is the transition cost. The solid arrows furthermore show one, of many, optimal solutions.

5 Always Interrupt Sequences

Definition (Always Interrupt Sequence): *We will refer to any mode string* $\sigma = \sigma_1 \cdots \sigma_n \in \Sigma^*_{total}$ *such that*

$$\begin{cases} M(\sigma) & \triangleq card\{\sigma_i \mid \sigma_i \in \sigma\} \\ & \triangleq card\{l(q) \mid q = 1, \ldots, n\} \qquad (6) \\ & = \max_{y \in Y}(card\{u \mid (y,u) \in S) \\ \xi_{l(q)}(y(q)) = 1, & q = 1, \ldots, n \qquad (7) \end{cases}$$

as an Always Interrupt Sequence (AIS)[2]
Here, Equation (6) means that the total number of distinct modes $M(\sigma)$ used in the AIS is equal to the maximum number of different input values u associated with an output value y in the sense that (u, y) appears as an input-output pair in the data string. One direct consequence of Equation (7) is that the length of an AIS is equal to the length n of the input-output string it is consistent with.

Existence: *Given an input-output string* $S \in (Y \times U)^n$ *there always exist an Always Interrupt Sequence consistent with the data.*

Proof: The consistency of a mode string with the data S is ensured by the two conditions:

$$\forall q \in \{1, \ldots, n\}, \quad \begin{cases} k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q) \end{cases}$$

Let M denote $\max_{y \in Y}(card\{u \mid (y,u) \in S\})$. For every $y \in Y$, there exist $m \leq M$ distinct values of u such that $(y, u) \in S$. One possible way to construct an AIS consistent with the data is to associate one distinct mode from the M available modes with each of the different values of u, i.e.

$$\forall (i,j) \text{ such that } y(i) = y(j),\ u(i) \neq u(j) \Rightarrow l(i) \neq l(j)$$

By doing so for every value of y encountered in S, we ensure that

$$\forall (i,j), \quad \begin{cases} l(i) = l(j) \\ y(i) = y(j) \end{cases} \Rightarrow u(i) = u(j)$$

so that the first condition is met. The second condition is always met since by definition of the AIS, $\xi_{l(q)}(y(q)) = 1,\ q = 1, \ldots, n$.
Hence we have constructed an AIS that is consistent with the data S. ■

One important fact should be noted here. In the proof, we proposed one particular AIS but there are many different ways to construct an AIS consistent with the

[2] Note that given a finite set C, by $card(C)$ we understand the number of different elements in C, e.g. $card(\{c_1, c_2, c_1, c_3\}) = 3$.

data.

Example. Given the following input-output string

y	0 0 1 2 2 0 1 1 0 1 2 2 1 2 1 0 2 0 2 1
u	4 2 1 2 3 0 3 3 1 1 4 4 0 2 3 4 4 0 1 0

We have $Y = \{0, 1, 2\}$ and:

$$\begin{aligned} card\{u \mid (0, u) \in S\} &= card\{4, 2, 0, 1, 4, 0\} = 4 \\ card\{u \mid (1, u) \in S\} &= card\{1, 3, 3, 1, 0, 3, 0\} = 3 \\ card\{u \mid (2, u) \in S\} &= card\{2, 3, 4, 4, 2, 4, 1\} = 4 \end{aligned}$$

so that an AIS will use $M = \max\{4, 3, 4\} = 4$ modes here.

As seen in the previous proof for existence, one way to build an AIS is to, for each $y \in Y$, establish an injective mapping between U and U^Y. For example, we can use:

mode	$y = 0$	$y = 1$	$y = 2$
1	$k_1(0) = 4$	$k_1(1) = 1$	$k_1(2) = 2$
2	$k_2(0) = 2$	$k_2(1) = 3$	$k_2(2) = 3$
3	$k_3(0) = 0$	$k_3(1) = 0$	$k_3(2) = 4$
4	$k_4(0) = 1$		$k_4(2) = 1$

Thus we get the following l-string:

y	0 0 1 2 2 0 1 1 0 1 2 2 1 2 1 0 2 0 2 1
u	4 2 1 2 3 0 3 3 1 1 4 4 0 2 3 4 4 0 1 0
l	1 2 1 1 2 3 2 2 4 1 3 3 3 1 2 1 3 3 4 3

and the corresponding mode sequence is
$\sigma = \sigma_1\sigma_2\sigma_1\sigma_1\sigma_2\sigma_3\sigma_2\sigma_2\sigma_4\sigma_1\sigma_3\sigma_3\sigma_3\sigma_1\sigma_2\sigma_1\sigma_3\sigma_3\sigma_4\sigma_3$.

Theorem: *Any mode string consistent with a given input-output string S is such that its number of modes is greater than or equal to*

$$M = \max_{y \in Y}(card\{u(q) \mid (y, u(q)) \in S,\ q \in \{1, \ldots, n\}\}).$$

Proof: Suppose that there exists a mode string σ consistent with the data using only $m < M$ modes. Consider the value of $y \in Y$ such that $card\{u(q) \mid (y, u(q)) \in S,\ q \in \{1, \ldots, n\}\} = M$ and label it y_M. In other words, there exist M different values of $u \in U$ such that $(y_M, u) \in S$. As $m < M$ there must exist two couples $(y_M, u(i))$ and $(y_M, u(j))$ in S with $u(i) \neq u(j)$ that are associated with the same mode, say $\sigma_x = \sigma_{l(i)} = \sigma_{l(j)}$. As the mode string is supposed to be consistent, we can write $k_x(y_M) = u(i)$ for the first couple and $k_x(y_M) = u(j)$ for the second one. But as

$u(i) \neq u(j)$ we have a contradiction.
Consequently, any mode string consistent with a given input-output string S must use at least M modes. ■

Corollary. *Any AIS consistent with the data is a solution to the problem* P_2.

Proof: To be consistent with the data, a mode string must use at least M modes. An AIS consistent with the data uses exactly M modes. Thus it solves P_2. ■

Theorem. *Given an input-output sequence $S \in (Y \times U)^n$ with a minimum number of distinct modes M, the number of possible AIS is bounded above by M^{n-M}.*

Proof: First, let us consider $y_M \in Y$ such that $card\{u(q) \mid (y_M, u(q)) \in S,\ q \in \{1, \ldots, n\}\} = M$ and $S_M = \{(y_M, u(q)) \in S,\ q \in \{1, \ldots, n\}\}$. To be consistent with the data, each distinct input-output pair in S_M must correspond to a different mode. There are $P_{S_M} = M!$ ways in which this can be achieved.
Now consider the other values of y. S_M contains at least M pairs $(y_M, u(q))$ so that we now have to look at the contribution of at most $n - M$ other pairs $(y(q), u(q))$ in S. Let S_m denote the corresponding set. Each element in S_m can potentially be associated with up to M modes so that S_m can add $P_{S_m} \leq M^{n-M}$ to the total number of possibilities in a multiplicative fashion.
The total number of modes can thus be bounded by: $P = \frac{1}{M!} P_{S_M} P_{S_m} \leq M^{n-M}$, where the division by $M!$ avoids counting sequences that differ from one another by permutations of the mode indexes. Note that this bound can be reached when S_M contains exactly M elements and S_m contains $n - M$ elements that all have a distinct value for y. ■

A conclusion to draw from this is that there is a large number of AIS and one question would be to pick the one that minimizes $\mathcal{S}^e(\sigma)$. However, as will be seen in the next paragraphs, there are potentially better ways of obtaining low complexity programs by abandoning the AIS structure by introducing a method that reduces the length of a given AIS. The idea is to modify the interrupt function ξ of each mode and make it be equal to zero (i.e. no mode change) whenever possible. Ideally, a sequence like $\sigma = \sigma_1\sigma_1\sigma_2\sigma_2\sigma_2\sigma_1\sigma_1\sigma_1\sigma_2\sigma_2$ could then be reduced to $\sigma' = \sigma_1\sigma_2\sigma_1\sigma_2$. This method, if plausible, would not add any new modes. The resulting sequence would still use exactly M distinct modes and would thus be another solution to P_2. But as the ξ functions are modified, the resulting mode sequence is no longer an AIS. In this matter, the resulting sequence will be referred to as a *Sometimes Interrupt Sequence* (SIS).

Algorithm (Sometimes Interrupt Sequence): *Given an AIS*
$\sigma = \sigma_{l(1)}, \sigma_{l(2)}, \ldots, \sigma_{l(n)}$ *consistent with an input-output sequence*
$S = (y(1), u(1)), \ldots, (y(n), u(n))$*, we construct the associated SIS by :*

1. *keeping* $\xi_x(y(q)) = 1$ *for all mode* σ_x *whenever* $\exists q \in \{1, \ldots, n\}$ *such that* $l(q) = x$ *and* $l(q+1) \neq x$,

2. *changing all the other values of* ξ *to zero, for all modes.*

Theorem. *The SIS derived from an AIS consistent with the data is consistent with the data.*

Proof: We recall here again the two conditions for consistency :

$$\forall q \in \{1,\dots,n\}, \quad \begin{cases} k_{l(q)}(y(q)) = u(q) \\ \xi_{l(q)}(y(q)) = 0 \Rightarrow l(q+1) = l(q). \end{cases}$$

The modifications of the AIS mode sequence only concern the ξ functions, i.e. the interrupts. Thus, we just have to prove that the modified sequence does not violate the second consistency condition.
Suppose we have a case where $\xi_{l(q)}(y(q)) = 0$ and $l(q+1) \neq l(q)$. This is impossible as it contradicts the first step in the construction of the SIS. Thus the second condition for consistency is always met and the SIS derived from an AIS which is consistent with the data is consistent with the data as well. ■

Example. Consider the following input-output string and the given AIS mode sequence σ (or equivalently the l string) which is consistent with this data.

y	1	0	2	0	2	2	1	2	2	0	1	2
u	0	1	1	0	0	0	1	1	0	1	1	1
l	$1 \to 2$		$2 \to 1$		1	$1 \to 2$		$2 \to 1 \to 2$			2	2

Now construct the associated SIS :

1. The arrows in the table show us whenever $l(q) \neq l(q+1)$, i.e. whenever we need to keep $\xi_{l(q)}(y(q)) = 1$. Here, we need to keep $\xi_1(1) = 1$, $\xi_1(2) = 1$ and $\xi_2(2) = 1$.

2. So we can set $\xi_1(0) = 0$, $\xi_2(0) = 0$ and $\xi_2(1) = 0$.

Consequently, the mode switches happening at $q = 2, 4, 7, 10$ and 11 have been suppressed and the above mode string has been reduced from
$\sigma = \sigma_1\sigma_2\sigma_2\sigma_1\sigma_1\sigma_1\sigma_2\sigma_2\sigma_1\sigma_2\sigma_2\sigma_2$ with length $N = 12$ to $\sigma = \sigma_1\sigma_2\sigma_1\sigma_1\sigma_2\sigma_1\sigma_2$ with length 7.

It can be easily shown that the action of removing one element from a mode string σ strictly reduces its specification complexity $\mathcal{S}^e(\sigma)$. The SIS is thus a mode sequence with lower complexity than the AIS it is derived from.

6 What Are the Ants Doing?

In this section we consider an example where ten ants *(Aphaenogaster cockerelli)* are placed in a tank with a camera mounted on top, as seen in Figure 3. 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 a vision-based tracking software. This experimental setup is provided by Tucker Balch and Frank Dellaert at the Georgia Institute of Technology Borg Lab[3] [18].

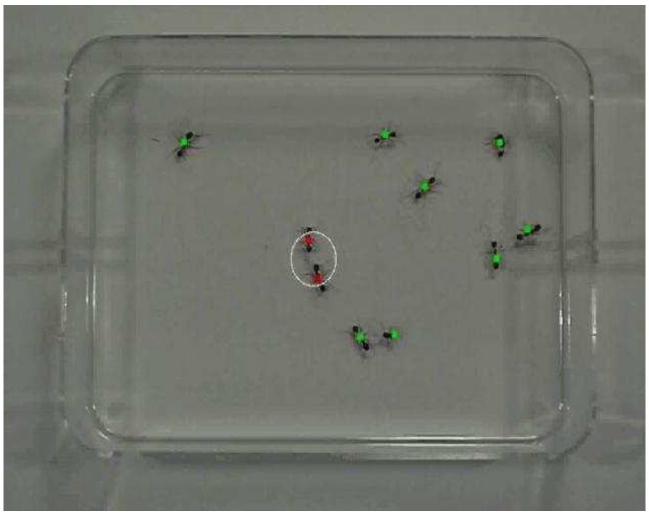

Figure 3: Ten ants are moving around in a tank. The circle around two ants means that they are "docking", or exchanging information.

From this experimental data, an input-output string is constructed for each ant as follows: At each sample time k, the input $u(k)$ is given by $(u_1(k), u_2(k))$ where $u_1(k)$ is the quantized angular velocity and $u_2(k)$ is the quantized translational velocity of the ant at time k. Moreover, the output $y(k)$ is given by $(y_1(k), y_2(k), y_3(k))$ where $y_1(k)$ is the quantized angle to the closest obstacle, $y_2(k)$ is the quantized distance to the closest obstacle, and $y_3(k)$ is the quantized angle to the closest goal. Here, an *obstacle* is either a point on the tank wall or an *already visited* ant within the visual scope of the ant, and a *goal* is an ant that has not been visited recently. Figure 4 gives a good illustration of these notions of visual scope, goals and obstacles.

In this example, we choose to quantize $u_1(k), u_2(k), y_1(k), y_2(k)$ and $y_3(k)$ using 8 possible values for each. Thus $u(k)$ and $y(k)$ can respectively take 64 and 512 different values. For each ant, a mode sequence σ_1 with the shortest length and the SIS associated with a particular choice of AIS 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 I.

In Table I, 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 $M(\sigma)$. It should however be noted that the length of σ_2 is in fact less than $n = 106$ as the mode

[3] http://borg.cc.gatech.edu

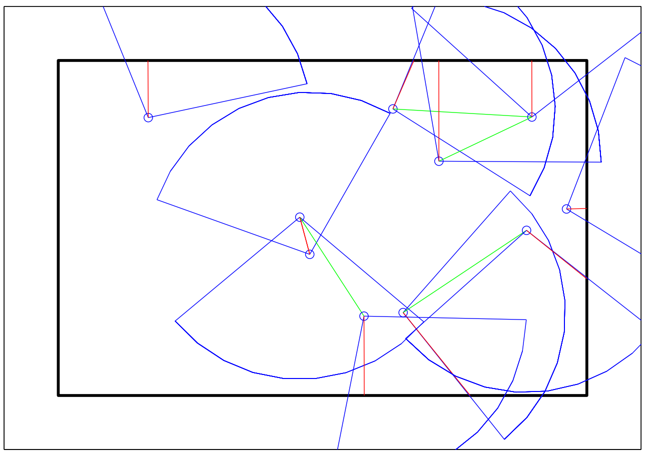

Figure 4: This figure shows the conical visual scope as well as the closest obstacles (dotted) and goals (dashed) for each individual ant.

sequence is not an AIS but a SIS.

The minimum length sequence σ_1 has been constructed using the dynamic programming algorithm in Section 4, in which every element of the mode sequence is a new mode. 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.[4] But, while executing

[4]From the construction of the modes, these two subsets are always identical, i.e. $Y_k = Y_\xi$.

Table 1:

ant#	$\lvert\sigma\rvert$		$M(\sigma)$		$\mathcal{H}^e(\sigma)$		$\mathcal{S}^e(\sigma)$	
	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

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 $k(y_p) \in U$ 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. However, more research is needed on this topic.

7 Conclusions

In this paper, we present a numerically tractable solution to the problem of recovering modes from empirical data. Given a string of input-output pairs, the shortest mode string as well as the string with the smallest number of distinct modes that is consistent with the data are characterized algorithmically. This has implications for how to generate multi-modal control laws by observing real systems, but also for the way the control programs should be coded. These algorithms can be thought of as providing a description of what modes are useful for solving a particular task, from which an empirical probability distribution over the set of modes can be obtained. This probability distribution can be put to work when coding the control programs, since a more common mode should be coded using fewer bits than an uncommon one. This work has thus a number of potential applications from teleoperated robotics, control over communication constrained networks, to minimum attention control.

Acknowledgments

The authors are grateful to Tucker Balch and Frank Dellaert for providing access to their bio-tracking data.

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] D.P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. 1.* Athena Scientific, Belmont, MA, 1995.

[5] A. Bicchi, A. Marigo, and B. Piccoli. Encoding Steering Control with Symbols. *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[6] 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.

[7] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, Norwell, MA, 1999.

[8] F. Delmotte and M. Egerstedt. Reconstruction of Low-Complexity Control Programs from Data. Submitted to *IEEE Conference on Decision and Control, Atlantis, Bahamas*, Dec. 2004.

[9] M. Egerstedt. Some Complexity Aspects of the Control of Mobile Robots. *American Control Conference*, Anchorage, Alaska, May, 2002.

[10] 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.

[11] M. Egerstedt and R.W. Brockett. Feedback Can Reduce the Specification Complexity of Motor Programs. *IEEE Transactions on Automatic Control*, Vol. 48, No. 2, pp. 213–223, Feb. 2003.

[12] E. Frazzoli. Explicit Solutions for Optimal Maneuver-Based Motion Planning. *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[13] T.A. Henzinger. Masaccio: A Formal Model for Embedded Components. *Proceedings of the First IFIP International Conference on Theoretical Computer Science*, Lecture Notes in Computer Science 1872, Springer-Verlag, 2000.

[14] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation, 2nd Ed.*, Addison-Wesley, New York, 2001.

[15] J.E. Hopcroft and G. Wilfong. Motion of Objects in Contact. *The International Journal of Robotics Research*, Vol. 4, No. 4, pp. 32–46, 1986.

[16] 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.

[17] D. Hristu-Varsakelis, M. Egerstedt, and P.S. Krishnaprasad. On The Structural Complexity of the Motion Description Language MDLe. *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[18] Z. Khan, T. Balch and F. Dellaert. *An MCMC-based Particle Filter for Tracking Multiple Interacting Targets*, Technical Report number GIT-GVU-03-35 October 2003

[19] D. Kortenkamp, R.P. Bonasso, and R. Murphy, Eds. *Artificial Intelligence and Mobile Robots.* The MIT Press, Cambridge, MA, 1998.

[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] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, World Scientific Series in Computer Science, Vol. 15, River Edge, NJ, 1989.

[22] C. E. Shannon. Mathematical Theory of Communication. *Bell Syst. Tech. J.*, Vol. 27, pp. 379-423, July and pp. 623-656, October, 1948.