Mode Reconstruction for Source Coding and Multi-Modal Control*

Adam Austin and Magnus Egerstedt

{austin,magnus}@ece.gatech.edu
Georgia Institute of Technology
School of Electrical and Computer Engineering
Atlanta, GA 30332, USA

Abstract. In this paper we take the point of view that control procedures have an information theoretic content that can be more or less effectively coded. Of particular interest are control procedures for navigation and obstacle avoidance for mobile robots, and we show how tokenized instructions can be used for understanding how computer generated inputs to robotics systems should be defined, selected, and coded. To this end, a dynamic programming algorithm is developed for generating control procedures that are useful in given robotics applications.

1 Introduction

In this paper we continue the development begun in [7–9] of viewing control procedures as having an information theoretic content, i.e. as being symbols that can be more or less effectively coded. Of particular relevance to this paper are control procedures for navigation and obstacle avoidance for mobile robots. When humans instruct each other how to carry out navigation tasks, only a limited number of tokenized instructions are used, which can be contrasted with classic control theory where a control action is specified at each time instant. We will show that such tokenized instructions are useful for understanding how computer generated inputs to robotics systems should be defined, selected, and coded. To this end, an information theoretic approach to control theory has been developed in [8], serving as a useful tool not only for source coding of control signals, but also for describing how symbolic instructions should be interpreted and operated on by continuous systems.

In this paper we model the way linguistic control signals affect mechanical devices using *trigger based hybrid systems* [5], serving as an abstraction between the symbolic computer programs and the continuous device dynamics. By combining this type of model with the notion of a *motion description language* (MDL) [4, 16], used for representing idealized motions, a model for linguistic control is arrived at in which a cost criterion for evaluating the control laws can be introduced, corresponding to the *description length* [17] of the control procedure.

* The support from NSF through the program EHS NSF-01-161 (grant # 0207411) is gratefully acknowledged.

This cost can be interpreted as the number of bits needed for uniquely coding a given control procedure.

However, the description length only provides the optimal code length if all control modes, or elements in the MDL, are equally likely. It would thus be desirable to generate a probability distribution over the set of modes, which would enable the use of optimal coding strategies. The construction of an algorithm that generates the shortest string of modes from strings of output-input pairs is thus the main contribution of this paper. This furthermore enables us to produce a probability distribution over the empirically obtained modes. This probability distribution can be put to work when coding the control procedures, since, by virtue of Shannon's source coding theorem [18], a more common mode should be coded using fewer bits than an uncommon one. This work has a number of potential applications from teleoperated robotics, control over communication constrained networks, to minimum attention control.

This paper is structured as follows: In Section 2 we introduce motion description languages defined both with respect to continuous machines and finite automata. In Section 3 we show how to pick control laws with short description lengths, followed by a discussion, in Section 4, about how to establish probability distributions over MDLs. A dynamic programming algorithm is proposed in Section 5 for solving this problem by generating control procedures from empirical data, which is applied in Section 6 to the problem of robot control.

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, A, by $A^\star$ we understand the set of all strings of finite length over A, with the binary operation of concatenation defined on $A^\star$. Relative to this operation, $A^\star$ is a semigroup, and if we include the empty string in $A^\star$ it becomes a monoid, i.e. a semigroup with an identity, and a *formal language* is a subset of thee free monoid over a finite alphabet. (See for example [13] 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 machine, define segments of motion [4, 8, 11, 14, 16]. A MDL is thus given by a set of strings that represent such idealized motions, i.e. a MDL is a subset of a 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. One such physical device is the so-called *quantized input-output machine*, given by $M = (U, X, Y, f, h)$, where U is a finite set of admissible inputs, Y is a finite set of outputs, $X \subset \Re^n$ is the state space of the system, $f : X \times U \to TX$ defines the system evolution, and $h : X \to Y$ is a measurable output function. The evolution of the machine is given by $\dot{x} = f(x, u),\ y = h(x)$, and from [9] we get the following definition:

Definition 1 (Motion Description Language). *Given a quantized input-output machine M. Relative to M we let a motion description language be given by a subset of the free monoid over the set $\Sigma = U \times U^{Y \times U} \times \{0,1\}^Y$.*

In other words, we let the letters in the motion alphabet be triples of the form (u, k, ξ), where $u \in U$, $k : Y \times U \to U$, and $\xi : Y \to \{0,1\}$. If, at time t_0, M receives the input string $(u_1, k_1, \xi_1), \ldots, (u_p, k_p, \xi_q)$, then x evolves according to

$$\begin{aligned} \dot{x} &= f(x, k_1(y, u_1)); \quad t_0 \leq t < T_1 \\ &\vdots \qquad\qquad\qquad \vdots \\ \dot{x} &= f(x, k_q(y, u_q)); \, T_{q-1} \leq t < T_q, \end{aligned}$$

where T_i denotes the time at which the interrupt ξ_i changes from 0 to 1.

Navigation Example

In order to make matters somewhat concrete, we illustrate these ideas 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

$$\begin{aligned} &\dot{x} = v, \;\; x, v \in \Re^2 \\ &y_1 = o_d(x), \;\; y_2 = c_f(x), \end{aligned}$$

where o_d is a quantized, odometric position estimate of x, and c_f is the 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 the goal 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 [12]. We let the MDL be given by the free monoid over the set

$$\big\{(u, k, \xi) \mid u = x_F, \;\; k(y_1, y_2, u) \in \{\kappa(u - y_1), cR(-\pi/2)y_2\}, \;\; \xi \in \{\xi_{GA}, \xi_{OA}\}\big\}.$$

The idea here is that the goal is located at x_F, and when the robot is not in contact with an obstacle, the open-loop part of the instruction basically provides x_F as a set-point. Furthermore, the closed-loop mapping $k(y_1, y_2, u) = \kappa(u - y_1)$ can thus be thought of as a simple, proportional feedback law. When the robot is in contact with an obstacle, no set-point is needed, and $k(y_1, y_2, u) = cR(-\pi/2)y_2$, where $c > 0$, $R(\theta)$ is a rotation matrix, and the choice of $\theta = -\pi/2$ corresponds to a clockwise negotiation of the obstacle.

In other words, the multi-modal control sequence used for negotiating obstacles is thus an element in the set $\sigma_{GA} \cdot (\sigma_{OA} \cdot \sigma_{GA})^\star$, where GA and OA denotes "goal-attraction" and "obstacle-avoidance" respectively, and where $a^\star = \{\emptyset, a, aa, aaa, ...\}$. The individual modes $\sigma_{GA} = (u_{GA}, k_{GA}, \xi_{GA})$ and $\sigma_{OA} = (u_{OA}, k_{OA}, \xi_{OA})$ are furthermore given by

$$\begin{cases} u_{GA} = x_F \\ k_{GA}(y_1, y_2, u) = \kappa(u - y_1) \\ \xi_{GA}(y_1, y_2) = \begin{cases} 0 \text{ if } \langle y_2, x_F - y_1 \rangle \geq 0 \\ 1 \text{ otherwise} \end{cases} \\ u_{OA} = x_F \\ k_{OA}(y_1, y_2, u) = cR(-\pi/2)y_2 \\ \xi_{OA}(y_1, y_2) = \begin{cases} 0 \text{ if } \langle y_2, x_F - y_1 \rangle < 0 \text{ or } \angle(x_F - y_1, y_2) < 0 \\ 1 \text{ otherwise.} \end{cases} \end{cases}$$

Here $\angle(\alpha, \beta)$ denotes the angle between the vectors α and β.

An example of using this multi-modal control sequence is shown in Figure 1.

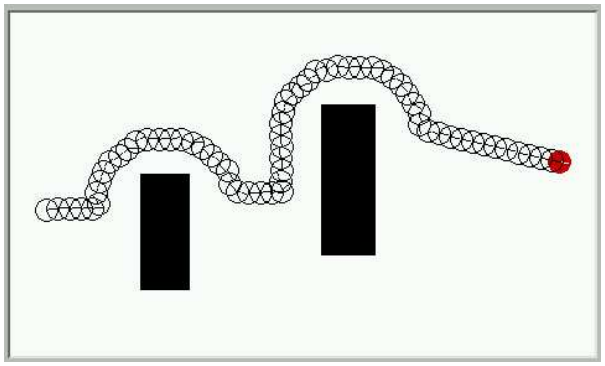

Fig. 1. A multi-modal input string is used for negotiating two rectangular obstacles. Depicted is a simulation of a Nomadic Scout in the Nomadic `Nserver` environment.

Now, in [8], a finite automata version of the quantized input-output machine was introduced, called a *Free-Running, Feedback Automaton* (FRF automaton) in order to arrive at an abstract model of, for example, a landmark-based navigation system for mobile robots [7, 14]. If we let X, U, Y be finite sets, and let $\delta \in X^{X \times U}$, $\gamma \in Y^X$, then we can identify $(X, Y, U, \delta, \gamma)$ with an *output automaton*, whose operation is given by $x_{k+1} = \delta(x_k, u_k)$ and $y_k = \gamma(x_k)$.

However, in order to let finite automata read strings of control modes, the model must be modified in such a way that instruction processing is akin to the way in which differential equations "process" piecewise constant inputs. The idea is to let the automaton read an input from a given alphabet, and then advance the state of the automaton repeatedly (free-running property) without reading any

new inputs until an interrupt is triggered. Additional structure is furthermore imposed on the input set to allow for feedback signals to be used. Hence a FRF-automaton is a free-running automaton whose input alphabet admits the structure $\Sigma = U \times K \times \Xi$, where, as before, U is a finite set, $K = U^{Y \times U}$, and $\Xi = \{0,1\}^Y$. Hence, the input to a FRF-automaton is a triple (u, k, ξ), where $u \in U,\ k : Y \times U \to U$, and $\xi : Y \to \{0,1\}$.

Definition 2 (Free-Running, Feedback Automaton [8]). *Let X, Y, U be finite sets and let $\delta : X \times U \to X,\ \gamma : X \to Y$ be given functions. Let $\Sigma = U \times K \times \Xi$, where U is a finite set, $K = U^{Y \times U}$, and $\Xi = \{0,1\}^Y$. We say that $(X, \Sigma, Y, \delta, \gamma)$ is a free-running, feedback automaton whose evolution equation is*

$$\begin{aligned} x_{k+1} &= \delta(x_k, k_{l_k}(y_k, u_{l_k})),\ y_k = \gamma(x_k) \\ l_{k+1} &= l_k + \xi_{l_k}(y_k), \end{aligned}$$

given the input string $(u_1, k_1, \xi_1) \cdots (u_p, k_p, \xi_p) \in \Sigma^\star$.

3 Source Coding of Control Signals

Now, assume that we are given a string of modes $\sigma \in \Sigma^\star$. This mode sequence can either be specified with respect to a FRF-automaton or a quantized input-output machine. The number of bits needed for describing σ uniquely is given by the following definition [6, 17]:

Definition 3 (Description Length). *Consider the finite set Σ. We say that a word $\sigma \in \Sigma^\star$ has description length*

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

where $|\sigma|$ denotes the length of σ, i.e. the number of modes in the string, and $card(\cdot)$ denotes cardinality.

The description length thus tells us how complicated σ is, i.e. how many bits we need for describing it, and since $\Sigma = U \times U^{Y \times U} \times \{0,1\}^Y$ we directly see that a higher resolution measurement results in a larger Y than what is the case for a lower resolution measurement. A better sensor might thus make the control procedures significantly more complicated while only providing marginally better performance. This trade-off between complexity and performance is something that can be capitalized on when designing control laws, which was done in [8] for FRF-automata, and in [7] for quantized input-output machines.

However, the description length does not tell the whole story. It may be a natural measure of the complexity of the control procedure, but if the string of modes is to be sent over a communication channel, which is the case for instance in teleoperated robotics, it would be desirable if a more effective representation of the mode string could be obtained. If we assume that we have been able to establish a probability distribution over Σ we can use optimal coding schemes, such as the Huffman code [15], for finding the shortest expected number of bits

$l^\star(\Sigma)$ needed for coding an element drawn at random from Σ. Shannon's classic source coding theorem [18] tells us that $\mathcal{H}(\Sigma) \leq l^\star(\Sigma) < \mathcal{H}(\Sigma) + 1$, where the *entropy* $\mathcal{H}(\Sigma)$ is given by

$$\mathcal{H}(\Sigma) = -\sum_{i=1}^{card(\Sigma)} p_i \log_2 p_i.$$

Here the interpretation is that the control triple $\sigma_i \in \Sigma$ occurs with probability p_i, and it should be noted, already at this point, that a probability distribution over Σ corresponds to a specification of what modes are potentially useful. But, to establish such a probability distribution over a structured set, such as the set of modes, is not a trivial task, and in the following sections we study the problem of recovering modes from empirical data. The idea is thus to observe how human operators instruct mobile robots, or how biological systems are structured, and reconstruct the modes from the experimental data.

4 Mode Recovery

As shown in the previous section, in order to employ optimal source coding strategies we first need to establish a probability distribution over Σ, i.e. over the set of modes. For this, a characterization of what modes are potentially useful is needed. In this section we show how such a characterization can be obtained by reconstructing mode-strings from empirical data. The main idea is to collect data, i.e. output-input strings, from real-world systems and then recover an appropriate multi-modal structure that is consistent with the data.

Regardless of whether the dynamic device model is a FRF-automaton or a kinematic machine, we assume that the given output-input string is a collection of pairs

$$(y(1), u(1)), (y(2), u(2)), \ldots, (y(q), u(q)) \in (Y \times U)^q \subset (Y \times U)^\star,$$

which we, with a slight abuse of notation, denote by $(\mathbf{y}, \mathbf{u})$, where $\mathbf{y} = y(1), y(2), \ldots, y(q)$ and $\mathbf{u} = u(1), u(2), \ldots, u(q)$. That a system evolving on a FRF-automaton would produce finite length strings of output-input pairs is clear, but some comments must be made in the continuous time case. In this case, one could sample the output-input data temporally, or generate a new pair whenever either the output or the input changes values, which corresponds to a so called *Lebesgue sampling*, in the sense of [2].

Now, given a collection of output-input pairs, the problem that we are interested in is how to produce mode strings that can generate this data. There are a number of ways in which one can conceivably achieve this, and in this paper we take the point of view that any numerically tractable algorithm that achieves the goal is useful. Once such mode strings have been obtained, the empirical probability distribution can simply be computed by letting p_i (the probability of using mode i) be equal to the number of times mode i was used over the total

number of modes used. This construction would furthermore be of importance as a multi-modal control synthesis tool, since empirical data would guide our selection of mode strings for accomplishing certain tasks.

In this paper we are interested in the problem of generating the *shortest* mode string $\sigma = \sigma_1 \cdots \sigma_M \in \Sigma^\star$ that is consistent with the data, i.e. to find σ that solves

$$\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u}) : \begin{cases} \min\limits_{\sigma} |\sigma| \\ \text{subject to} \\ \quad \begin{cases} \sigma_{l_k} = (u_{l_k}, k_{l_k}, \xi_{l_k}) \in \Sigma, \ k = 1, \ldots, q \\ k_{l_k}(y(k), u_{l_k}) = u(k), \ k = 1, \ldots, q \\ l_{k+1} = l_k + \xi_{l_k}(y(k)), \ k = 1, \ldots, q, \ l_1 = 1, \end{cases} \end{cases}$$

where the last two constraints ensure that σ is in fact consistent with the data.

We say that $\sigma \in sol(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u}))$ if σ solves $\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})$. The reason for the inclusion is that we can not hope for a unique solution. We furthermore let $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u}))$ denote the unique string length of the solutions to $\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})$.

Lemma 1. *For any output-input string* $(\mathbf{y}, \mathbf{u}) \in (Y \times U)^q$ *it holds that* $\emptyset \neq sol(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u}))$ *as well as* $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})) \leq q$.

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. In other words, $\sigma = \sigma_1 \cdots \sigma_q$ is consistent with the data if

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

□

If we now assume that our primary concern is to construct *feedback laws*, i.e. we limit our focus to mode sets of the form $\Sigma_{cl} = \{u_{cl}\} \times U^{\{u_{cl}\} \times Y} \times \{0,1\}^Y$, where u_{cl} is any arbitrary member of U, we can directly note the following fact:

Lemma 2. $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})) = length(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u}))$.

Proof: Assume that $\sigma = \sigma_1 \cdots \sigma_m \in sol(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u}))$, i.e. that $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})) = M$, where $\sigma_i = (u_i, k_i, \xi_i), \ i = 1, \ldots, M$. By Lemma 1 we know that such a σ exists.

Now, form $\sigma' \in \Sigma_{cl}^\star$ as $\sigma' = \sigma'_1 \cdots \sigma'_M$, where

$$\begin{cases} \sigma'_i = (u_{cl}, k'_i, \xi'_i), \ i = 1, \ldots, M \\ k'_i(u_{cl}, y) = k_i(u_i, y), \ i = 1, \ldots, M \\ \xi'_i(y) = \xi_i(y), \ i = 1, \ldots, M. \end{cases}$$

Now, since $\Sigma_{cl} \subset \Sigma$ it is clear that $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})) \leq length(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u}))$. But, by construction, σ and σ' are both consistent with the data, and hence $length(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u})) = M$. □

What Lemma 2 tells us is that we can search for the shortest closed-loop multi-modal instruction that is consistent with the data and by doing so solve the original problem directly since the lemma furthermore tells us that

$$sol(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u})) \subset sol(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})).$$

However, we can restrict the search further by only investigating interrupt 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 an even smaller set of modes as $\hat{\Sigma}_{cl} = \{u_{cl}\} \times U^{\{u_{cl}\} \times Y} \times \hat{\Xi} \subset \Sigma_{cl} \subset \Sigma$, and state the following lemma:

Lemma 3. $length(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u})) = length(\mathcal{P}(\hat{\Sigma}_{cl}, \mathbf{y}, \mathbf{u}))$.

Proof: We directly note that $length(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u})) \leq length(\mathcal{P}(\hat{\Sigma}_{cl}, \mathbf{y}, \mathbf{u}))$ since $\hat{\Sigma}_{cl} \subset \Sigma_{cl}$. Now, assume that $\sigma = \sigma_1 \cdots \sigma_M \in sol(\mathcal{P}(\Sigma_{cl}, \mathbf{y}, \mathbf{u}))$, where $\sigma_i = (u_{cl}, k_i, \xi_i),\ i = 1, \ldots, M$. If we let $trig(i, \sigma, \mathbf{y}) \in Y$ 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 $\hat{\sigma} = \hat{\sigma}_1 \cdots \hat{\sigma}_M \in \hat{\Sigma}_{cl}^\star$, with

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

It is clear that $\hat{\sigma}$ is consistent with the data, and hence $\hat{\sigma} \in sol(\mathcal{P}(\hat{\Sigma}_{cl}, \mathbf{y}, \mathbf{u}))$. □

Corollary 1. $\emptyset \neq sol(\mathcal{P}(\hat{\Sigma}_{cl}, \mathbf{y}, \mathbf{u})) \subset sol(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u}))$.

As a direct consequence of Corollary 1 we can focus our attention on the solutions to the reduced problem $\mathcal{P}(\hat{\Sigma}_{cl}, \mathbf{y}, \mathbf{u})$ directly. Without loss of generality we, in the next section, will use this observation as a basis for searching for strings of pairs $(k, \xi) \in U^Y \times \hat{\Xi}$, with the understanding that these pairs can be directly augmented to produce triples in $\hat{\Sigma}_{cl}$ in a straightforward manner.

Remark 1. It should be noted that Lemma 3 would no longer hold if we were interested in finding the mode string that contained the least number of distinct modes 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 3.

5 Dynamic Programming

By restricting the search to finding solutions in $\mathcal{P}(\hat{\Sigma}_{cl}, \mathbf{y}, \mathbf{u})$ we note that the closed loop 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 $(y(1), u(1)), \ldots, (y(q), u(q))$. 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 $\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})$, by computing

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

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

$$\xi(\Pi(\xi')) = \begin{cases} 0 & \text{if } \xi \neq \xi' \\ 1 & \text{if } \xi = \xi'. \end{cases}$$

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

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

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

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

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 we know that $\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})$ has a finite solution, i.e. that the dynamic programming problem is guaranteed to produce the optimal solution [3] 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. *If*

$$\begin{aligned} &\xi(q) = \operatorname{argmin}_{\xi \in \hat{\Xi}} \{\mathcal{V}_q(\xi)\} \\ &\xi(k-1) = \operatorname{argmin}_{\xi' \in \hat{\Xi}} \{\mathcal{C}_k(\xi(k), \xi') + \mathcal{V}_{k-1}(\xi')\}, \ k = q, \ldots, 2, \end{aligned}$$

then $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})) = \mathcal{V}_q(\xi(q)) = card\Big(\{k \in \{2, \ldots, q\} \mid \Pi(\xi(k)) = y(k)\}\Big) + 1$.

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

Proposition 2. *Given the output-input string* $(y(1), u(1)), \ldots, (y(q), u(q))$. *The number of operations needed for solving the Bellman equation is bounded above by* $\mathcal{O}(q card(Y)^3)$.

Proof: When solving the dynamic programming problem a total number of $card(\hat{\Xi}) = card(Y)$ possible interrupts must be investigated at each step $k = 1, \ldots, q$, i.e. a total number of $q card(Y)$ 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 $(y(1), u(1)), \ldots, (y(6), u(6))$ is given by

$$\begin{array}{c|cccccc} y & 0 & 0 & 1 & 2 & 0 & 1 \\ \hline u & 2 & 0 & 0 & 2 & 1 & 0 \end{array}$$

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

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

6 Autonomous Robots

The interest in constructing control procedures that could be coded using few bits was originally triggered by a desire to teleoperate mobile robots in communication constrained environments. We therefore report on our findings when collecting empirical data from an experiment where we let an autonomous robot navigate through a cluttered environment using a behavior-based control algorithm [1]. The key observation to be made is that for the dynamic programming algorithm to be applicable, the cardinality of Y and U as well as the length of the string of output-input pairs must be finite. In this example we therefore quantize the temporally sampled measurements and control actions made by the

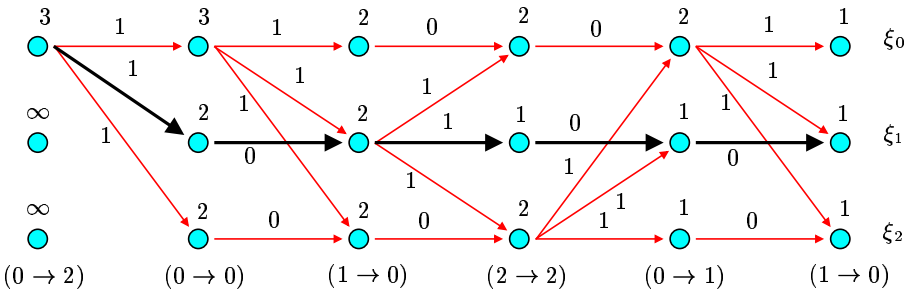

Fig. 2. Depicted is the dynamic programming graph associated with the example. The number over each node gives the cost-to-go, and the arc-number is the transition cost. The solid arrows furthermore show one, of many, optimal solutions.

robot. In particular, we let the robot be of a unicyle-type, i.e. its dynamics is given by

$$\begin{aligned}\dot{x}_1 &= v\cos\phi\\ \dot{x}_2 &= v\sin\phi\\ \dot{\phi} &= \omega,\end{aligned}$$

where (x_1, y_2) is the position and ϕ is the heading of the robot. The translational and angular velocities (v, ω) are the controlled variables, and we quantize them according to $u \in \{(v,\omega) \mid v \in V,\ \omega \in \Omega\}$, where

$$\begin{aligned}V &= \{\text{slow},\ \text{medium},\ \text{fast}\}\\ \Omega &= \{\text{fast left},\ \text{slow left},\ \text{straight},\ \text{slow right},\ \text{fast right}\}.\end{aligned}$$

In a similar manner the measurements made by the robot are sampled and quantized to produce an output string. We let $y \in \{(y_1, y_2, y_3) \mid y_1 \in Y_1,\ y_2 \in Y_2,\ y_3 \in Y_3\}$, where y_1 gives the distance to the closest obstacle, y_2 gives the relative angle to the closest obstacle, and y_3 gives the relative angle to the goal. By letting the angular quantization be given by the positions of the individual sonars on the sonar-ring, as shown in Figure 3(a), we get

$$\begin{aligned}Y_1 &= \{\text{close},\ \text{medium},\ \text{far}\}\\ Y_2 &= \{1, 2, \ldots, 8\}\\ Y_3 &= \{1, 2, \ldots, 8\}.\end{aligned}$$

In the experiment, we let the actual robot be controlled using

$$\begin{aligned}v &= v_0 \min\{1, (d_{ob}/D)^2\}\\ \omega &= C_{ob}(d_{ob})(\phi_{ob} + \pi - \phi) + C_g(\phi_g - \phi),\end{aligned}$$

where D is a prespecified safety distance, d_{ob}, ϕ_{ob} is the distance and direction to the closest obstacle, ϕ_g is direction to the goal, and $C_{ob}(d_{ob}) = 0$ if $d_{ob} \geq D$ and $(d_{ob} - D)/d_{ob}^3$ otherwise.

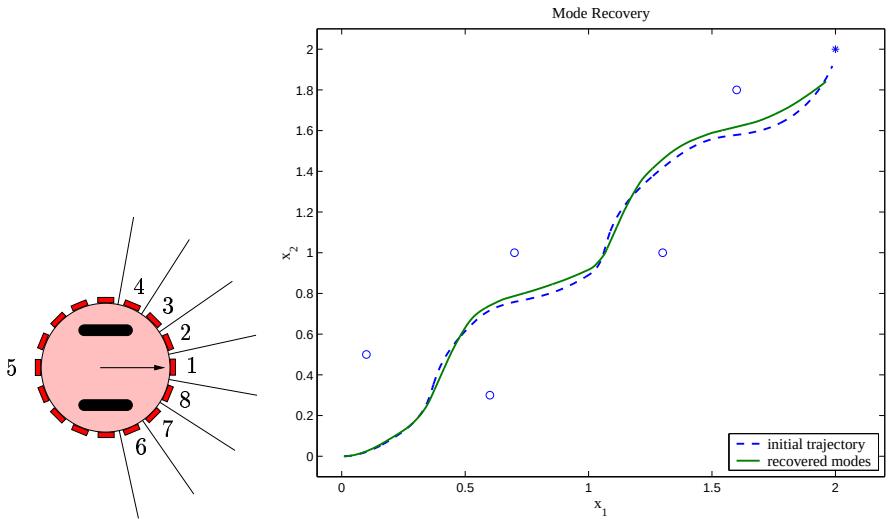

Fig. 3. In the left figure, the angular quantization is depicted, while the right figure shows the effect of controlling the robot using a recovered mode string.

The results from applying the dynamic programming algorithm to the data obtained from one string of output-input pairs is shown in Figure 3(b). There the real execution (dashed) is shown together with the effect of controlling the robot using the recovered mode string (solid). In the example, $card(Y) = 192$, $card(U) = 15$, $q = 233$, and the total number of modes is three, i.e. $length(\mathcal{P}(\Sigma, \mathbf{y}, \mathbf{u})) = 3$.

7 Conclusions

In this paper we present a numerically tractable solution to the problem of recovering modes from empirical data. Given a string of output-input pairs, the shortest string of mode triples that is consistent with the data is computed using dynamic programming. This has implications for how to generate multi-modal control laws by observing real systems, but also for the way the control procedures should be coded. The dynamic programming algorithm 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 procedures, 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.

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. D.P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. 1*. Athena Scientific, Belmont, MA, 1995.
4. 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.
5. R.W. Brockett. Hybrid Models for Motion Control Systems. In *Perspectives in Control*, Eds. H. Trentelman and J.C. Willems, pp. 29–54, Birkhäuser, Boston, 1993.
6. T.M. Cover and J.A. Thomas. *Elements of Information Theory*, John Wiley & Sons, Inc., New York, 1991.
7. M. Egerstedt. Some Complexity Aspects of the Control of Mobile Robots. *American Control Conference*, Anchorage, Alaska, May, 2002.
8. M. Egerstedt and R.W. Brockett. Feedback Can Reduce the Specification Complexity of Motor Programs. To appear in *IEEE Transactions on Automatic Control*, 2002.
9. M. Egerstedt. On the Specification Complexity of Linguistic Control Procedures. *International Journal of Hybrid Systems*, Vol. 2, No. 2, 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. 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.
12. 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.
13. J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation, 2nd Ed.*, Addison-Wesley, New York, 2001.
14. 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.
15. D.A. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proceedings of IRE*, Vol. 40, pp. 1098–1101, 1952.
16. 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.
17. J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, River Edge, NJ, 1989.
18. C.E. Shannon. A Mathematical Theory of Communication. *Bell Systems Technical Journal*, Vol. 27, pp. 379–423, 1948.