Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Motion Alphabet Augmentation Based on Past Experiences

Tejas R. Mehta, Florent Delmotte and Magnus Egerstedt

Abstract— **Multi-modal control is a commonly used design tool for breaking up complex control tasks into sequences of simpler tasks. It has previously been shown that rapidly-exploring randomized trees (as well as other viable approaches) can be used for reachability computations given a set of modes, and reinforcement learning can be performed over the reachable set to obtain the optimal control sequence. In this paper, we investigate the problem of adding new modes to a motion description language in a structured manner. We formalize an approach for augmenting the motion alphabet by adding new modes to reduce the complexity of the control program. In particular, we show a general technique for combining recurring mode sequences into one smooth "meta-mode". This problem is solved using a variational approach and numerical examples illustrate the feasibility of the proposed method.**

I. INTRODUCTION

In order to manage the rapidly growing complexity associated with many modern control applications, multi-modal control has emerged as a viable option. The main idea is to design a collection of modes, or control laws, defined with respect to a particular task, data source, or operating point, and then concatenate these control laws in order to produce the desired overall behavior. Given that such a mode string is the design objective, the control task thus involves mapping symbols (tokenized mode descriptions) to signals rather than signals (control values) to signals. A number of modelling paradigms facilitating this construction have been proposed from *Hybrid Automata* [1] to *Motion Description Languages* [2], [3].

The *expressiveness* of a given set of control modes (together with rules for their concatenation) can be characterized through the set of trajectories producible by the mode set, and hence the computation of the reachable set is one of the key topics in the hybrid systems literature. To name a few, [4], [5], [6], [7] proposed analytical methods for achieving this, while [8], [9] concerned the development of numerical algorithms for computing the reachable set. In this paper we follow the work in [10], [11], where rapidly-exploring randomized trees (RRTs) [12], [13] were put to work for reachability computations, and reinforcement learning was performed over the reachable set in order to obtain the optimal modal sequence. An example of using this approach to estimate the reachable set and learn the optimal control program is shown in Figure 1. Here we find the optimal path to drive a unicycle from x_0 to x_g while avoiding obstacles and minimizing the length of the control sequence along with the total distance travelled.

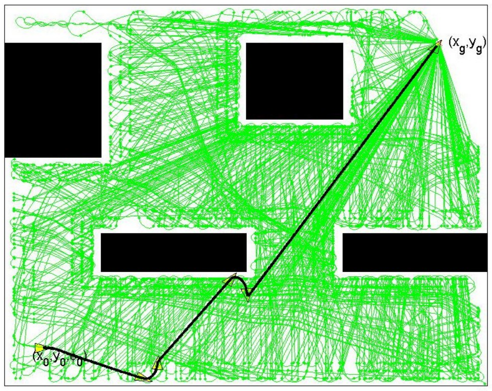

Fig. 1. Estimated reachable set along with the optimal learned path (thick) to drive a unicycle from x_0 to x_g given a set of modes.

In this paper we continue the development begun in [11], where it was shown how reinforcement learning over motion description languages transformed learning in continuous time over smooth manifolds to learning in discrete time over finite sets. The question under consideration in this paper is: "Given a set of modes and a corresponding characterization of the reachable set, what new modes, if any, should be added in order to improve the performance?" Moreover, this question should be answered without too much computational overhead.

The outline of this paper is as follows: In Section 2 we will formulate the problem and show how calculus of variations provides the tools needed for this solution. In Section 3 we will investigate a number of particular example problems, followed by a brief discussion about extensions, generalizations, and applications in Section 4.

II. MOTION ALPHABET AUGMENTATION

Formally, we define a mode σ as a pair (κ,ξ), where $\kappa : X \to U$ corresponds to a particular feedback law, and the interrupt $\xi : X \to \{0,1\}$ encodes conditions for its termination. Note here that X, U denote the state and input

This work was sponsored by the National Science Foundation through the program ECS NSF-CAREER award (grant # 0237971).

T. R. Mehta is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA tmehta@ece.gatech.edu

F. Delmotte is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA florent@ece.gatech.edu

M. Egerstedt is with the Faculty of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA magnus@ece.gatech.edu

space respectively. Given a finite set of feedback mappings K and interrupts Ξ, we let $\Sigma = K \times \Xi$ denote the set of all modes, or control-interrupt pairs. Moreover, by $\Sigma^\star$ we understand the set of all finite length strings over Σ, while Σ^N is the set of strings with length less than or equal to N. Now let $X_R^{\Sigma^\star} \subset X$ denote the reachable subset of X induced by $\Sigma^\star$. As mentioned earlier, the calculation of the reachable subset has been thoroughly studied and there is an abundance of literature pertaining to the estimation of $X_R^{\Sigma^\star}$. In particular, in [11] RRT's were employed to estimate the reachable set. Now, if we bound the length of the control program (i.e. $|\bar{\sigma}| \leq N$, where $\bar{\sigma} = \sigma_1\sigma_2\cdots\sigma_k$ ($k \leq N$) is the control sequence that transfers the system from x_0 to a state $x \in X_R^{\Sigma^N}$) and fix x_0, then the set $X_R^{\Sigma^N}$ is finite and we can use reinforcement learning over the reachable set to find the optimal mode string $\bar{\sigma}^* \in \Sigma^N$ to transition the system from x_0 to an open ball

$$\mathbf{B}(x_f, \varepsilon) = \{x \in X \quad | \quad \| x - x_f \| < \varepsilon\}.$$

Note above that $\bar{\sigma}^*$ is the optimal control program learned with respect to minimizing some predefined cost $J(x, \bar{\sigma})$.

Once we have obtained $\bar{\sigma}^*$ (i.e. using reinforcement learning), what new modes, if any, should be added in order to improve the performance? For this question to be well-posed, we first need to quantify performance: Say that we want to increase the expressiveness while reducing the *complexity* of the control program. The complexity of a control program (see [2], [3] for more details) $\bar{\sigma}$ can be represented by $|\bar{\sigma}|\log_2(card(\Sigma))$, hence there is a tradeoff between expressiveness and complexity (Note here that $card(\cdot)$ denotes the cardinality). Observe that the length of the control string is directly proportional to the complexity, while the cardinality of Σ is only logarithmically proportional to the complexity. Hence decreasing the string length at the cost of increasing the $card(\Sigma)$ may result in a reduction of the specification complexity.

So, in light of this discussion, we propose to add new modes in a highly structured manner by merging or combining recurring mode sequences into one smooth mode, if possible. Suppose for example there are multiple occurrences of $\sigma_1 = (\kappa_1, \xi_1)$ followed by $\sigma_2 = (\kappa_2, \xi_2)$ in our control program $\bar{\sigma}$ (i.e. $\bar{\sigma} = \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_4\sigma_1\sigma_2$), we would like to replace this mode pair $(\sigma_1\sigma_2)$ with one single mode $\sigma_{new} = (\kappa_{new}, \xi_2)$ that produces the combined behavior of $\sigma_1\sigma_2$, to some degree of accuracy. Hence, defining this new mode is equivalent to defining a new feedback law κ_{new}. In order to manage the complexity, we constrain κ_{new} to be a function of the existing feedback laws, i.e. $\kappa_{new} = \delta(\kappa_1, \kappa_2, \ldots, \kappa_{card(K)})$ for some $\delta : K^{card(K)} \to (X \to U)$. This problem can in fact be posed as a general optimization problem and solved using calculus of variations. This will be the topic of the next section assuming, without loss of generality, that we are merging two modes and that the interrupts are time-based (i.e. they trigger after a mode has been executing for certain time).

A. General Problem Formulation

Consider the following system:

$$\dot{x}(t) = \begin{cases} f_1(x(t)) & \text{if } t \in [0, \tau] \\ f_2(x(t)) & \text{if } t \in [\tau, T] \end{cases}, \tag{1}$$

where $x \in \mathbb{R}^n$ and $x(0) = x_0$ is given. Here $f_i : \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, 2$, are continuously differentiable functions. Now assume we have a set of continuously differentiable functions $g_i : \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, 2, \ldots, N$, and let

$$\dot{z}(t) = \sum_{i=1}^{N} \alpha_i g_i(z(t)), \text{where } \alpha \in \mathbb{R},\ z \in \mathbb{R}^n, \text{ and } z(0) = x_0. \tag{2}$$

We want to choose $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ so that the cost function

$$J(\vec{\alpha}) = \int_0^T L(x(t), z(t))dt + \psi(x(T), z(T)) \tag{3}$$

is minimized, where L and ψ are continuously differentiable in their second argument. Note that for our problem, $g_i = \delta_i(f_1, f_2)$, and that $L(x(t), z(t))$ may be 0 for all t if we are only concerned with the final position, but we will derive the solution to this more general problem formulation.

By adding the constraint with a co-state $\lambda(t)$ to (3), we obtain

$$\tilde{J}(\vec{\alpha}) = \int_0^T \Big[L(x(t), z(t)) + \\ + \lambda(t)\Big(\sum_{i=1}^{N} \alpha_i g_i(z(t)) - \dot{z}(t)\Big)\Big]dt + \psi(x(T), z(T)). \tag{4}$$

Note above that $\tilde{J}(\vec{\alpha})$ denotes the unperturbed cost. Now we perturb (4) in such a way that $\vec{\alpha} \to \vec{\alpha} + \varepsilon\vec{\theta}_k$, where $\vec{\theta}_k = [0, \ldots, \theta_k, \ldots, 0]^T$ (note the k^{th} entry is θ_k and all other entries are 0's), and $\varepsilon << 1$, then $z \to z + \varepsilon\eta$ is the resulting variation in $z(t)$. Note that above, we dropped the argument t when referring to $z(t)$ and will continue this convention in the following development for compactness with the implicit understanding that x, z and λ are functions of t. Now the perturbed cost is given by

$$\tilde{J}(\vec{\alpha} + \varepsilon\vec{\theta}_k) = \int_0^T \Big[L(x, z + \varepsilon\eta)) + \\ + \lambda\Big(\alpha_1 g_1(z + \varepsilon\eta) + \alpha_2 g_2(z + \varepsilon\eta) + \ldots + \\ + (\alpha_k + \varepsilon\theta_k)g_k(z + \varepsilon\eta) + \ldots + \alpha_N g_N(z + \varepsilon\eta) \\ - \dot{z}(t) - \varepsilon\dot{\eta}\Big)\Big]dt + \psi(x(T), (z + \varepsilon\eta)(T)). \tag{5}$$

Hence the Gateaux (also referred to as directional) derivative of $\tilde{J}$ in the direction of $\vec{\theta}_k$ is

$$\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \lim_{\varepsilon \to 0} \frac{\tilde{J}(\vec{\alpha} + \varepsilon\vec{\theta}_k) - \tilde{J}(\vec{\alpha})}{\varepsilon} \\ = \int_0^T \Big[\frac{\partial L}{\partial z}\eta + \sum_{i=1}^{N} \lambda\alpha_i \frac{\partial g_i}{\partial z}\eta + \lambda\theta_k g_k(z) \\ - \lambda\dot{\eta}\Big]dt + \frac{\partial\psi}{\partial z}\eta(T) \tag{6}$$

Now by integrating $\lambda\dot{\eta}$ in (6) by parts and rearranging terms, we obtain

$$\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \int_0^T \Big[\frac{\partial L}{\partial z} + \lambda \sum_{i=1}^{N} \alpha_i \frac{\partial g_i}{\partial z} + \dot{\lambda}\Big]\eta dt + \\ +\theta_k \int_0^T \lambda g_k(z)dt - \Big[\lambda\eta\Big]_0^T + \frac{\partial \psi}{\partial z}\eta(T) \qquad (7)$$

Note that $\eta(0) = 0$ since $z(0) = x(0) = x_0$. Now choose

$$\lambda(T) = \frac{\partial \psi}{\partial z} \qquad (8)$$

$$\dot{\lambda}(t) = -\frac{\partial L}{\partial z}(x,z) - \lambda(t)\sum_{i=1}^{N}\alpha_i \frac{\partial g_i}{\partial z}(z) \qquad (9)$$

With this choice of the co-state $\lambda(t)$, which can be solved by integrating (9) backwards with initial condition (8), we obtain

$$\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \Big[\int_0^T \lambda g_k(z(t))dt\Big]\theta_k \qquad (10)$$

Finally, note that (10) gives access to the partial derivative $\frac{\partial \tilde{J}}{\partial \alpha_k}$ since we know that

$$\nabla_{\vec{\theta}}\tilde{J}(\vec{\alpha}) = \frac{\partial \tilde{J}}{\partial \alpha_1}\theta_1 + \cdots + \frac{\partial \tilde{J}}{\partial \alpha_N}\theta_N, \qquad (11)$$

where $\vec{\theta} = [\theta_1, \ldots, \theta_N]^T$. Hence using (10) , (11), and the fact that α_k's are independent of each other, we deduce that

$$\frac{dJ}{d\alpha_k} = \int_0^T \lambda(t) g_k(z(t))dt. \qquad (12)$$

The motivation for obtaining an expression for the gradient (12) is that we can now employ a gradient descent method. The following numerical algorithm is proposed:

At each iteration n, where $\vec{\alpha}^{(n)}$ is the current vector of control variables, we follow these steps :

1) Compute the approximation function $z(t)$ forward in time from 0 to T using (2).

2) Compute the co-state $\lambda(t)$ backward in time from T to 0 using (8) and (9).

3) Compute the gradient $\nabla\tilde{J}(\vec{\alpha}^{(n)}) = \Big[\frac{\partial \tilde{J}}{\partial \alpha_1^{(n)}}, \ldots, \frac{\partial \tilde{J}}{\partial \alpha_N^{(n)}}\Big]^T$ using (12).

4) Update the control variables as follow :
$$\vec{\alpha}^{(n+1)} = \vec{\alpha}^{(n)} - \gamma^{(n)}\nabla\tilde{J}(\vec{\alpha}^{(n)})$$

Note that the choice of the stepsize $\gamma^{(n)}$ can be critical for the method to converge. An efficient method among others is the use of Armijo's algorithm presented in [14]. Because of the non-convex nature of the cost function J, this gradient descent algorithm will only converge to a local minimum. Hence the attainment of a "good" local minimum can be quite dependent on the choice of a "good" initial guess for the control variables. However, the method presented here still offers significant reductions in the cost function. The association of such a local method with heuristic strategies in order to find a global minimum is not investigated here.

III. EXAMPLES

A. Linear System

Consider the following autonomous linear system:

$$\dot{x}(t) = \begin{cases} A_1 x(t) & \text{if } t \in [0, \frac{T}{2}] \\ A_2 x(t) & \text{if } t \in [\frac{T}{2}, T] \end{cases} . \qquad (13)$$

Again $x \in \mathbb{R}^n$ and $x(0) = x_0$ is given, then the evolution of x is given as follows:

$$x(0) = x_0$$
$$x(\tfrac{T}{2}) = e^{A_1 \frac{T}{2}} x_0$$
$$x(T) = e^{A_2 \frac{T}{2}} x(\tfrac{T}{2}) = e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0$$

Now suppose we want to find a new A such that

$$x(T) = e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0 \approx e^{AT} x_0. \qquad (14)$$

Hence we need to find $\vec{\alpha}$ that ensures (14) holds when $A = \sum_{i=1}^{N} \alpha_i G_i$, where we let G_i belong to the *P. Hall basis* $L(A_1, A_2)$, where L is the Lie algebra. This approach of using Lie brackets from the P. Hall basis to obtain a control for solving the Motion Planning Problem was explored in [15]. For the definition of P. Hall basis, see [16] and [17], here we just present an example.

Example: The P. Hall basis of $L(X,Y)$ up to degree four is X, Y, $[X,Y]$, $[X,[X,Y]]$, $[Y,[X,Y]]$, $[X,[X,[X,Y]]]$, $[Y,[X,[X,Y]]]$, $[Y,[Y,[X,Y]]]$, $[X,[X,[X,[X,Y]]]]$, $[Y,[X,[X,[X,Y]]]]$, $[Y,[Y,[X,[X,Y]]]]$, and $[Y,[Y,[Y,[X,Y]]]]$.

One way of obtaining A is through the use of the well known Campbell-Baker-Hausdorff (CBH) formula, which can be stated as follows:

Campbell-Baker-Hausdorff (CBH) Formula For any two matrices X, Y sufficiently close to 0 , there exists a matrix $Z \in L(X,Y)$ such that $e^Z = e^X e^Y$. Moreover, Z can be explicitly expressed in the Dynkin form as: $Z = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}[X,[X,Y]] + \frac{1}{12}[Y,[Y,X]] + \ldots$, where $[X,Y] = XY - YX$ is the matrix commutator.

Since CBH formula gives an infinite series, we have to be concerned about the convergence when applying the formula. The convergence of the CBF formula has been well studied [18], [19], and it is shown that the Dynkin series converges for matrices X, Y if there is a Lie norm for which

$$\| X \|_{Lie} + \| Y \|_{Lie} \leq \log(2). \qquad (15)$$

Here $\| \cdot \|_{Lie}$ denotes the Lie norm, which is a norm on matrices compatible with Lie multiplication, i.e.

$$\| [X,Y] \|_{Lie} \leq \| X \|_{Lie} \| Y \|_{Lie} .$$

Clearly if T is sufficiently small, then $\| A_i \frac{T}{2} \|_{Lie}$ will meet the bound above (15) for $i = 1, 2$. In this case we should be able to approximate this result by using finite number elements from the Lie algebra.

Using CBH formula it is clear that,

$$\begin{aligned} A &= \tfrac{1}{2}A_1 + \tfrac{1}{2}A_2 + \tfrac{T}{4}[A_1, A_2] + \tfrac{T^2}{8}[A_1, [A_1, A_2]] + \dots \\ &\equiv \tfrac{1}{2}A_1 + \tfrac{1}{2}A_2 + \tfrac{T}{4}[A_1, A_2] + \Delta(T^2), \end{aligned} \quad (16)$$

where $\Delta(T^2)$ is the remaining part of the series which is polynomial in T of degree greater than T^2. Now let us denote $\tilde{A} = \frac{1}{2}A_1 + \frac{1}{2}A_2 + \frac{T}{4}[A_1, A_2]$, we will show that $\| e^{\tilde{A}+\Delta(T^2)} - e^{\tilde{A}} \|$ is bounded by $o(T^2)$. Hence $x(T) \approx e^{\tilde{A}T}$ for a small enough T. First, note the following expression derived in [20],

$$e^{A+\Delta} - e^{A} = \int_0^1 e^{(1-\tau)A} \Delta e^{\tau A} d\tau + o(\| \Delta \|). \quad (17)$$

Hence, by manipulating (17) we obtain

$$\begin{aligned} \| e^{A+\Delta} - e^{A} \| &\leq \| \int_0^1 e^{(1-\tau)A} \Delta e^{\tau A} d\tau \| + o(\| \Delta \|) \\ &\leq \int_0^1 \| e^{(1-\tau)A} \Delta e^{\tau A} \| d\tau + o(\| \Delta \|) \\ &\leq \int_0^1 e^{\|A\|} \| \Delta \| e^{\|A\|} d\tau + o(\| \Delta \|) \\ &= e^{2\|A\|} \| \Delta \| + o(\| \Delta \|) \end{aligned} \quad (18)$$

So in our case (18) is reduced to

$$\| e^{\tilde{A}+\Delta(T^2)} - e^{\tilde{A}} \| \leq e^{2\|\tilde{A}\|} o(T^2). \quad (19)$$

We have thus shown that A given by the CBH formula can be approximated by $\tilde{A}$ for a small enough T. Of course we can approximate A by using higher-order Lie brackets to obtain a better approximation if desired.

Alternatively, we can also use the calculus of variations solution derived earlier to approximate A. Consequently, let $g_1(x) = A_1 x$, $g_2(x) = A_2 x$, $g_3(x) = [A_1, A_2]x$, $g_4(x) = [A_1, [A_1, A_2]]x$, and so on. Now we can calculate the coefficients $\vec{\alpha}^* = [\alpha_1, \alpha_2, \dots]^T$ as outlined earlier. So let us take a specific example and compare the two methods described here. Let

$$\dot{x}(t) = \begin{cases} \begin{pmatrix} 1 & 0.3 \\ 0 & -1 \end{pmatrix} x(t) & \text{if } t \in [0, \frac{T}{2}] \\ \begin{pmatrix} -1.2 & 0.1 \\ -0.3 & 1 \end{pmatrix} x(t) & \text{if } t \in [\frac{T}{2}, T] \end{cases}. \quad (20)$$

Suppose $x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and let's derive a new matrix A_{new} that tranfers the system from x_0 to $x(T) \approx e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0$ in time T without the switch at time $\frac{T}{2}$. Figure 2 shows the trajectories obtained using the first-order approximation of the CBH formula and the corresponding calculus of variation approximation using $\hat{A}_{new} = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_2 [A_1, A_2]$ with $T = 2$. Note that the calculus of variations approach obtained a virtually perfect match, while the CBH approximation was not very accurate. In this case $\vec{\alpha}^* = (0.8763, 0.8112, 0.4134)^T$, hence $\hat{A}_{new} = 0.8763A_1 + 0.8112A_2 + 0.4134[A_1, A_2]$ as opposed to $\tilde{A}_{new} = 0.5A_1 + 0.5A_2 + 0.5[A_1, A_2]$, given by the CBF formula. The CBF formula expectedly provides a better approximation when $T = 1$ (i.e. for a smaller T) as shown in Figure 3. The evolution of $\vec{\alpha}$ in the steepest descent algorithm for both cases ($T = 2, 1$) is shown in Figure 4. Here $\gamma^{(n)} = 0.05$ is the step size at iteration n, and it should be noted that the algorithm converges quickly. Observe that the calculus of variations result depends on the initial condition x_0 and hence $\vec{\alpha}^*$ will vary as x_0 varies, however the CBH formula provides global results that are independent of the initial condition x_0.

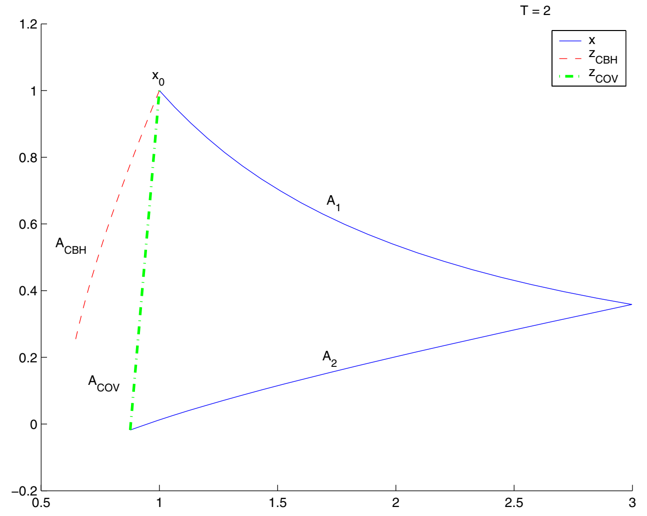

Fig. 2. Comparison of the two methods described with $T = 2$. In this case $\| x(T) - z_{CBH}(T) \| = 0.3459$, while $\| x(T) - z_{COV}(T) \| = 1.81 \cdot 10^{-5}$.

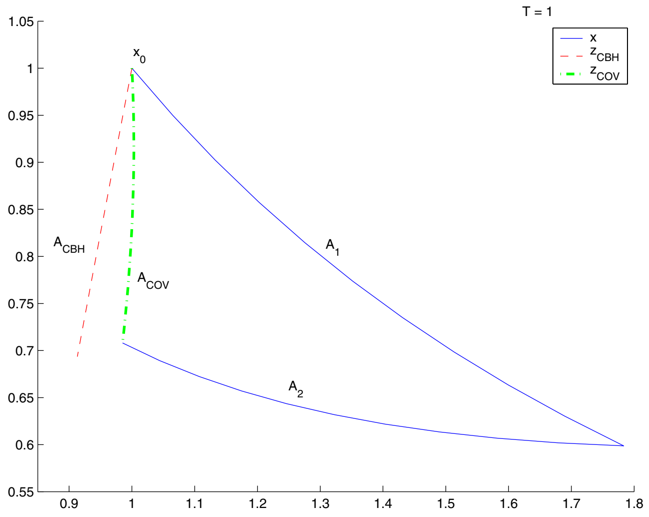

Fig. 3. Comparison of the two methods described with $T = 1$. In this case $\| x(T) - z_{CBH}(T) \| = 0.0736$, while $\| x(T) - z_{COV}(T) \| = 5.84 \cdot 10^{-4}$.

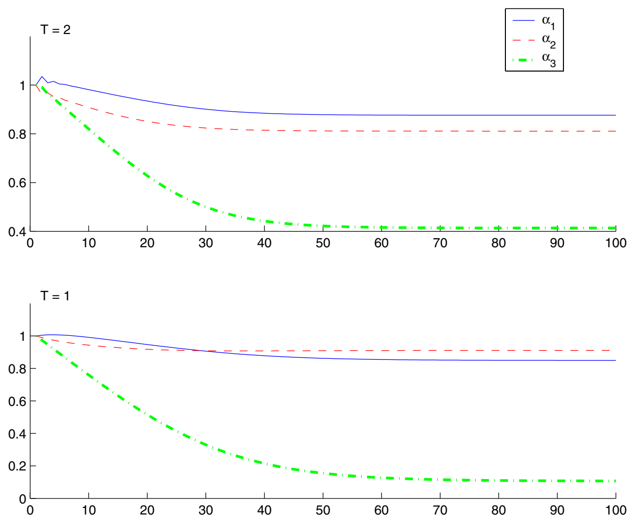

Fig. 4. The evolution of $\vec{\alpha}$ for T = 2 and T = 1, note here that the step size $\gamma^{(n)} = 0.05$.

B. Robotics Example

Here we consider the problem of steering a unicycle. The equations for this system are

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega. \end{aligned} \tag{21}$$

In the system above (x,y) are the Cartesian coordinate of the center of the unicycle and ϕ is its orientation with respect to the x-axis. Assume that v is constant and ω is the control variable. Given that the system initially has two behaviors, namely "go-to-goal" and "avoid-obstacle", the feedback mappings associated with each behavior are as follow:

$$\begin{aligned} \kappa_g(x,y,\phi) &= \omega_g = C_g(\phi_g - \phi), \\ \kappa_o(x,y,\phi) &= \omega_o = C_o(\pi + \phi_o - \phi). \end{aligned}$$

Note here that C_g and C_o are the gains associated with each behavior, and ϕ_g and ϕ_o are the angles to the goal and nearest obstacle respectively. Both of these angles are measured with respect to the x-axis and can be expressed as

$$\begin{aligned} \phi_g &= \arctan(\frac{y_g - y}{x_g - x}) \text{ and} \\ \phi_o &= \arctan(\frac{y_{obs} - y}{x_{obs} - x}), \end{aligned}$$

where (x_g, y_g) and (x_{obs}, y_{obs}) are the Cartesian coordinates of the goal and the nearest obstacle respectively. We also have a set of three interrupts, $\xi_{1,2,3}(x)$, that trigger at three different distances away from the nearest obstacle (x_{obs}, y_{obs}), and all three interrupts always trigger at the goal (x_g, y_g). Hence the total number of available modes is six, i.e. $card(\Sigma) = 6$. The problem then is to plan a path from an initial state (x_0, y_0, ϕ_0) to an open ball around (x_g, y_g) given the set of modes above while minimizing the string length of the control program (i.e. number of switches) along with the total distance travelled.

Given this set of modes we begin by exploring the reachable space using RRTs. The general algorithm for accomplishing this is given by

```
$\chi := \{x_0\}$
for $k = 1$ to $N$
    $x := rand(\chi)$
    $\sigma := rand(\Sigma)$
    $x' := f(x, \sigma)$
    if $x' \notin \chi$ then
        $\chi := \chi \cup \{x'\}$
    end if
end for
$X_R^{\Sigma^\star} \approx \chi$
```

The algorithm starts by exploring from x_0, i.e. set $\chi = \{x_0\}$. At each iteration, we select a state x randomly from χ (Note we can also select a state $x \in \chi$ with the largest Voronoi region instead of the random selection, see [12] for more details and tradeoff considerations). Next, we select a mode $\sigma = (\kappa, \xi)$ randomly from the set of available modes to drive the unicycle from state x to x', i.e. $x' = f(x, \sigma)$ where f is the state transition function. If $x' \notin \chi$, then augment χ accordingly. We continue in this manner for a large number of iterations N, and then denote $X_R^{\Sigma^\star} \approx \chi$. Once $X_R^{\Sigma^\star}$ is determined, we can use Q-learning to determine the optimal mode sequence to reach x_g starting from x_0. Note more detailed explanation of RRTs and Q-learning can be found in [10],[11],[12],[13],[21],and [22].

The resulting path for our problem (obtained using the technique briefly outlined here) is shown in Figure 5. The optimal control sequence in this case is

$$\begin{aligned} \bar{\sigma}^* = &(\kappa_g,\xi_1)(\kappa_o,\xi_3)(\kappa_g,\xi_1)(\kappa_o,\xi_3)(\kappa_g,\xi_1)(\kappa_o,\xi_3) \\ &(\kappa_g,\xi_1)(\kappa_o,\xi_1)(\kappa_g,\xi_1). \end{aligned}$$

So clearly $(\kappa_o,\xi_3)(\kappa_g,\xi_1)$ is repeated often in the optimal control program, thus it would be beneficial to replace it with a single mode (κ_n, ξ_1), where $\kappa_n = \alpha_g \kappa_g + \alpha_o \kappa_o$. Using the variational techniques given here, it is found that $\alpha_g^* = 0.211$ and $\alpha_o^* = 0.801$. Now we recalculate the optimal path with the new feedback mapping $\kappa_n(x)$ and again the three existing interrupts for its termination added to the mode set. The resulting path is shown in Figure 6 and the optimal control sequence is given by $\tilde{\sigma}^* = (\kappa_g,\xi_1)(\kappa_n,\xi_1)(\kappa_g,\xi_1)$. The augmentation of the motion alphabet results in great improvement in terms of the optimal mode sequence and the resulting optimal trajectory. Although we only designed the new feedback map to "merge" two modes, the overall affect of adding the new modes reduced the size of the control program from $|\sigma^*| = 9$ to $|\tilde{\sigma}^*| = 3$. Moreover, the complexity of the control program is reduced from $9 \cdot log_2(6) = 23.2647$ to $3 \cdot log_2(9) = 9.5098$.

IV. CONCLUSIONS

When humans acquire new motor skills, they are typically obtained from a combination of previously established skills.

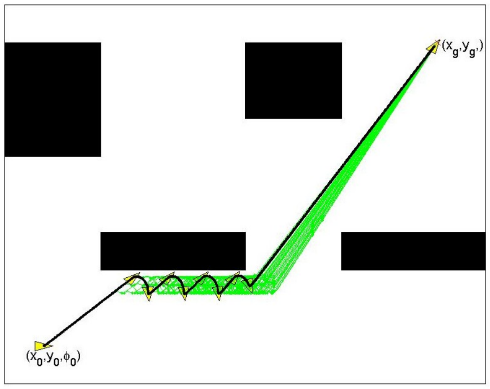

Fig. 5. The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the available set of modes.

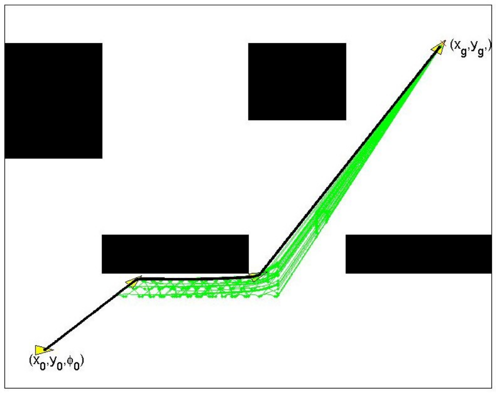

Fig. 6. The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the augmented set of modes.

This observation constituted the starting point for this work that addressed the following question: "Given a set of modes, what new modes (if any) should be added to the mode set in order to improve the overall system performance?" Our solution was based on two main assumptions. First, we begin with a characterization of the reachable set obtained from the original mode set. The new modes should be added in such a way so that frequently recurring mode combinations can be combined into single "meta-modes". Secondly, this combination is obtained through a linear combinations of the known modes (or any generalizing functions, such as the P. Hall basis). The solution was obtained using the calculus of variations, and two different numerical examples clearly illustrates the usefulness of the proposed method.

V. ACKNOWLEDGMENTS

The authors would like to thank Tryphon T. Georgiou for pointing out the matrix equation (17).

REFERENCES

[1] T. A. Henzinger. The theory of hybrid automata. *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society Press, 1996, pp. 278-292.

[2] 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*, Dec. 2003.

[3] M. Egerstedt. On the Specification Complexity of Linguistic Control Procedures. *International Journal of Hybrid Systems*, Vol. 2, No. 1-2, pp. 129-140, March & June, 2002.

[4] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035-2047, December 2003.

[5] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences* 57:94–124, 1998.

[6] A. Bicchi, A. Marigo, and B. Piccoli. On the reachability of quantized control systems. *IEEE Transactions on Automatic Control*, 4(47):546-563, April 2002.

[7] A. Bicchi, A. Marigo, and B. Piccoli. Encoding steering control with symbols. *IEEE International Conference on Decision and Control*, pages 3343-3348, 2003.

[8] E. Asarin, O. Bournez, T. Dang, O. Maler. Reachability Analysis of Piecewise-Linear Dynamical Systems *Hybrid Systems: Computation and Control*, 20-31 LNCS 1790, Springer, 2000

[9] I. Mitchell and C. J. Tomlin. Overapproximating Reachable Sets by Hamilton-Jacobi Projections *Journal of Scientific Computation*, Volume 19, Number 1, pp. 323-346, December 2003.

[10] A. Bhatia, and E. Frazzoli. Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems. *Hybrid Systems: Computation and Control. Springer-Verlag*, 2004.

[11] T. Mehta and M. Egerstedt. Learning Multi-Modal Control Programs. *Hybrid Systems: Computation and Control, Springer-Verlag*, March 2005.

[12] P. Cheng, Z. Shen, and S. M. LaValle. Using randomization to find and optimize feasible trajectories for nonlinear systems. *Proc. Annual Allerton Conference on Communications, Control, Computing*, pages 926–935, 2000.

[13] S. M. LaValle. From dynamic programming to RRTs: Algorithmic design of feasible trajectories. *Control Problems in Robotics*, pages 19–37. Springer-Verlag, 2002.

[14] L. Armijo. Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives. *Pacific Journal of Mathematics*, Vol. 16, ppm. 1-3, 1966.

[15] G. Lafferriere, H. J. Sussmann. A Differential Geometric Approach to motion Planning. *Nonholonomic Motion Planning, Z. Li and J.F. Canny, Eds, Kluwer Academic Publishers*, pp 235-270, 1993

[16] J. A. Bathurin. *Lectures on Lie algebras, Akademie Verlag*, Berlin, 1978.

[17] H. J. Sussmann. Lie brackets and local controllabilty: a sufficient condition for scalar input systems. *SIAM J. Control and Optimization*, 21(5): 686-713, 1983.

[18] R. C. Thompson. Lecture 10 : Part I - Convergence domains of the Campbell Baker Hausdorff formula, *John Hopkins Lecture Notes*, 1988.

[19] W. Rossmann. *Lie Groups: An Introduction Through Linear Groups, Oxford University Press*, Chapter 1.3, 2002.

[20] T. T. Georgiou. Relative Entrophy and the multi-variable multi-dimesional Moment Problem. submitted to *IEEE Transaction on IT* CLN 04-326. Revised Dec, 2004.

[21] L.P. Kaebling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal Of Artificial Intelligence Research*, 1996.

[22] C.J.C.H. Watkins, and P. Dayan. Q-learning. *Machine Learning* 8(3/4):257-277, May 1992.