Behavior-Based Switch-Time MPC for Mobile Robots

Greg Droge, Peter Kingston, Magnus Egerstedt

***Abstract*—Model predictive control can be computationally intensive as it has to compute an optimal control trajectory at each time instant. As such, we present a method in which parametrized behaviors are introduced as a level of abstraction to give a finite representation to the control trajectory optimization. As these control laws can be designed to accomplish different tasks, the robot is able to use the presented framework to tune the parameters online to achieve desirable results. Moreover, we build on switch-time optimization techniques to allow the model predictive control framework to optimize over a series of given behaviors, allowing for an added level of adaptability. We illustrate the utility of the framework through the control of a nonholonomic mobile robot.**

I. Introduction

Model predictive control (MPC) is a control scheme which adds feedback to otherwise typically open-loop optimal control solutions [1]. This is beneficial as optimal control allows for the satisfaction of constraints while minimizing some defined cost, but sometimes suffers when uncertainties are introduced, e.g., [2]. MPC is able to add feedback by solving the optimal control problem at each time instant, applying one control input, and repeating the process, e.g., [1].

However, one drawback to MPC is the cost of computing the optimal control solution at every time instant. This comes from the fact that the state needs to be simulated into the future over some time horizon to find the optimal control trajectory. Unless a closed form solution is known, this typically requires solving a set of differential equations where some initial conditions and some final conditions are known, e.g., [2]. This is known as a two-point boundary value problem and while numerical solutions to this problem do exist (e.g., [2]), they are often computationally intensive, e.g., [3].

To remedy this computational burden, we look to outsource the state trajectory generation to behaviors designed to accomplish the desired task. While behavior-based control schemes constitute an entire class of robotic control paradigms [4], we will only consider those behaviors which can be considered as parameterized feedback control laws. In other words, given a system at time t with state $x(t)$ and control input $u(t)$ whose dynamics can be expressed as

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

we consider behaviors of the form $\kappa(x(t), \theta)$ where θ is a parameter vector. This allows the dynamics to be expressed as

$$\dot{x} = f(x(t), \kappa(x(t), \theta)). \tag{2}$$

Therefore, instead of optimizing over $u(t)$ for some time horizon, we optimize over a finite dimensional parameter vector which is considered constant over the time horizon. Thus we exchange the two-point boundary value problem for a parameter optimization problem.

Using this concept, we extend work presented in [5] where MPC was used to coordinate different schema-based behaviors. First, we generalize the approach to be applicable to control schemes which can be expressed as in (2). Second, we incorporate techniques from switched-time optimization (e.g., [6], [7]) to allow the robot to optimize over a series of behaviors during a single optimization step. These two contributions will allow for the application of this control method to a much broader class of applications.

The remainder of this paper will proceed as follows. In the next section we will present a framework and optimality conditions for a behavior-based MPC scheme. In sections III and IV, we develop two control schemes for a nonholonomic vehicle to illustrate the utility and versatility of the MPC framework. We will then end the paper with some concluding remarks in Section V.

II. Behavior-Based Switch-Time MPC

In this section, we will explain the type of behaviors we will utilize in our framework, layout the formulation of the behavior-based MPC, and finish by giving the first order necessary optimality conditions which can be used to solve the optimization problem at each time step.

A. Behavior-Based Control

We use the term behavior to infer the notion that we will be working with control laws that are capable of accomplishing certain tasks. More specifically, we seek to utilize behaviors which are tunable feedback control laws, as shown in (2), to generate state trajectories.

Email: {gregdroge,kingston,magnus}@gatech.edu.
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. This work was sponsored in part by the DARPA M3 Program

We are then able to optimize over the different tunable parameters to achieve the desired result.

Moreover, it may be desirable to have the robot execute a string of such behaviors, i.e. $(\kappa_0, \tau_0), (\kappa_1, \tau_1), ..., (\kappa_N, \tau_N)$, where τ_i indicates the time when the system switch from κ_{i-1} to κ_i. This allows us to write the dynamics of the system as

$$\dot{x} = f(x(t), \kappa_i(x(t), \theta_i)) \text{ for } \tau_i \leq t < \tau_{i+1}, \quad (3)$$

which we simplify as

$$\dot{x} = f_i(x(t), \theta_i) \text{ for } \tau_i \leq t < \tau_{i+1}. \quad (4)$$

Utilizing this form of behaviors, we can build upon a wealth of different control applications which all use some form of parameterized control. Examples include schema-based behaviors [4], gait design for robotic snakes [8] and legged locomotion [9], orbiting for unmanned aerial vehicles [10] and ground vehicle obstacle avoidance [11], and even potential fields methods which are used in a wide variety of robotic motion applications [12], [13], just to name a few.

B. MPC Framework

As a naive parameter assignment can lead to a poor outcome, we present an MPC scheme which will allow for the parameters to be optimized online, admitting feedback into the parameter selection. To do so, we build upon results from switch time optimization (e.g., [6], [7]) to optimize over both the parameters associated with each behavior as well as the time to switch between behaviors.

To find the optimal parameters and switch times at each time step, we present a cost which related to a general form found in many optimal control texts, .e.g. [2], as follows

$$J(\tau, \theta) = \sum_{i=0}^{N} \int_{\tau_i}^{\tau_{i+1}} L_i(x, \theta_i, O(t_0))dt + \quad (5)$$
$$\Phi(\theta) + \Psi((x(\tau_{N+1}))$$
$$\text{s.t. } \dot{x} = f_i(x, \theta_i) \text{ for } \tau_i \leq t < \tau_{i+1},$$

where $x(t) \in \mathbb{R}^n$ is the state at time t, $\tau = \{\tau_i\}$ is a set of switch times, θ_i denotes the parameters that will be used on in the dynamics on the interval $t \in [\tau_i \tau_{i+1}]$, $\theta = \{\theta_i\}$, and $O(t_0)$ denotes the environmental data available to the robot at time t_0, when the optimization takes place. For ease of notation we allow $\tau_0 = t_0$ and $\tau_{N+1} = t_0 + \Delta$ where Δ is the time horizon of the optimal control problem.

By formulating our cost and dynamics, we can then define our MPC strategy as follows.

Behavior-Based MPC

1) Minimize (5) with respect to the behavior parameters, θ_i, and the time instances to switch between these parameters, τ_i.
2) Apply the behavior associated with the first set of parameters for one time instant.
3) Repeat.

C. First Order Optimality Conditions

In order to minimize (5) with respect to the desired variable, we present the first order necessary conditions of optimality which can be used with gradient decent strategies to find the optimal parameters (e.g. [14]).

Theorem 2.1: The first order necessary conditions of optimality for optimizing (5) with respect to the switch times, τ_i, and the parameter vectors, θ_i, are given by

$$\frac{\partial J}{\partial \tau_i} = \Big(L_{i-1} - L_i + \lambda^T (f_{i-1} - f_i)\Big) = 0 \quad (6)$$

$$\frac{\partial J}{\partial \theta_i} = \xi_i(\tau_i) = 0 \quad (7)$$

where

$$\dot{\lambda} = -\frac{\partial L_i}{\partial x}^T - \frac{\partial f}{\partial x}^T \lambda, \quad (8)$$
$$\text{for } \tau_i \leq t < \tau_{i+1},\ i = 0, ..., N$$
$$\lambda(\tau_{N+1}) = \frac{\partial \Psi}{\partial x}(x(\tau_{N+1}))$$
$$\dot{\xi}_i = -\frac{\partial L_i}{\partial \theta_i}^T - \frac{\partial f}{\partial \theta_i}^T \lambda \quad (9)$$
$$\xi_i(\tau_{i+1}) = \frac{\partial \Phi}{\partial \theta_i}$$

Proof The proof of Theorem 2.1 follows standard variational methods. We first augment (5) with the dynamics:

$$\bar{J}(\tau, \theta) = \sum_{i=0}^{N} \int_{\tau_i}^{\tau_{i+1}} \Big(L_i(x, \theta_i, O(t_0)) + \quad (10)$$
$$\lambda^T (f_i(x, \theta_i) - \dot{x})\Big)dt + \Phi(\theta) + \Psi((x(\tau_{N+1}))$$

We now vary the switch times and parameter vectors as $\tau \to \tau + \epsilon v$ and $\theta_i \to \theta_i + \epsilon \gamma_i$ which causes the state to vary as $x \to x + \epsilon \eta$. Similar to [6] and [7] we can take the Taylor expansion and write

$$\frac{1}{\epsilon}\Big(\bar{J}(\tau + \epsilon v, \theta - \epsilon \gamma) - \bar{J}(\tau, \theta)\Big) = \quad (11)$$

$$= \sum_{i=0}^{N} \Bigg[\int_{\tau_i}^{\tau_{i+1}} \Big(\frac{\partial L_i}{\partial x} + \lambda^T \frac{\partial f}{\partial x} + \dot{\lambda}^T \Big) \eta dt + \int_{\tau_i}^{\tau_{i+1}} \Big(\frac{\partial L_i}{\partial \theta_i} - \lambda^T \frac{\partial f}{\partial \theta_i} \Big) dt \gamma_i + v_{i+1} \big(L_i - L_{i+1} + \lambda^T (f_i - f_{i+1}) \big)|_{\tau_{i+1}} + \frac{\partial \Phi}{\partial \theta_i} \gamma_i \Bigg] + \frac{\partial \Psi}{\partial x} \eta|_{\tau_{N+1}}$$

Allowing λ to be defined as in (8) and ξ_i to be defined as

$$\xi_i(t) = \int_t^{\tau_{i+1}} \Big(\frac{\partial L_i}{\partial \theta_i} - \lambda^T \frac{\partial f}{\partial \theta_i} \Big) ds + \frac{\partial \Phi}{\partial \theta_i}, \tag{12}$$

we can simplify (11) to

$$\frac{1}{\epsilon} \Big(\bar{J}(\tau + \epsilon v, \theta - \epsilon \gamma) - \bar{J}(\tau, \theta) \Big) = \tag{13}$$

$$= \sum_{i=0}^{N} \Big(\xi_i(\tau_i) \gamma_i + v_{i+1} \big(L_i - L_{i+1} + \lambda^T (f_i - f_{i+1}) \big)|_{\tau_{i+1}} \Big)$$

which gives the partials in (6) and (7). We can also simplify the costate ξ_i to get the dynamics given in (9) by differentiating (12) with respect to t. ∎

III. A Vector-Field Approach to Motion Control for Nonholonomic Mobile Robots

To illustrate the utility of the MPC approach presented in the previous section, we present a control method amenable to the proposed framework which will allow a nonholonomic mobile robot to follow a vector field. This has an array of applications as vector field approaches are the basis of many control schemes for mobile robots, e.g. [4], [10], [11], [12], [13]. More importantly, however, this provides for a good example for the MPC framework as the behavior is able to overcome the nonholonomic constraints and the MPC scheme is able to optimize over the parameters of the behaviors. We will proceed by outlining the control law, giving optimality conditions necessary for use with Theorem 2.1, and ending with an example utilizing the MPC framework for orbiting.

A. Non-Linear Unicycle Control

To account for the motion constraint present in mobile platforms, we utilize the unicycle motion model which is a common method used to model planar motion in mobile robotic platforms, e.g., [11], [12]. Figure 1 shows a diagram of a typical unicycle robot where the state dynamics are given as

$$\dot{x} = \begin{bmatrix} v \cos(x_3) \\ v \sin(x_3) \\ \omega \end{bmatrix}, \tag{14}$$

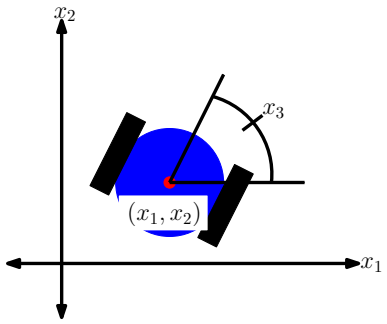

Fig. 1: This figures shows a diagram of the states of a unicycle robot. (x_1, x_2) gives the position and x_3 gives the orientation.

and v and w correspond to the input translational and rotational velocities of the vehicle, respectively.

One common method of making a unicycle robot follow a vector field is to use a proportional-derivative (PD) control, e.g., [11]. However, due to the differential term, this type of control is difficult to use in optimization as the partial derivative of the control is needed. Therefore, we present a nonlinear unicycle control which is capable of following a vector field while being easily incorporated into our optimization framework.

To do so, we give an alternate expression for the unicycle dynamics which makes our controller very simple to express. The unicycle dynamics given in (14), with control input $u = \begin{bmatrix} v & \omega \end{bmatrix}^T$, can be rewritten in Cartesian coordinates as

$$\begin{aligned} \dot{p} &= vh \\ \dot{h} &= \omega J h \end{aligned} \tag{15}$$

where $p = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, $h = \begin{bmatrix} \cos(x_3) & \sin(x_3) \end{bmatrix}^T$, and

$$J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{16}$$

is the 90-degree rotation matrix. The state space of (15) is $X \triangleq \mathbb{R}^2 \times S_1$ – the plane (which represents positions), together with the circle (which represents orientations).

Given a compact workspace $\Omega \subset \mathbb{R}^2$ containing the origin, together with positive definite,[1] continuously-differentiable function $U : \Omega \to \mathbb{R}$, we define the controller,

$$\begin{aligned} \omega &= -k_\omega (\operatorname{grad} U)^T J h \\ v &= -k_v (\operatorname{grad} U)^T h \end{aligned} \tag{17}$$

where $k_\omega, k_v > 0$ are arbitrary constants.

The controller, (17), globally asymptotically stabilizes the robot to the origin of the workspace, without regard

[1]This positive-definiteness requirement can be omitted, in which case stabilization to a *local minima* is guaranteed.

to the robot's orientation. This is stated formally by the next theorem:

Theorem 3.1 (Unicycle Stabilization): Let $\Omega \subset \mathbb{R}^2$ be a compact set (the workspace), and $U : \Omega \to \mathbb{R}$ a positive definite, continuously-differentiable scalar field. Then (17) globally stabilizes the dynamics in (15) to the set $X_0 \triangleq \{(p,h) \in \mathbb{R}^2 \times S_1 \mid p = \mathbf{0}\}$.

Proof: The proof uses LaSalle's Theorem, and the candidate Lyapunov function,

$$X \ni (p,h) \xrightarrow[V]{} U(p) \ ; \tag{18}$$

i.e., we treat U, which is a function defined only on the workspace Ω, as a function V on the entire state space $X = \Omega \times S_1$.

Differentiating V in time and substituting from (15) and (17), we obtain

$$\dot{V} = -k_v \left\langle \operatorname{grad} U, h \right\rangle^2 \leq 0 \ , \tag{19}$$

so the nonincreasing-Lyapunov-value condition of LaSalle's Theorem is satisfied. We will denote by E the set of states where (19) holds.

Moreover, $\dot{V} = 0$ only when $\operatorname{grad} U \perp h$, in which case (17) implies

$$|\omega| = k_\omega || \operatorname{grad} U|| \tag{20}$$

and $\dot{x} \neq 0$ (so long as $|| \operatorname{grad} U|| \neq 0$). Consequently, X_0 is not just positively-invariant, but also the largest positively-invariant set in E, and by LaSalle's Theorem is the positive limit set of (15) under the controller (17). ■

This shows that the control law will follow a gradient field to a minima. For a general vector field, where $u \in \mathbb{R}^2$ is an element of that field, we can modify (17) to follow the vector field as

$$\begin{aligned} \omega &= -k_\omega \|u\| \sin(\phi) \\ v &= -k_v \|u\| \cos(\phi), \end{aligned} \tag{21}$$

where $\phi = \operatorname{atan2}(u_2, u_1) - x_3$. This can be found by noting that $Jh \perp h$ and the use of the definition of the inner product (i.e. $\langle a, b \rangle = \|a\|\|b\| \cos(\psi)$).

B. Partials for Cost Optimization

While the given unicycle control is able to follow a vector field, it is also important in this context for its ability to easily be incorporated into the optimization framework presented in Section II. To make this clear, we set $k_v = k_\omega = 1$ and give the control in the form of (4) as

$$f(x,\theta) = \begin{bmatrix} \|u(x,\theta,O)\| \cos(\phi) \cos(x_3) \\ \|u(x,\theta,O)\| \cos(\phi) \sin(x_3) \\ \|u(x,\theta,O)\| \sin(\phi) \end{bmatrix}. \tag{22}$$

Since u is an element of a vector field, it can be a function of the state, x, the environmental data present to the robot, O, as well as a vector of parameters, θ.

Defining the control as such allows us to write the following theorem which can then be used to find the optimal parameters at each time step in conjunction with Theorem 2.1 and a definition of the vector field.

Theorem 3.2: The partial of (22) with respect to a parameter γ, where γ can be x_i or an element of θ_i given in (5), is given as

$$\frac{\partial f}{\partial \gamma} = \begin{bmatrix} \frac{\partial v}{\partial \gamma} \cos(x_3) - v \sin(x_3) \frac{\partial x_3}{\partial \gamma} \\ \frac{\partial v}{\partial \gamma} \sin(x_3) + v \cos(x_3) \frac{\partial x_3}{\partial \gamma} \\ \frac{\partial w}{\partial \gamma} \end{bmatrix}, \tag{23}$$

where

$$\frac{\partial v}{\partial \gamma} = \frac{1}{\|u\|} u^T R(\phi) \frac{\partial u}{\partial \gamma} + \|u\| \sin(\phi) \frac{\partial x_3}{\partial \gamma},$$

$$\frac{\partial w}{\partial \gamma} = \frac{1}{\|u\|} u^T R(\phi - \frac{\pi}{2}) \frac{\partial u}{\partial \gamma} - \|u\| \cos(\phi) \frac{\partial x_3}{\partial \gamma},$$

$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix},$$

and $v = \|u\| \cos(\phi)$.

Proof The derivation comes directly from taking the partial derivative with respect to (22) and algebraic simplification using rotation matrices. ■

C. Orbit Example

To demonstrate the ability of the MPC framework presented in Section II to optimize the parameters of the behavior, we present an orbiting example. Orbiting is often accomplished by having the vehicle follow a vector field that creates a stable limit cycle [10], [11]. As such, we parameterize the control law given in [11] to allow our method to adapt the vector field online and express it as

$$u(x) = g_s \begin{bmatrix} \gamma & \omega_{orb} \\ -\omega_{orb} & \gamma \end{bmatrix} \hat{x}, \tag{24}$$

where

$$\gamma = g_{lc}\Big(r^2 - \|\hat{x}\|^2\Big),$$

$\hat{x} = x_p - c$, $x_p = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$ is the two-dimensional position of the robot, $c \in \mathbb{R}^2$ is the center of the orbit, $g_s \in \mathbb{R}$ is a gain on the speed, $g_{lc} \in \mathbb{R}$ is a gain on convergence to the limit cycle, $\omega_{orb} \in \mathbb{R}$ is the desired frequency of the orbit, and $r \in \mathbb{R}$ is the radius of the orbit. To adapt the vector field using our MPC scheme we allow the parameter vector to be optimized to be $\theta = \begin{bmatrix} g_s & g_{lc} & \omega_{orb} & r \end{bmatrix}$.

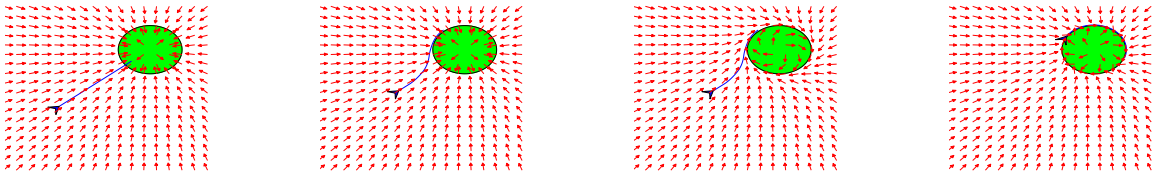

Fig. 2: This shows different snapshots in time of the adaptation of the vector-field given in equation (24) executed by the behavior in (22). The robot is shown with its planned trajectory extending from it in each case. The middle two images are actually the same time instance where the middle-left image shows the vector field produced in the first time window and the middle-right image shows the vector field produced in the second time window.

The goal we set to accomplish is to approach a desired orbit while maintaining a given velocity, v_d, and with as little angular velocity as possible. Therefore we define our instantaneous cost as

$$L_i = \frac{\rho_1}{2}(v - v_d)^2 + \frac{\rho_2}{2}\omega^2, \tag{25}$$

and our terminal cost as

$$\Psi = \frac{\rho_3}{2}\big(\|x - c\| - r\big)^2, \tag{26}$$

and set $\Phi = 0$.

Figures 2 illustrates the result of using the MPC framework to adapt the parameters in order to minimize the cost. We were also able to see improvement as we used a string of behaviors where each behavior was the orbiting behavior with separate parameters to be optimized. This allowed for the anticipation of needed changes especially as the robot reached the point where it began circling.

IV. A Circular Arc Approach to Navigation

To further illustrate the versatility of the proposed MPC approach, we provide a second method of control for unicycle motion. It builds on the concept presented in [15] where the robot is able to perform well by planning circular arcs through constant velocity inputs over some time horizon. It then re-evaluates this plan at each time step. As opposed to [15] where the velocities are chosen from some finite set, we allow the MPC scheme to find the optimal values at each time step. This allows the robot to navigate around obstacles by continuously optimizing with the most current information. Moreover, the MPC framework allows multiple arcs to be planned in sequence. This becomes quite useful in planning paths through an unknown environment.

To perform the task, we work straight from the dynamics given in (14) where our parameter vector can be defined as $\theta_i = \big[\omega, v\big]$. We define the costs as

$$L_i = \frac{\rho_1}{2}(v - v_d)^2 + \frac{\rho_2}{2}\omega^2 + \rho_3 \sum_{i=1}^{N_s} r(x, o_i), \tag{27}$$

$$r(x, o_i) = \exp\Big(-(x - o_i)^T \begin{bmatrix} \rho_4 & 0 \\ 0 & \rho_4 \end{bmatrix} (x - o_i)\Big),$$

$$\Psi = \frac{\rho_5}{2}\|x - x_g\|^2, \tag{28}$$

where x_g is the goal position, o_i is the i^{th} obstacle measurement of N_s sensor measurements, and $\{\rho_i\}$ are adjustable weights in the cost function. The cost structure in L_i allows the robot to maintain a given translational velocity while punishing high rotational velocities and proximity to obstacles. Also, as Ψ is a terminal cost, it encourages progress toward the goal without the need to move directly towards it at each time instant.

To evaluate the MPC framework, we ran the navigation simulations shown in Figure 4. We assume that we have $N_s = 16$ sensors distributed evenly about the

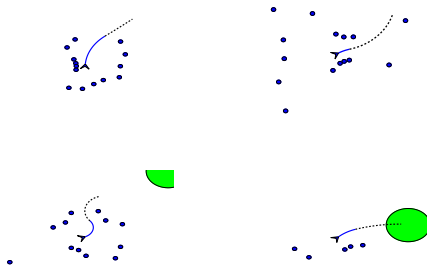

Fig. 3: This shows an example of different trajectories generated using the MPC framework with two modes. The robot is shown as a triangle with the expected trajectory extending from it. The solid line corresponds to the contribution from the first mode and the dotted line corresponds to the second mode. The small circles show the sensor measurements.

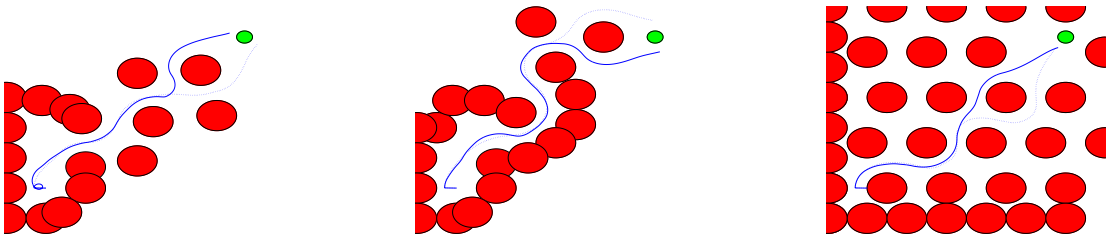

Fig. 4: This shows the different environments the robot navigated through along with a couple of representative trajectories in each environment. Obstacles are shown in red while the goal position is shown in green.

robot and that the robot plans only according to the sensor measurements that it has at the moment without any mapping. This is shown in Figure 3 which shows multiple "snapshots" of the robot traversing through an environment.

The results of the simulations show the utility of our MPC framework. The robot was able to successfully navigate the three different environments without changing any of weights on the costs. This shows that it is able to adapt to the different enviroments online. This example also shows the benefit of allowing multiple switch times as using a single set of parameters did not allow for the traversal of two of the environments.

V. Conclusion

In this paper we have presented an MPC strategy which utilizes the ability of behaviors to create desirable trajectories, exchanging a two-point boundary value optimization problem for a parameter optimization problem. We demonstrated the versatility of this method in Sections III and IV through two different examples for the control of a nonholonomic mobile robot. Both examples showed the ability of the robot to adapt the behaviors online to achieve the desired result. In particular, in the example in Section IV the adaptability of the MPC framework was emphasized as the robot was able to successfully navigate through different environments without changing any of the weights on the costs. We also saw the utility of allowing the optimization framework to anticipate changes in the behaviors through the addition of optimal switch times.

References

[1] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[2] D. Kirk, *Optimal control theory: an introduction*. Dover Pubns, 2004.

[3] K. Ling, B. Wu, and J. Maciejowski, "Embedded model predictive control (mpc) using a fpga," in *Proc. 17th IFAC World Congress*, 2008, pp. 15 250–15 255.

[4] R. Arkin, *Behavior-based robotics*. The MIT Press, 1998.

[5] G. Droge and M. Egerstedt, "Adaptive look-ahead for robotic navigation in unknown environments," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 1134–1139.

[6] M. Egerstedt, Y. Wardi, and F. Delmotte, "Optimal control of switching times in switched dynamical systems," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 3, dec. 2003, pp. 2138 – 2143 Vol.3.

[7] P. Martin and M. Egerstedt, "Optimization of multi-agent motion programs with applications to robotic marionettes," *Hybrid Systems: Computation and Control*, pp. 262–275, 2009.

[8] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, 2009.

[9] A. Jan and Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008.

[10] D. Nelson, D. Barber, T. McLain, and R. Beard, "Vector field path following for miniature air vehicles," *Robotics, IEEE Transactions on*, vol. 23, no. 3, pp. 519–529, 2007.

[11] D. Kim and J. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robotics and Autonomous Systems*, vol. 42, no. 1, pp. 17–30, 2003.

[12] S. M. LaVelle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[13] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 5, pp. 501–518, 1992.

[14] S. Boyd, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

[15] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, 1997.