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 an element of feedback and robustness 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], [3]. 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., [4].

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 [5], we will only consider those behaviors which can be considered as parameterized feedback control laws. In other words, given a system whose dynamics can be expressed as

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

where $x(t)$ is the state and $u(t)$ is the control at time t. We consider behaviors which can be expressed as a feedback control law 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 finite dimensional parameters. Thus we exchange the two-point boundary value problem for a parameter optimization problem.

This idea of optimizing over behaivors extends the work presented in [6] where MPC was used to coordinate different schema-based behaviors [5]. We extend this work with two significant contributions. First, we generalize the approach to be applicable to any behavior that allows the dynamics to be expressed as in (2). Second, we incorporate techniques from switched-time optimization (e.g., [7], [8]) to allow the robot to optimize over a series of behaviors during a single optimization step. The framework we present will optimize over both the parameters associated with the behaviors as well as the time to switch between each behavior. These two contributions will allow for the application of this control method to a much broader class of systems and applications.

To illustrate the operation and versatility of the proposed framework, we present two control schemes for a nonholonomic mobile robot which are amenable to the framework. As vector fields are a common motion control framework (e.g., [5], [9], [10], [11], [12]), we present a nonlinear control law which will allow a nonholonomic vehicle to follow a vector field that we adapt based on our MPC strategy. This will demonstrate the ability of the framework to optimize over a given behavior. Second, we will present a method of control based on [13] where inputs are held constant over a given time period. This will show the utility of multiple switch times as well as an application to a behavior which cannot achieve the task without adaptation.

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

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 section III and IV, we develop two control schemes for a nonholonomic vehicle to illustrate the utility and versatility of the MPC framework. We will end the paper with some concluding remarks in Section V.

II. Behavior-Based Switch-Time MPC

MPC is a method of introducing feedback into the solution of an optimal control problem by performing the optimization at each time instance, e.g., [1]. To reduce computation, we build upon a concept introduced in [6] as we present a method which depends upon behavior-based control schemes to generate the state trajectory. This adds a level of abstraction which allows us to turn the two-point boundary value problem into a parameter optimization problem. 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 utilize 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 to generate state trajectories and then optimize over the different tunable parameters to achieve the desired result. As such, when we refer to behavior, we are referring to any control scheme in which the resulting dynamics only depend on the current state and a tunable vector of parameters as shown in 2.

This builds upon a wealth of different control applications which all use some form of parameterized control. Examples include schema-based behaviors [5], gait design for robotic snakes [14], [15] and legged locomotion [16], 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 [9], [12], just to name a few.

B. MPC Framework

As can be seen in Figure 1, a naive parameter assignment can quickly lead to a poor outcome. Therefore, we present an MPC scheme which will allow for the parameters to be optimized online, admitting feedback into the parameter selection.

Moreover, it may be desirable to have the robot execute a string of behaviors in order to accomplish a desired task. This could also be used as a method of anticipating needed changes in the parameters of a behavior. Therefore the same behavior could be executed with different parameters, as is done in Sections III and IV.

In either case, we assume the robot has a given string of behaviors to execute, $(\kappa_0, \tau_0), (\kappa_1, \tau_1), ..., (\kappa_N, \tau_N)$, 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

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

We build upon results from switch time optimization (e.g., [7], [8]) 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 minimize a cost of the form

$$J(\tau, \theta) = \sum_{i=0}^{N} \int_{\tau_i}^{\tau_{i+1}} L_i(x, \theta_i, O(t_0))dt + \quad (5)$$

$$\sum_{i=1}^{N-1} \Phi_i(\theta_i, \theta_{i-1}) + \Psi((x(\tau_{N+1}))$$

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

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.

This cost has a general form to allow for different applications. It has an instantaneous term L_i which is a function of the state, environment, and the parameters of the behavior during the time interval $t \in [\tau_i, \tau_{i+1}]$. An example of this could be a cost on proximity to obstacles and control input. It also includes a cost, Φ_i, which is a function of the parameters used before and after the switch time τ_i. An example could be a function which punishes the difference between the parameters and some nominal value. Finally, there is a terminal cost, Ψ, on the state which could be used to ensure that the robot is progressing towards a set of goal states.

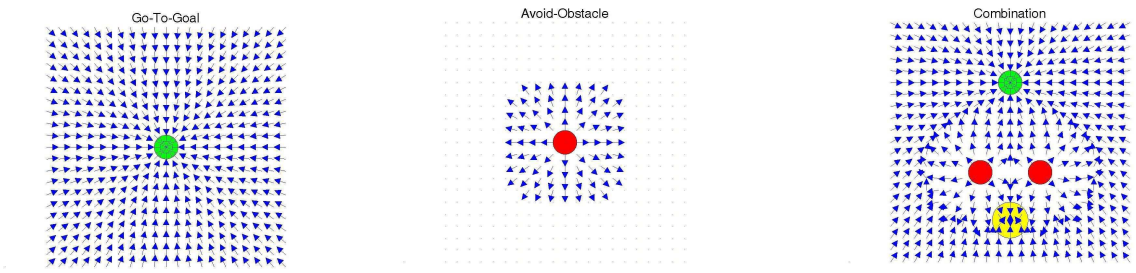

Fig. 1: This figure shows an example of Behavior-Based design for navigation. Goal positions are shown in green, obstacles in red, and local minima in yellow. The far right image shows an example of combining the vector fields in a manner which can produce a local minima

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 which can then be used with any gradient descent methods (see, for example [17]).

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

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

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

where

$$\dot{\lambda} = -\frac{\partial L_i}{\partial x}^T - \frac{\partial f}{\partial x}^T \lambda, \tag{8}$$

$$\text{for } \tau_i \leq t < \tau_{i+1},\ i = 0, ..., N$$

$$\lambda(\tau_{N+1}) = \frac{\partial \Psi}{\partial x}(x(\tau_{N+1}))$$

$$\dot{\xi}_i = -\frac{\partial L_i}{\partial \theta_i}^T - \frac{\partial f}{\partial \theta_i}^T \lambda \tag{9}$$

$$\xi_i(\tau_{i+1}) = \begin{cases} \frac{\partial \Phi_1}{\partial \theta_0} & , i = 0 \\ \frac{\partial \Phi_i}{\partial \theta_i} + \frac{\partial \Phi_{i+1}}{\partial \theta_i} & , i = 1, ..., N-1 \\ \frac{\partial \Phi_N}{\partial \theta_N} & , i = N \end{cases}$$

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

$$\bar{J}(\tau, \theta) = \sum_{i=0}^{N} \int_{\tau_i}^{\tau_{i+1}} \Big(L_i(x, \theta_i, O(t_0)) + \tag{10}$$

$$\lambda^T(f_i(x, \theta_i) - \dot{x})\Big)dt + \sum_{i=1}^{N-1} \Phi_i(\theta_i, \theta_{i-1}) + \Psi((x(\tau_{N+1}))$$

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 [7] and [8] we can take the Taylor expansion and write

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

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

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

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

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. A very similar proof of the variation in the negative direction gives the same results. ■

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 for a nonholonomic mobile robot which is amenable to the proposed framework. Nonholonomic systems provide for an extra level of difficulty in control as they have non-integrable constraints on the state space, e.g., [18]. In motion control for mobile robots, this often comes about because the vehicle is not able to move orthogonal to the direction of motion. This provides for a good example for our framework as we can design behaviors to overcome these constraints and then optimize over the parameters affecting those behaviors.

In this section, we design a behavior which will allow for a kinematically-constrained vehicle 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 such as schema-based control [5], potential fields methods [9], [12], and orbiting methods [11], [10] to name a few.

The MPC framework presented in section II will allow for the adaptation of this vector field to minimize some cost while taking into account the effect it will have on the robot. We will now proceed by outlining the control law, giving optimality conditions necessary for use with Theorem 2.1, and ending with an example which will demonstrate the utility of the MPC framework as it adapts a vector field to reach an orbit in a desirable fashion while taking the dynamics into account.

A. Non-Linear Unicycle Control

To account for the motion constraint present in mobile platforms, we utilize the unicycle robot model which allows the robot to move with certain translational and angular velocities, e.g., [9]. This is a common method used to model planar motion in mobile robotic platforms such as cars and differential drive systems, e.g., [9], [11]. It is also closely related to the Dubins model used for unmanned aerial vehicles which assumes a constant velocity, e.g., [9]. Figure 2 shows a diagram of a typical unicycle robot where the state dynamics are given as

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

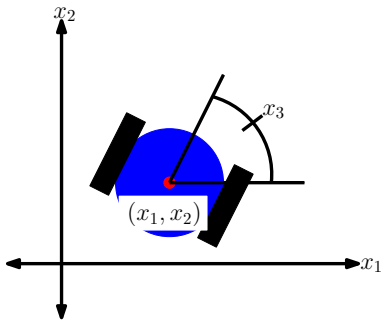

Fig. 2: This figures shows a diagram of the states of a unicycle robot. (x_1, x_2) gives the position and x_3 gives the orientation.

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

One common method of controlling a unicycle robot with a vector field is to use a proportional-derivative (PD) control, e.g., [11]. The basic idea being to allow the translational velocity to be determined by the magnitude of the vector field and then using PD control to adjust the angular velocity to give the desired direction of motion.

However, this is difficult to use with Theorem 2.1 as the partial derivative of the control is needed. This is made difficult by the differential term in the PD control. Therefore, we present a nonlinear unicycle control which is capable of following a vector field while being easily incorporated into our optimization framework. In the remainder of this section, we give an alternate expression for the unicycle dynamics which makes our controller very simple to express, before describing the controller and proving its asymptotic stabilization of the unicycle in the case of positive-definite potential fields.

The unicycle dynamics (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

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

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. For example, if the potential used in (17) is $U(p) = \frac{1}{2} p^T p$, then one obtains the specialization of (17),

$$\begin{aligned} \omega &= -k_\omega p^T J h &&= -k_\omega ||p|| \sin(\phi) \\ v &= -k_v p^T h &&= -k_v ||p|| \cos(\phi) \end{aligned} \tag{18}$$

where ϕ is the angular deviation of the robot's heading vector, from the vector towards the goal.

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

$$X \ni (p, h) \xrightarrow[V]{} U(p) \ ; \tag{19}$$

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

$$\dot{V} = -k_v \langle \operatorname{grad} U, h \rangle^2 \leq 0 \ , \tag{20}$$

so the nonincreasing-Lyapunov-value condition of LaSalle's Theorem is satisfied. We will denote by E the set of states where (20) holds.

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

$$|\omega| = k_\omega || \operatorname{grad} U || \tag{21}$$

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

$$\begin{aligned} \omega &= -k_\omega \|u\| \sin(\phi) \\ v &= -k_v \|u\| \cos(\phi). \end{aligned} \tag{22}$$

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 given vector field, it is more 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

$$f(x, \theta) = \begin{bmatrix} \|u(x, \theta, O)\| \cos(\phi) \cos(x_3) \\ \|u(x, \theta, O)\| \cos(\phi) \sin(x_3) \\ \|u(x, \theta, O)\| \sin(\phi) \end{bmatrix} \tag{23}$$

where, to be precise, $\phi = \operatorname{atan2}(u_2, u_1) - x_3$. 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 (23) with respect to a parameter γ, where γ can be x_i or an element of θ_i given in (5), is given as

$$\frac{\partial f}{\partial \gamma} = \begin{bmatrix} \frac{\partial v}{\partial \gamma} \cos(x_3) - v \sin(x_3) \frac{\partial x_3}{\partial \gamma} \\ \frac{\partial v}{\partial \gamma} \sin(x_3) + v \cos(x_3) \frac{\partial x_3}{\partial \gamma} \\ \frac{\partial w}{\partial \gamma} \end{bmatrix}, \tag{24}$$

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 (23) and recognizing the structure which can be simplified using the rotation matrices. ■

C. Orbit Example

To demonstrate the ability of the MPC framework presented in Section II to adapt 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 [11], [10]. As such,

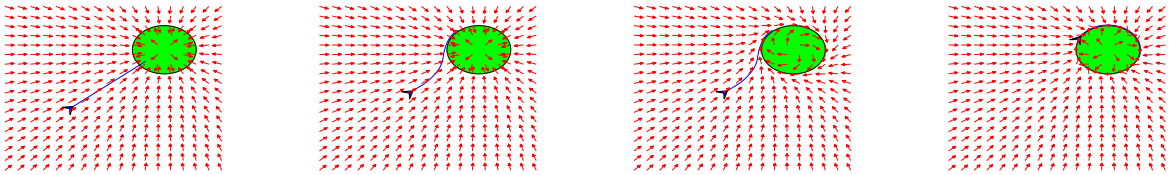

Fig. 3: This shows different snapshots in time of the adaptation of the vector-field given in equation (25) executed by the behavior in (23). The robot is shown with its planned trajectory extending from it in each case. The middle two images are actually the same time instance where the middle-left 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.

we parametrize the control law given in [11] to allow our method to adapt the vector field online and express it as

$$u(x) = g_s \begin{bmatrix} \gamma & \omega_{orb} \\ -\omega_{orb} & \gamma \end{bmatrix} \hat{x}, \tag{25}$$

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 $\theta = \begin{bmatrix} g_s & g_{lc} & \omega_{orb} & r \end{bmatrix}$.

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

$$L_i = \frac{\rho_1}{2}(v - v_d)^2 + \frac{\rho_2}{2}\omega^2, \tag{26}$$

and our terminal cost as

$$\Psi = \frac{\rho_3}{2}\big(\|x - c\| - r\big)^2, \tag{27}$$

We set $\Phi_i = 0$.

Figures 3 and 4 illustrate the results of using our MPC framework to minimize the cost. Figure 3 shows that the field changes to allow the robot to minimize the cost. It also shows the ability of multiple switch times to anticipate the needed changes. This is also demonstrated in Figure 4 which shows that significant improvements can be seen when multiple switches are made.

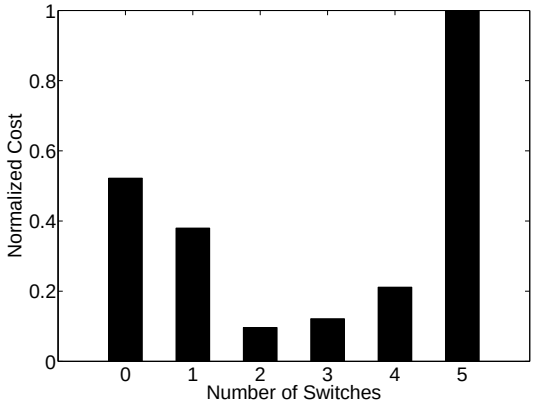

Fig. 4: This figure shows the costs associated with different numbers of switches between different sets of parameters. The costs are normalized so that the largest cost is scaled to one.

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 a unicycle motion which deals directly with the dynamics of a unicycle and provides an ideal setup for our MPC framework. It builds on the concept presented in [13] and modified in [19]. The basic idea is that for mobile navigation in unknown environments, the robot can perform well by choosing constant velocities over some time horizon. This gives the behavior of having the robot executing a circular arcs or straight lines for some given time horizon.

It is not difficult to see that a single set of parameters cannot successfully navigate a cluttered environment. In [13], [19], the authors present methods of choosing the constant velocities from a finite, admissible set of controls. This admissible set is defined in terms of avoiding obstacles and the velocities are chosen by selecting those that minimize a defined cost. In our method, we allow the cost to steer the robot away from the obstacles while navigating to the goal. Therefore, the MPC scheme adapts the velocities online without restricting them to a finite set. Moreover, we can have multiple time windows to allow for multiple arcs to be planned in sequence. This becomes quite useful in planning paths through an unknown environment.

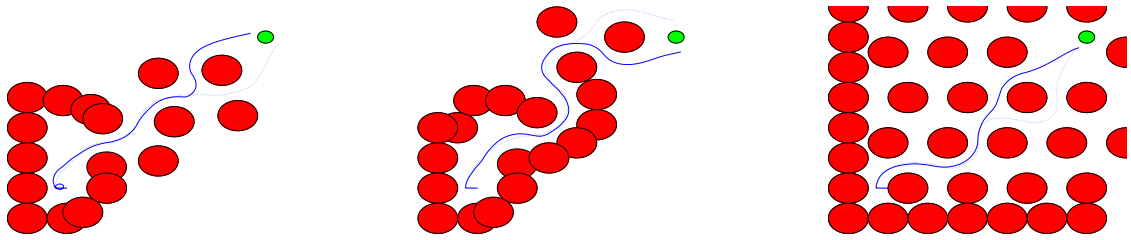

Fig. 5: This shows the different environments the robot navigated through along with a couple of representative trajectories in each environment. Obstacles are shown in red while the goal position is shown in green.

To perform the task, we work straight from the dynamics given in (14) where our parameter vector can be defined as $\theta_i = \big[\omega, v\big]$. To define our costs, we change (26) to have a term which punishes proximity to obstacles and modify (27) to punish distance from the goal. We write the costs as

$$L_i = \frac{\rho_1}{2}(v - v_d)^2 + \frac{\rho_2}{2}\omega^2 + \rho_3 \sum_{i=1}^{N_s} r(x, o_i), \qquad (28)$$

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

$$\Psi = \frac{\rho_5}{2}\|x - x_g\|^2, \qquad (29)$$

where x_g is the goal position, o_i is the i^{th} obstacle measurement of N_s sensor measurements, and $\{\rho_i\}$ are the 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 perform an evaluation of the utility of our MPC framework, we ran a series of navigation simulations through three different environments, the results of which can be seen in Figures 5 and 6. We assume that we have $N_s = 16$ sensors distributed evenly about the robot and that the robot plans only according to the sensor measurements that it has at the moment, in other words there is no mapping involved. This is shown in Figure 7 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

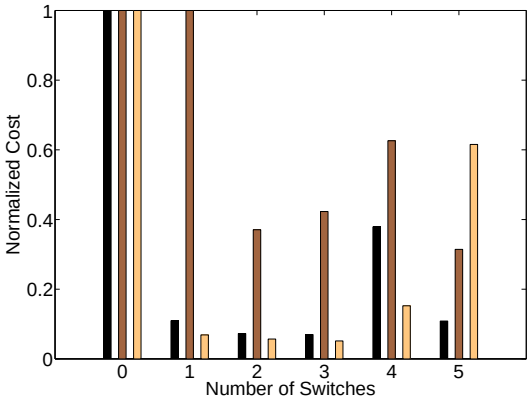

Fig. 6: This figure shows the costs associated with different numbers of switches between different sets of parameters. The costs are normalized so that the largest cost of each set is scaled to one. Each grouping of three bars correspond to the left, middle, and right environments shown in Figure 5

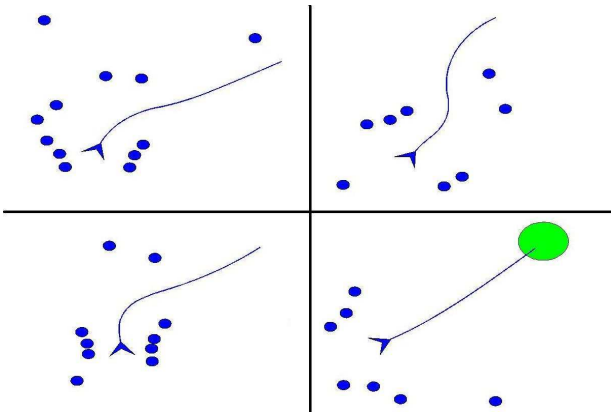

Fig. 7: This shows an example of the different trajectories generated at each time-step of the MPC framework. The robot is shown with the expected trajectory extending from it. This also shows the amount of information used at each time-step as the robot created the trajectory by planning over the sensor measurements shown as small circles.

adapt to the different enviroments online. This example also shows the benefit of allowing multiple switch times as the results in Figure 6 show that significant improvement is made by allowing the optimization to anticipate changes in the parameters.

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. Each example provides distinct behaviors to guide a nonholonomic mobile robot on a desirable trajectory, showing the applicability of the presented framework to different control schemes.

Both examples also 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. This provided for significantly reduced costs in the execution of the both of the examples.

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] A. Bryson and Y. Ho, *Applied optimal control: optimization, estimation, and control*. Hemisphere Pub, 1975.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] S. M. LaVell, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[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] 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.

[13] 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.

[14] 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.

[15] J. Gonzalez-Gomez, "Modular robotics and locomotion: Application to limbless robot," Ph.D. dissertation, Universidad Autonoma de Madrid, 2008.

[16] 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.

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

[18] H. Khalil and J. Grizzle, *Nonlinear systems*. Prentice hall, 1992, vol. 3.

[19] P. Ogren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance," *Robotics, IEEE Transactions on*, vol. 21, no. 2, pp. 188–195, 2005.