Trajectory Tracking among Landmarks and Binary Sensor-Beams

Benjamín Tovar Todd Murphey

***Abstract*— We study a trajectory tracking problem for a mobile robot moving in the plane using combinatorial observations from the state. These combinatorial observations come from crossing binary detection beams. A binary detection beam is a sensing abstraction arising from physical sensor beams or virtual beams that are derived from several sensing modalities, such as actual detection beams in the environment, changes in the angular order of landmarks around the robot, or recognizable markings in the plane. We solve the filtering problem from a geometric perspective and present its relation to linear recursive filters in control theory. Subsequently, we develop the acceleration control of the robot to track a given input trajectory, with a finite control set consisting on moving toward landmarks naturally modeling the robot as a switched dynamical system. We present experiments using an e-puck differential-drive robot, in which a useful estimate of the state for tracking is produced regardless of nontrivial uncertainty.**

I. Introduction

We adapt techniques developed for switched dynamical systems [3], [9], [10] to robots whose main source of information consists of combinatorial aspects of the environment, and whose control is limited to a finite set of primitives. In particular, we study a robot in the plane with control limited to moving toward landmarks, and whose observations of the state come from abstract binary detection beams [19], [20]. We naturally describe this robot as a switched dynamical system, in which each of the modes moves the robot closer to a particular landmark. We aim to design each of the modes to behave well over periods of open-loop operation, and to provide state feedback only when there is a combinatorial change in the observation of the sensor (see Figure 1). Our work is inspired by chaining a sequence of motion primitives such as in [6], [7]. Note that in our case, however, observations do not provide a perfect knowledge of the state, as it was assumed in [13] for computing navigation among landmarks.

Combinatorial observations decompose the environment into a countable number of cells; the robot obtains the same observation for any points inside a cell [4], [11]. This means that we somehow must convert the combinatorial information into metric information that a stabilizing control law may require. Moreover, the metric information generated when crossing a beam cannot depend on where the particular beam was crossed, since this information is not available during execution. In this regard, the key insight is that the decomposition of the plane is given by the *linear* segments supporting the abstract beams. Such linear segments appear naturally in several sensing modalities, such as a landmark entering the field-of-view of the robot, changes in the landmark's cyclic order around a robot [21], crossing inflection rays and bi-tangent complements of the boundary when new regions become visible [22], or crossing actual markings in the plane. As it will be developed in Section III, the update to state estimate is a function of the intercept of the line segment crossed. Given the limitations in sensing during online execution, we need to compute the sequence and time periods in which the landmarks should be chased in order to minimize a given cost for being away of the desired trajectory. Treating the robot as a switched dynamical system, our problem reduces to computing optimal switching times [5], [10] for the system.

Department of Mechanical Engineering, Northwestern Uni-versity, 2145 Sheridan Road, Evanston, IL 60208, USA.{b-tovar,t-murphey}@northwestern.edu

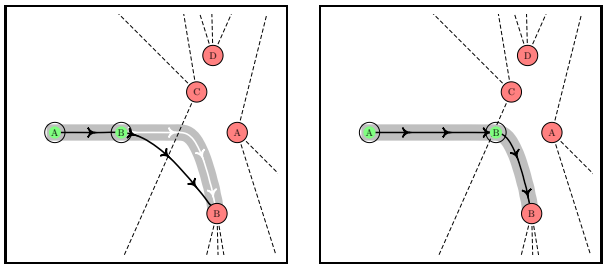

Fig. 1. Trajectory tracking by moving toward landmarks. The gray (thick) curve indicates the desired trajectory, and the black curve indicate the trajectory followed by the robot. Even in the presence of control uncertainty, and sensor observations consisting on combinatorial information, the robot is able to approximately track the desired trajectory. In this example, the combinatorial information comes from a change of the cyclic ordering of landmarks around the robot.

The paper is conceptually divided in two parts. In the first part we study the filtering problem: we show by geometric means that the configuration of the robot can be uniquely determined by crossing three non-parallel beams, and we make the connection to linear recursive filters in control theory. In the second part, we first assume that the robot has a perfect knowledge of the state, linear dynamics, and no uncertainty in control, and in this idealized setting, we compute the sequence of landmarks and switching times to track a given desired trajectory. Later, we assume an imperfect knowledge of the state, with the inclusion of combinatorial observations coming from the detection beams, and in Section V we adapt the previous ideas to a differential-drive robot with nonlinear dynamics, and present experiments with an *e-puck* robot[14].

II. Model

The robot is modeled as a point with configuration $\mathbf{q} = (q^x, q^y, q^\theta) \in Q \subset SE(2)$, state $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}}) \in TQ$, and

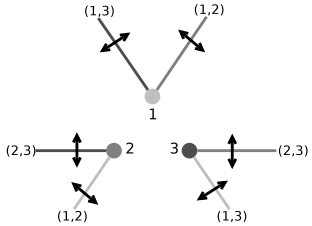

Fig. 2. Example of a detection beam through landmark crossings. Crossing a half-line swaps the order of the respective landmarks in the reading of the landmark order detector. Such half-lines are called *swap lines*.

state transition equation $\dot{\mathbf{x}} = f(\mathbf{x}, u)$ given by:

$$\dot{\mathbf{x}} = f^c(\mathbf{x}, u) = \begin{pmatrix} \dot{q}^x \\ \dot{q}^y \\ \dot{v} \\ \ddot{q}^\theta \end{pmatrix} = \begin{pmatrix} v\cos q^\theta \\ v\sin q^\theta \\ u^v \\ u^\theta \end{pmatrix} \tag{1}$$

with control signal $u = (u^v, u^\theta)$, and initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. With f^c, we have $v = \sqrt{(\dot{q}^x)^2 + (\dot{q}^y)^2}$ and we can write the state as $\mathbf{x} = (q, v, \dot{q}^\theta)$. We assume that the robot can only measure its state *locally*, meaning that the robot cannot immediately sense its configuration $\mathbf{q}$ with regard to a global reference frame, but that it can measure $\dot{\mathbf{q}}$ with regards to a local reference frame.

For this purpose, we further assume that the robot has an on-board clock. Between the global and local reference frames we only assume that they have the same "right-handedness".

The robot moves among a set of binary detection beams B, in which each beam $b \in B$ is assumed to be supported by a straight line. A binary detection beam detects whether the robot has crossed it, and it establishes a sensing abstraction that encapsulates several physical sensing modalities, such as actual detection beams in the environment (e.g., a sensor that indicates a customer enters a store), linear markings in the plane (see Figure 6), or *swap-lines* indicating the change of angular order of landmarks around a robot [21] (see Figure 2).

To facilitate the exposition, in the further developments we consider only *proper crossings of beams*, in which the robot cannot cross two beams at the same time (i.e., at the intersection of the beams), the robot cannot travel along a beam, the trajectory of the robot projected to $\mathbf{R}^2$ cannot have an inflection point along a beam, and no two beams in B have the same label (that is, the robot knows exactly which beam has been crossed).

III. The Filtering Problem

In this the section we study the problem of determining the configuration $\mathbf{q}$ of the robot when the state is unknown, and the robot moves among a known set of beams B. To facilitate the discussion, we first proceed by assuming that the robot has a compass and show that two beams crossings are needed and sufficient to localize the robot as long as such beams are not supported by parallel lines. Our argument will be purely geometric, but we will draw connections to the well-known case of Kalman-filtering. We will drop the compass assumption and show that three beams crossings are sufficient to localize the robot, as long as such beams are not parallel.

A. Robot with a compass

When the robot has a compass, the rotation of its local reference frame with respect to the beams' reference frame is known (i.e., q^θ is known). Therefore, we only need to determine q^x and q^y. Assume that the robot crossed the beams β_1, and $\beta_2 \in B$, in that order, and that the lines supporting β_1 and β_2 are not parallel. Further, assume that the line supporting β_i is given by $a_i q^x + b_i q^y + c_i = 0$.

Lemma 1: When the robot has a compass, after crossing β_i, there is a unique $d_i(\mathbf{q}) \in \mathbf{R}$ such that $a_i q^x + b_i q^y + c_i + d_i(\mathbf{q}) = 0$.

Proof: Let $\mathbf{q}_i = (q_i^x, q_i^y, q_i^\theta)$ be the unknown configuration of the robot when it crossed beam β_i, and let $\mathbf{q} = (q^x, q^y, q^\theta)$ be its current unknown configuration. When the robot crosses β_i, the line $a_i q_i^x + b_i q_i^y + c_i = 0$ represents the set of its possible configurations. As the robot moves, the slope of this linear set representing the possible configurations does not change, but the intercept does. With $q^x = q_i^x + \Delta x$ and $q^y = q_i^y + \Delta y$, we can write:

$$\begin{aligned} a_i q^x + b_i q^y + c_i + d_i(\mathbf{q}) &= 0 \\ a_i(q_i^x + \Delta x) + b_i(q_i^y + \Delta y) + c_i + d_i(\mathbf{q}) &= 0 \\ a_i \Delta x + b_i \Delta y + d_i &= 0 \\ -a_i \Delta x - b_i \Delta y &= d_i(\mathbf{q}) \end{aligned} \tag{2}$$

Since we assumed that the robot can measure $\dot{\mathbf{q}}$ with respect to a local reference frame, the value of Δx and Δy is completely determined by integrating $\dot{\mathbf{q}}$ for any initial condition on the line supporting β_i. ■

Corollary 1: When the robot has a compass, after the robot crosses the beams β_1 and $\beta_2 \in B$, in that order, and if they are supported by non-parallel straight lines, the robot lies on the intersection of the lines $a_1 q^x + b_1 q^y + c_1 + d_1(\mathbf{q}) = 0$ and $a_2 q^x + b_2 q^y + c_2 + d_2(\mathbf{q}) = 0$.

Proof: At the moment of the second crossing, $d_1(\mathbf{q}) = -a_1 \Delta x - b_1 \Delta y$ and $d_2(\mathbf{q}) = 0$. We have two equations for q^x, and q^y, which values are computed as:

$$\begin{aligned} q^x &= \frac{b_1 c_2 - b_2(c_1 + d_1)}{a_1 b_2 - a_2 b_1} \\ q^y &= \frac{a_2(c_1 + d_1) - a_1 c_2}{a_1 b_2 - a_2 b_1}. \end{aligned} \tag{3}$$

Note that $a_1 b_2 - a_2 b_1 = 0$ implies that the supporting lines are parallel. ■

B. Robot without a compass

When the robot does not have a compass, we will show that after crossing two beams supported by non-parallel lines, there are only two configurations in which the robot might be, and the configuration is completely determined by crossing a third, non-parallel beam.

Lemma 2: After crossing two distinct non-parallel beams, β_1 and β_2, there are at most two configurations in which the robot might be.

Proof: Consider the norm of the vector $(\Delta x, \Delta y)$, $l = ||(\Delta x, \Delta y)||$, and consider all the circles with center at $a_1 q^x + b_1 q^y + c_1 = 0$ and radius l. There are exactly two such circles that are tangent to $a_2 q^x + b_2 q^y + c_2 = 0$. Since the robot can only move in the direction given by its heading, this also determines two possible values for q^θ. ■

Corollary 2: After crossing three distinct beams, supported by non-parallel lines without a common point of intersection, the configuration of the robot is uniquely determined.

Proof: The proof consist of two applications of the previous Lemma, between beams β_1 and β_2, with the center of the circles on the line supporting β_1, and between beams β_3 and β_2, with the center of the circles on the line supporting β_3. Since the supporting lines are not parallel, and do not have any point of intersection, only two of the circles will be tangent at the same point along β_2, which determines the configuration of the robot. ■

C. Relations to Kalman Filtering

We now draw parallels between the state filtering presented above and the popular Kalman filtering approach. This is particularly insightful in the case of the robot with a compass. To start our discussion, suppose that the robot has previous knowledge of its configuration, say $\tilde{\mathbf{q}}_0 = (\tilde{q}_0^x, \tilde{q}_0^y, q_0^\theta)$, and after crossing some $\beta_1 \in B$, we find that $\tilde{\mathbf{q}}_1$, which is obtained by integrating $f^c(\mathbf{x})$ from $\tilde{\mathbf{q}}_0$ does not lie on the supporting line of β_1. By Lemma 1, we know the actual configuration $\mathbf{q}_1$ to be somewhere along the line supporting β_1. Now (without any indication that this is the correct manner to proceed) we project $\tilde{\mathbf{q}}_1$ into the line supporting β_1 by finding the closest point $(\tilde{q}_{1+}^x, \tilde{q}_{1+}^y)$ in the Euclidean sense between $(\tilde{q}_1^x, \tilde{q}_1^y)$ and the supporting line. Such point, easily found with a geometric argument, is given in matrix form by:

$$\begin{bmatrix} \tilde{q}_{1+}^x \\ \tilde{q}_{1+}^x \end{bmatrix} = \frac{1}{a_i^2 + b_i^2} \left(\begin{bmatrix} b_i^2 & -a_i b_i \\ -a_i b_i & a_i^2 \end{bmatrix} \begin{bmatrix} q_i^y \\ q_i^y \end{bmatrix} - \begin{bmatrix} a_i c_i \\ b_i c_i \end{bmatrix} \right) \tag{4}$$

In the case of the Kalman filter, we need to convert the combinatorial information into metric information that we can use to update the state. As in the previous discussions, the major constraint is that the robot obtains the same sensor reading regardless of where it crossed a beam. We can write the equations of the lines supporting the beams as:

$$\begin{bmatrix} a_i & b_i \end{bmatrix} \begin{bmatrix} q^x \\ q^y \end{bmatrix} = -c_i. \tag{5}$$

Further, writing

$$z_i = \begin{bmatrix} a_i & b_i \end{bmatrix} \begin{bmatrix} q^x \\ q^y \end{bmatrix} = H_i \begin{bmatrix} q^x \\ q^y \end{bmatrix}, \tag{6}$$

we can interpret $z_i = c_i$ as the unique metric value assigned to the combinatorial observation of crossing a particular beam regardless of where the beam was crossed, and $[a\ b\ 0\ \cdots]\mathbf{x}$ as the expected value of such observation. When a beam β_i is crossed, we can identify its supporting line and we write $H_i = [a_i\ b_i\ ]$ and z_i for the corresponding matrix and observation. Due to uncertainty on the initial position, we expect the quantity $c_i - H_i \begin{bmatrix} q^x \\ q^y \end{bmatrix}$ to be different from zero. Using the standard Kalman filter, assuming prediction covariance $P = \begin{bmatrix} \sigma_P^2 & 0 \\ 0 & \sigma_P^2 \end{bmatrix}$, and observation covariance $R = [\sigma_P^2 \sigma_R^2]$, (written in such a way for convenience), we find that the Kalman gain is:

$$\begin{aligned} K_i &= PH_i[HPH^T + R]^{-1} \\ &= \begin{bmatrix} \sigma_P^2 & 0 \\ 0 & \sigma_P^2 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} \left[\begin{bmatrix} a_i & b_i \end{bmatrix} \begin{bmatrix} \sigma_P^2 & 0 \\ 0 & \sigma_P^2 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix} + \sigma_P^2 \sigma_R^2 \right]^{-1} \\ &= \frac{1}{a_i^2 + b_i^2 + \sigma_R^2} \begin{bmatrix} a_i \\ b_i \end{bmatrix}. \end{aligned}$$

This gain is used to update the state as:

$$\begin{aligned} \mathbf{q}_{i+} &= \mathbf{q}_i + K_i(z_i - H_i \mathbf{q}_i) \\ &= \begin{bmatrix} q_i^x \\ q_i^y \end{bmatrix} + \frac{1}{a_i^2 + b_i^2 + \sigma_R^2} \begin{bmatrix} a_i \\ b_i \end{bmatrix} \left(-c_i - \begin{bmatrix} a_i & b_i \end{bmatrix} \begin{bmatrix} q_i^x \\ q_i^y \end{bmatrix} \right) \\ &= \frac{1}{a_i^2 + b_i^2 + \sigma_R^2} \left(\begin{bmatrix} b_i^2 & -a_i b_i \\ -a_i b_i & a_i^2 \end{bmatrix} \begin{bmatrix} q_i^x \\ q_i^y \end{bmatrix} - \begin{bmatrix} a_i c_i \\ b_i c_i \end{bmatrix} \right) \end{aligned} \tag{7}$$

which as $\sigma_R^2 \to 0$, gives the same update rule as Equation 4, but certainly does not quite look like Equation 3. In particular, Equation 7 depends explicitly on the initial position. To remove this dependency, without errors in the observations we can write:

$$\begin{aligned} c_1 &= -a_1(x_0 + \Delta x_{0,1}) - b_1(y_0 + \Delta y_{0,2}) \\ c_2 &= -a_2(x_0 + \Delta x_{0,1} + \Delta x_{1,2}) - b_2(y_0 + \Delta y_{0,1} + \Delta y_{1,2}), \end{aligned}$$

in which $\Delta x_{i,j}$ and $\Delta y_{i,j}$ are the net changes on the respective coordinates between crossings. These give two linear equations for (x_0, y_0), for which the value can be substituted into Equation 7 to give the same update rule as in Equation 3, which was obtained by geometric arguments.

IV. The Trajectory Tracking Problem

In this section we study the problem of trajectory tracking among a given set B of beams, using the filters developed in the previous section. Given a desired trajectory $\mathbf{x}_d$, the tracking problem consists of finding the control input signal u such that the following *cost* function is minimized:

$$J(u) = \int_{t_0}^{t_f} l(\mathbf{x}(t), u(t), \mathbf{x}_d)\mathrm{d}t, \tag{8}$$

subject to $\dot{\mathbf{x}} = f(\mathbf{x}, u)$, with initial time t_0 and final time t_f. In Equation 8, l is a real-valued, non-negative function. A switched robotic system is a dynamic system governed by a sequence of N dynamic models [10]:

$$\dot{\mathbf{x}} = f_i(\mathbf{x}), \;\; \tau_i \leq t < \tau_{i+1}, \tag{9}$$

with boundary condition $\mathbf{x}(t_0) = \mathbf{x}_0$. For $u : [t_0, t_f] \rightarrow \{1, \ldots, N\}$, and $f(\mathbf{x}, u, t) = f_{i=u(t)}(\mathbf{x})$ by minimizing Equation 8 we optimize the sequence of modes and switching times of the system in Equation 9 to track the desired trajectory $\mathbf{x}_d$. The minimization of Equation 8 depends on the particular dynamical system we are interested in. In the rest of this section we present the setting of landmarks in which we focus on, and how it is naturally expressed as a switched dynamical system.

A. Landmark Motion Primitives

The movements of the robot are described by *chasing landmarks*. By this we mean that the robot chooses a particular landmark, and moves towards it. Following [21], let L be a finite set of n indexed points in $\mathbf{R}^2$. A point $p_i \in L$ is referred to as a *landmark*, with *landmark label* i. We assume that the landmarks are labeled from 1 to n. For each landmark $p_i = (p_i^x, p_i^y) \in L$ we define a *motion primitive*, denoted by $\dot{\mathbf{x}} = \text{chase}_L(i, \mathbf{x})$. The motion primitive $\text{chase}_L(i, \mathbf{x})$ is constructed such that $f(\mathbf{x}, u, t)$ stabilizes at p_i. In other words, $\text{chase}_L(i, \mathbf{x})$ takes the robot to $(p^x, p^y, \theta, 0, 0)$, with some free orientation θ.

Our motion primitives are based on the linearization of Equation 1 assuming that the heading of the robot is oriented with the landmark. Further, it is easier to design the motion primitives for a landmark at the origin, asuming that the robot is somewhere along the positive $x-$axis, to then transform this solution to the corresponding location of the landmarks in the plane. With this, the linear approximation of Equation 1 consists of two double integrators, one modeling the change of distance d to the landmark, and the other the change of the robot's heading orientation ϕ with respect to the landmark:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{d} \\ \dot{v} \\ \dot{\phi} \\ \ddot{\phi} \end{pmatrix} = A\mathbf{x} + Bu, \tag{10}$$

with

$$A = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \text{ and } B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{11}$$

A motion primitive is based on the solution of a *Linear Quadratic Regulator* (LQR)[1], which generates the control input $u_l = (u^x, u^y)$ such that

$$J_{LQR}(u_l) = \int_{t1}^{t2} (x^T Q x + u_l^T R u_l)dt + x(t_2)^T P_2 x(t_2) \tag{12}$$

is minimized, subject to Equation 10. In J_{LQR}, P_2 is a positive-semi-definite matrix that encodes the cost of $x(t_2)$ being different from the origin, and both Q and R are positive-definite parameter matrices. The solution that minimizes $J_{LQR}(u_l)$ is given by $u_l = -R^{-1}B^T P(t)x(t)$, with $P(t)$ found by integrating from t_2 to t_1 the *Riccati equation*

$$A^T P(t) + P(t)A + P(t)BR^{-1}B^T P(t) + Q = -\dot{P}(t), \tag{13}$$

subject to the final cost $P(t_2) = P_2$. Since u_l is a feedback law, Equation 10 can be written now as:

$$\dot{\mathbf{x}} = A_{LQR}(t)\mathbf{x} \tag{14}$$

with $A_{LQR}(t) = A - BR^{-1}B^T P(t)$.

Now we introduce some notation to describe sequences of motion primitives. For a sequence of m motion primitives the order in which they are applied is encoded by a function $\sigma : \{1, 2, \ldots, m\} \rightarrow \{1, 2, \ldots, |L|\}$, in which the i_{th} motion primitive in the sequence is given by $\sigma(i)$. The times at which a motion primitive is applied are specified by the sequence $\tau = [\tau_1, \tau_2, \ldots, \tau_{m+1}]$. Given τ, $\sigma(i)$ is applied for $\tau_i \leq t < \tau_{i+1}$, with τ_1 and τ_{m+1} being the initial and final times, respectively. A motion primitive sequence is therefore completely specified given σ and τ.

Before going further, we note that the minimal sensing requirements for detecting beams and chasing landmarks may seem incompatible, since it is possible to perform a full metric SLAM using bearing only sensors [2]. However, we are not interested in building a map, but tracking a dynamic trajectory. In this regard, the "landmarks" are considered as part of the description of a switched system, rather than environment's features for the robot to be localized.

B. Tracking Trajectories with Perfect Control and Sensing

In this section we assume that the robot is perfectly localized, and can sense the precise position of the landmarks. To start our discussion, we solve an easier problem: we assume that the desired trajectory $\mathbf{x}_d$ is produced by following the motion primitive sequence (σ, τ^*). Suppose we are given σ (that is, the order in which the primitives appear), the initial and final times,

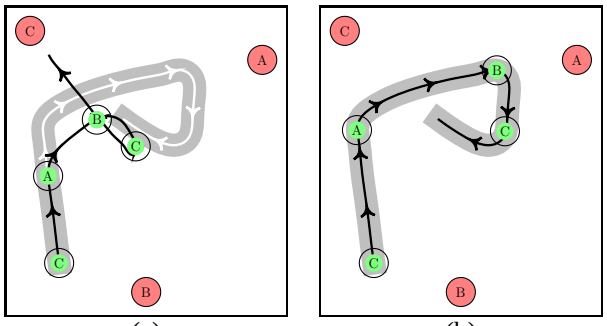

Fig. 3. Trajectory tracking when the desired trajectory is generated by moving toward landmarks. The figures show the initial and final iteration of the optimization, with the gray curve showing the desired trajectory and the black curve the actual trajectory followed. We label the black curve at the instant a new landmark is chased. In this case, the sequence of landmarks chased is $[C, A, B, C]$.

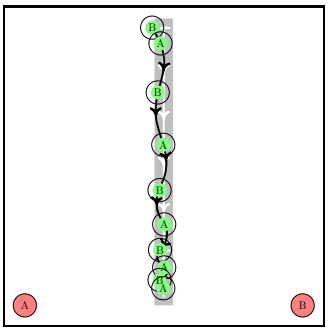

Fig. 4. Tracking a line with two landmarks. Switching back and forth between the landmarks tracked might provide an acceptable tracking performance, as it is shown in the figure. However, the figure also shows a potential drawback: note that as the robot gets closer to the landmarks, the number of switches increases.

and assume that the initial state of the robot coincides with $x_d(0)$, but we are not given τ^*. How do we compute switching times τ so that the difference between the actual trajectory of the robot x, and the desired trajectory x_d is minimized?

Following Equation 9 for switched dynamical systems, we write:

$$\dot{\mathbf{x}}(t, \sigma, \tau) = \text{chase}_L(\sigma(i), \mathbf{x}), \quad \tau_i \leq t < \tau_{i+1}. \tag{15}$$

Therefore, we have that $\mathbf{x}(t, \sigma, \tau^*) = \mathbf{x}_d(t)$. We encode the difference between $\mathbf{x}$ and $\mathbf{x}_d$ as a function of τ with the following cost function:

$$J(\tau) = \int_{\tau_1}^{\tau_{m+1}} ||\mathbf{x}(t, \sigma, \tau) - \mathbf{x}_d(t)|| \mathrm{d}t, \tag{16}$$

subject to Equation 15. We minimize Equation 16 using the method presented in [10], in which first and second derivatives of a cost functional subject to a switched dynamical systems are computed, and then are used in a gradient descent optimization. Since each motion primitive is a linear system, both first and second derivatives of Equation 16 exist and are continuous [10]. This allowed us to minimize Equation 16 with quadratic convergence using the Newton method with trust region [15] (see Figure 3).

C. Switching between motion primitives: General Case

In the previous example, we restricted $\mathbf{x}_d$ to be generated by a sequence of motion primitives. However, this is unusual, as we cannot expect the landmarks to coincide with the desired trajectories for the robot. We found that good approximations can be found by a heuristic procedure, in which the robot switches several times between two motion primitives to generate motion directions not originally available by tracking landmarks. This simple idea is reminiscent of procedures in motion planning such as in [12], or bang-bang control [17]. Consider for example Figure 4, in which $\mathbf{x}_d$ is a vertical line, with the corresponding trajectory approximation by chasing two landmarks.

The advantage to this scheme is that we can apply the optimization procedure presented earlier in the section with some minor modifications. We have to be careful: the optimal sequence of motion primitives will consider infinitely many switches [3] (that is, the optimal sequence *chatters*). To avoid this, minimization of Equation 16 is stopped once its value reaches a prescribed, small number $\epsilon > 0$.

In order to apply the optimization earlier in the section, there are three issues we need to solve: 1) we need to compute an initial motion primitive sequence; 2) we should specify when an additional motion primitive should be added to the sequence; and 3) we should specify when a motion primitive should be removed from the sequence.

The initial motion primitive sequence is found by partitioning x_d according to the pairs of landmarks its tangent falls between. For this, let $x_d(t) = (q_d(t), v_d, \dot{q}_d^\theta)$, with $q_d(t) = (q_d^x(t), q_d^y(t), q_d^\theta(t))$. Consider an infinite ray starting at $(q_d^x(t), q_d^y(t))$ with angle $q_d^\theta(t)$. As t varies, the ray sweeps over the landmarks, and the corresponding pairs are computed. Once the pairs of landmarks are found, we heuristically add four equidistant time switches per segment of x_d, which gives the motion primitive sequence (σ^0, τ^0). This sequence is refined with the following iteration:

1) For $k = 0, 1, \ldots$
   a) **Optimization.** Minimize Equation 16 subject to $\dot{\mathbf{x}}(t, \sigma^k, \tau^k)$, to obtain optimal switching times τ^{k*}.
   b) **Reduce switches.** If for some i, $\tau_i^{k*} = \tau_{i+1}^{k*}$, $(\sigma^{k+1}, \tau^{k+1})$ are constructed so that $\sigma^k(i)$ is not included.
   c) **Switches insertion.** If no switches were eliminated, then we compute the value of Equation 16 for each time segment $[\tau_i^{k*}, \tau_{i+1}^{k*}]$. If it exceeds a prescribed performance parameter, $(\sigma^{k+1}, \tau^{k+1})$ is constructed from (σ^k, τ^{k*}), with an additional motion primitive inserted between $(1/3)(\tau_i^{k*} + \tau_{i+1}^{k*})$ and $(2/3)(\tau_i^{k*} +$

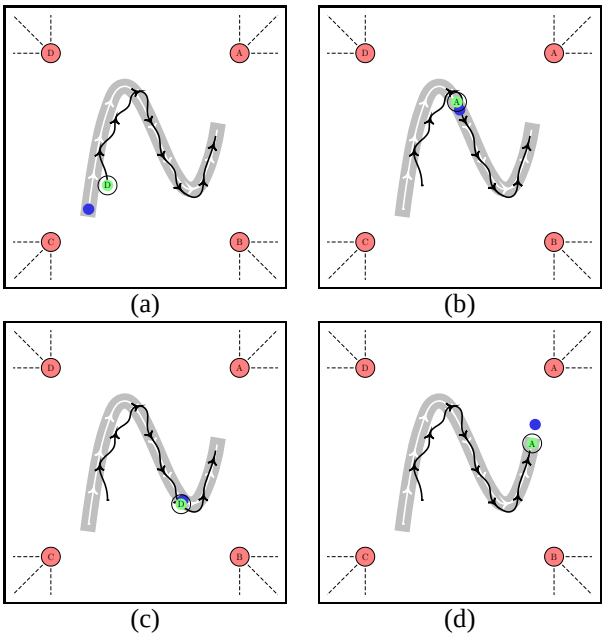

Fig. 5. Tracking a trajectory for an arbitrary arrangement of landmarks. The gray line shows the desired trajectory, and the black line the actual trajectory followed. The black line is labeled according to the instant a new landmark is chased. We show four snapshots, in which the blue (darker) disc represents the instantaneous desired state, and the labeled disc the actual state. The label indicates the landmark being followed at the given time.

τ_{i+1}^{k*}). The corresponding motion primitive is found in the same manner as for (σ^0, τ^0). The idea here is to reduce the value of Equation 16 by inserting a motion primitive in the sequence where the robot is "far" from the given trajectory.

d) **Iterate.** If no switches were reduced or added, return (σ^k, τ^{k*}). Otherwise increment k and iterate.

A computed example is shown in Figure 5.

V. Differential-Drive E-puck Implementation

Our objective in this section is to track a desired trajectory $\mathbf{x}_d$, with a robot that has imperfect information of the state during online execution. We assume that the robot can orient its heading with a given landmark, but its sensors cannot measure the distance towards the landmark, nor the angle between the robot and the landmark with respect to a global reference frame. As before, the robot has complete knowledge of the location of the detection beams, and we assume that it can determine which detection beam was crossed. The robot cannot, however, determine the location of the crossing along the beam. Further, we assume that internally the robot has a time sensor, and noisy observations for v and q^θ, useful for rough odometry estimates.

To test our preliminary ideas, we used an e-puck[14] robot to test out ideas on a real platform. The e-puck is a differential drive robot, thus their dynamics are nonlinear. In previous sections we defined the chase motions primitives by linearizing 1, but given that the linear model is not a good approximation when the robot's heading is not pointing near the location of a landmark (e.g., when the wheels' axis is collinear with the landmark), we must specify how the different motion primitives must be smoothly chained together. This is achieved by replacing Equation 14 with

$$\dot{\mathbf{x}} = A_{LQR}(t) \begin{pmatrix} d|\cos\phi|^k \\ v \\ \phi \\ \dot{\phi} \end{pmatrix}. \qquad (17)$$

for some $k \geq 1$. This straightforward change ensures that the robot moves towards the landmark at full speed only when the linear model is a good approximation, and make the robot to tend to rotate in place to point towards the landmark when this is not the case. In our experiments, we used $k = 2$. Since both first and second derivatives of Equation 17 exist and are smooth, we can use it instead of Equation 14 when defining the chase motion primitives, and proceed as before.

In our implementation, we decided to model the abstract sensor beams as linear markings in the plane that the e-puck can recognize with its ground sensors. By the principle of separation, while no observations are received (that is, beams crossings) our motion primitive is defined as before, operating as an open loop controller over an odometry estimate of the state. With respect to the e-puck camera, we found its field-of-view to be very limited for our purposes, and we decided to test our ideas representing the landmarks purely by odometry (that is, the robot has a landmark "map" and "simulates" landmark chasing based on odometry). That is, in our experiments the robot does not have a visual feedback for q^θ. Even though this camera did not fit our current model, it presented an interesting future direction since we can model the limited field-of-view as three distinct detection beams that move with the robot. We will explore these ideas in a future paper.

Figures 6 and 7 (with the accompanying video) show example runs of our implementation. In Figure 6, the robot performs about a dozen loops around the common point of the beams. Without the combinatorial updates, the odometry estimate effectively looses the robot near the end of the second loop. In Figure 7, the robot follows a complicated trajectory, in which the combinatorial updates give a correct estimate of the state even after the robot stabilizes to an incorrect landmark position given the lack of visual feedback. Note that in Figure 7, nominally there are only four beams (three horizontal and one vertical). Without the combinatorial updates from the beams, the robot is effectively lost after visiting for a second time the right-middle square.

VI. Conclusions

In this paper we present our results for studying robots that sense combinatorial information as switched

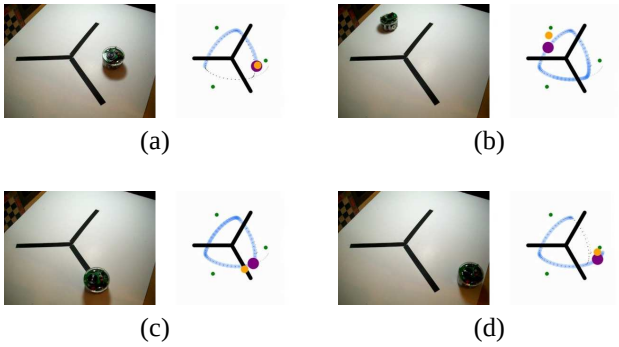

Fig. 6. Loops around the origin. On each subfigure, the orange (smaller and lighter) disc represents the estimate of the robot left. The dotted line represents the input trajectory, and the purple disc (bigger and darker) disc represents the desired instantaneous robot configuration. The position of the landmarks are represented by the green (darker) dots. In this experiment, the robot performs about a dozen loops around the common point of the beams, in which the observation of the state from the sensors consist only on which beam was crossed, but not at which point among the beam the crossing took place. Without the updates from the detection beams, the odometry estimate effectively looses the robot near the second loop.

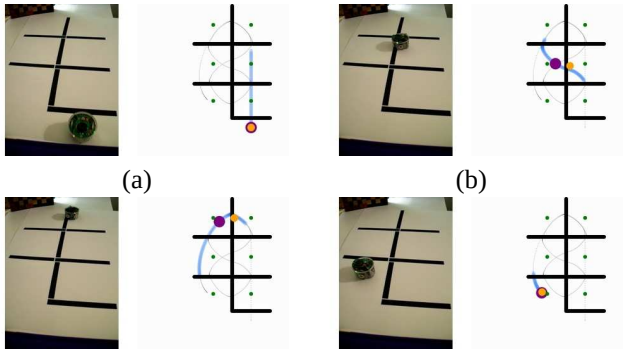

Fig. 7. Trajectory tracking. As in Figure 6, the orange (smaller and lighter) disc represents the estimate of the robot left and the purple disc (bigger and darker) disc represents the desired instantaneous robot configuration. Given that the desired trajectory crosses itself, we highlight configurations close to the desired instantaneous configuration with the blue thick curve. In this experiment, the robot is able to track the desired instantaneous configuration even when it is limited to move according to six motion primitives and can receive only four different observations from the sensors (there are only four distinct beams, three horizontal and one vertical). The complete run of this experiment is included in the accompanying video.

dynamical systems. We describe simple schemes that incorporate combinatorial information into dynamic state estimates. While our results are preliminary, we find them encouraging thus far. We are particularly interested in computing switching times as a function of combinatorial information. Up until now combinatorial information is only utilized to have a better estimate of the state; however, we would like this information to play a role in the sequence of motion primitives selected.

Perhaps the biggest issue with the current results is that the motion primitives are based on a linearization which assumes the robot to be pointing toward the landmarks. This linearization assumption is greatly violated when a switch produces an overly sharp turn. In the present paper, even though we took some provisions for this in Section V, our future work considers the computation of basins of attractions as present in [18]. This means that not all motion primitives are available all the time.

More interesting, as mentioned in Section V as future work, is the case when the camera is not omnidirectional. In this case, the motion primitives have to be designed to keep a particular landmark in sight, as in [8], [16], with the combinatorial events provided by the field-of-view, which can be modeled as three detection beams (the three segments of a the boundary of a circular section).

REFERENCES

[1] B.D. Anderson and J.B. Moore. *Optimal Control: Linear-Quadratic Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[2] K. E. Bekris, M. Glick, and L. E. Kavraki. Evaluation of algorithms for bearing-only slam. In *IEEE International Conference on Robotics and Automation (ICRA06)*, Orlando, FL, May 2006 2006.

[3] S.C. Bengea and R.A. DeCarlo. Optimal control for switching systems. *Automatica*, 41, 2005.

[4] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented matroids*. Cambridge University Press, 1993.

[5] M. Egerstedt, Y. Wardi, and F. Delmotte. Optimal control of switching times in hybrid systems. In *IEEE Conf. Decision & Control*, December 2003.

[6] M. Erdmann. Understanding action and sensing by designing action-based sensors. *Int. J. Robotic Research*, 14(5):483–509, 1995.

[7] E. Frazzoli, M.A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicle motion planning. In *IEEE Conf. on Decision and Control*, Sydney, Australia, 2000.

[8] J.B. Hayet, C. Esteves, and R. Murrieta-Cid. A motion planner for maintaining landmark visibility with a differential drive robot. In *Workshop on the Algorithmic Foundations of Robotics*, pages 333–347, 2008.

[9] S. Hedlund and A. Rantzer. Optimal control for hybrid systems. In *IEEE Conf. Decision & Control*, 1999.

[10] E. Johnson and T. Murphey. Second-order switching time optimization for non-linear time-varying dynamic systems. In *IEEE Conf. Decision & Control*, December 2003.

[11] J.J. Koenderink and A.J. van Doorn. The singularities of the visual mapping. *Biological Cybernetics*, 24:51–59, 1976.

[12] J.-P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. Robot. & Autom.*, 10(5):577–593, October 1994.

[13] A. Lazanas and J. C. Latombe. Landmark-based robot navigation. In *AAAI National Conference on Artificial Intelligence*, pages 816–822, 1992.

[14] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat J.-C., Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *9th Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65, 2009.

[15] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2000.

[16] P. Salaris, D. Fontanelli, L. Pallottino, and A. Bicchi. Shortest paths for a robot with nonholonomic and field-of-view constraints. *IEEE Trans. Robotics*, 26(2):269–281, 2010.

[17] L. Sonneborn and F. Van Vleck. The bang-bang principle for linear control systems. *SIAM J. Control*, 2:151–159, 1965.

[18] R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Robotics: Science and Systems*, 2009.

[19] B. Tovar. *Minimalist Models and Methods for Visibility-based Tasks*. PhD thesis, University of Illinois at Urbana-Champaign, August 2009.

[20] B. Tovar, F. Cohen, and S. M. LaValle. Sensor beams, obstacles, and possible paths. In *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2008.

[21] B. Tovar, L. Freda, and S. M. LaValle. Learning combinatorial map information from permutations of landmarks. *Accepted for publication in Int. J. Robotic Research*, 2010.

[22] B. Tovar, R Murrieta, and S.M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, June 2007.