Minimum-time trajectories for kinematic mobile robots and other planar rigid bodies with finite control sets

Andrei A. Furtuna, Wenyu Lu, Weifu Wang, and Devin J. Balkcom

***Abstract*— This paper presents first attempts at a method for searching for time-optimal trajectories for a general model of mobile robots that includes Dubins and Reeds-Shepp cars, differential-drive robots, and omnidirectional robots as special cases. The paper takes as a starting point recent results by the authors that describe necessary conditions on the trajectories, based on Pontryagin's Maximum Principle. These necessary conditions reduce the problem of finding an optimal trajectory between start and goal to a few one-dimensional search problems. This search is not formally guaranteed to find a near-optimal trajectory if the sampling of the search space is not fine enough, but comparison to existing analytical results for specific systems, and a complete numerical search over trajectories with only a few control switches, demonstrates effectiveness of the method.**

I. Introduction

There are different designs for simple mobile robots: steered cars, differential drives, and three-wheeled omnidirectional robots are common examples. This paper explores the minimum-time motion of a very simple model of these and other mobile robot designs. Figure 1 shows an example of the type of trajectory our search procedure finds for an omnidirectional robot, for the given start and goal. This trajectory is faster than any simple spin-translate-spin trajectory to the goal configuration, and we believe it to be the fastest possible for the available controls.

Consider any rigid body in the plane, with controls that can be described by rotation centers and translational directions rigidly attached to the body. Most simple mobile robots have attached controls, as do models of polygonal objects pushed in the plane without slip [16] by a robot arm.

In symbols, if the configuration is $q = (x, y, \theta)$, the controls u represent the generalized velocity in the frame of the robot, and $\mathcal{R}$ is the matrix that transforms the velocity into the world frame (formed by replacing the upper left block of a 3x3 identity matrix with a 2x2 rotation matrix), then the system equations are

$$q(T) = q(0) + \int_0^T \mathcal{R}(\theta(t))u(t)dt. \qquad (1)$$

The problem is, given a desired starting configuration $q(0)$ and goal location q_g, find a function $u(t)$ such that $q(T) = q_g$, and T is minimized. We assume that $u(t)$ is piecewise constant, with values chosen from some fixed finite set U.

We denote the elements of the set U by integer subscripts, $u_i = (\dot{x}_i, \dot{y}_i, \dot{\theta}_i)$. Trajectories with piecewise constant controls are represented by a sequence of (control index, time) pairs. The $[(i_1, t_1), (i_2, t_2), \ldots (i_n, t_n)]$ trajectory indicates the application of control u_{i_1} for time t_1, followed by control u_{i_2} for time t_2, *etc*.

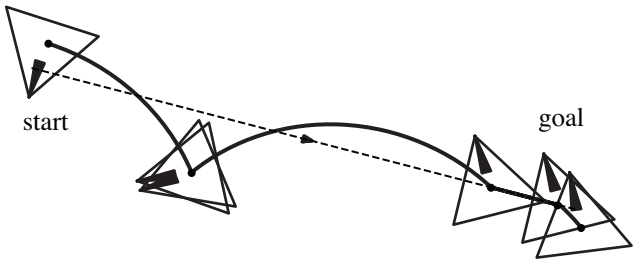

Fig. 1: A complicated but fast trajectory for an omnidirectional omni-wheeled robot. The robot rotates about a rotation center, spins in place, rotates again, translates, and finally rotates.

The kinematic model is a generalization of the steered cars studied by Dubins [10], Reeds and Sheep [18], of the differential-drive studied by Balkcom and Mason [4], and of an omnidirectional vehicle [3]. Different robot designs may be considered by choosing different control sets, as shown in the following table.

	Controls $(\dot{x}, \dot{y}, \dot{\theta})$, in robot frame
Dubins	(1, 0, 0), (1, 0, $\pm$1)
Reed-Sheep	($\pm$1, 0, 0), ($\pm$1, 0, $\pm$1)
Diff-Drive	($\pm$1, 0, 0), (0, 0, $\pm$1)
Omnidrive	$(\pm\sqrt{3}/3, \pm1, 0)$, $(\pm2\sqrt{3}/3, 0, 0)$ (0, 0, $\pm$1),(0, $\pm4/3, \mp1/3$) $(\pm2\sqrt{3}/3, 2/3, 1/3)$ $(\pm2\sqrt{3}/3, -2/3, -1/3)$

The original systems all allow controls chosen from a continuous set. However, we have shown in related work [11] that a finite set of constant controls and piecewise constant control laws are sufficient for optimality, as long as the bounds on the generalized velocities are polyhedral – a condition that all of these previous models satisfy. The same work also addresses the issue of the existence of optimal trajectories.

There are many reasonable objections to the model we study. Real robots must contend with obstacles, and cannot instantaneously switch velocities. However, just as kinematic models are useful and necessary for understanding the motion of arms, this kinematic model is a good starting place to understand the fundamental behavior of simple mobile robots. The fastest curves, or robot geodesics, may be motion primitives for planning systems, a way to compare competing robot designs, or allow creation of a metric or measure that can be used to sample trajectory and configuration spaces more uniformly.

Our primary approach to finding the fastest trajectories between a pair of points for a given set of discrete controls is an *indirect* method. Rather than searching directly over the possibly infinite-dimensional space of all trajectories connecting a particular start and goal, we apply results derived from Pontryagin's Maximum Principle [17] by our recent journal paper [12], giving strong necessary conditions that optimal trajectories must satisfy. Trajectories satisfying these conditions are called *extremal* trajectories. These extremal trajectories form a number (quadratic in the number of available controls) of one-parameter families of curves. We search over each family and sample the single parameter to find the fastest trajectories that connect a given start and goal, within some error tolerance.

There is a difficulty with this approach – although every optimal trajectory must be extremal, we cannot guarantee that sampling the single continuous parameter at some fixed resolution finds the correct parameter. For this reason, as a basis for comparison, we also derive a direct approach for searching for optimal trajectories. Although the direct approach is computationally infeasible for trajectories with more than a few (five or six) control switches, it is reassuring that the very-different direct and indirect approaches appear to agree well in the cases where the indirect method provides a trajectory with few switches, and more-complicated trajectories found by the indirect method are reliably faster.

A. Previous work

In 1957, Dubins characterized the shortest paths for what is essentially a simple model of a car with bounded steering angle and fixed velocity [10]. In 1990, Reeds and Shepp characterized the trajectories for a car that can also reverse [18]. Sussman and Tang described a general methodology for solving problems of this type [24], and Souères, Boissonnat, and Laumond [22], [23] discovered the mapping from pairs of configurations to optimal trajectories for the Reeds-Shepp car. The approaches developed enabled our discovery of the time-optimal trajectories for two other simple models of vehicles: the differential drive [4], and a particular three-wheeled robot that can drive sideways as well as forwards [3].

Our recent journal paper [12] presents theoretical results that attempt to generalize these results across robot designs, using the kinematic rigid-body model described. The current paper represents our first attempt at using these results to build algorithms that search for optimal trajectories efficiently for the general system.

The problem of finding optimal trajectories has been studied for many other specific models of vehicles, ranging from work by Chitsaz [6] on differential drives, to work by Coombs and Lewis [8] on a simplified model of a hovercraft, to Chyba and Haberkorn's [7] work on underwater vehicles. Papers by Reister and Pin [19], and Renaud and Fourquet [20] present numerical and partial geometric results for dynamic steered cars, and Kalmár-Nagy *et al.* [14] present algorithms for numerical approximations of optimal trajectories for dynamic omnidirectional robots. The interaction of optimal trajectories of particular robot types with various classes of obstacles has also been an active research area [9], [25], [1], [21], [13].

Dubins and Reeds-Shepp curves arise in many robotics problems. For example, the optimality of Reeds-Shepp curves is the motivation for Barraquand and Latombe's choice of discrete controls for their motion planner for a non-holonomic cart [5]. Recent work by Alterovitz *et al* [2] models the motion of a surgical needle through the body as a Dubins curve. LaValle's [15] work on rapidly-exploring random trees relies on a metric (or pseudometric) between configurations in the free space, and Reeds-Shepp curves can be used to generate a metric for steered cars.

II. Forward kinematics of trajectories

The transformation matrix that describes the frame of the robot with respect to the world frame is

$$T_{WR} = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Since this transformation matrix encodes the same information as $q = (x, y, \theta)$, we may use it as an alternate means of indicating the vehicle's position. Omitting the first frame subscript indicates the matrix given with respect to the world frame, *e.g.* T_0 is the robot's initial state in the world frame.

Applying a constant control causes the body to either rotate or translate. We therefore expect to be able to describe the motion of the robot by multiplying homogeneous frame transforms corresponding to each rotation and translation in order. If T_n is the configuration of the robot after n control actions, the configuration may be computed by matrix composition:

$$T_n = T_0 T_{01} T_{12} T_{23} T_{34} \ldots T_{n-1,n} \tag{3}$$

For example, T_{23} would express the frame of the robot after applying the third control action in a trajectory, with respect to the frame of the robot after applying the second control action. This notation is exactly the same as that commonly used for the kinematics of a robot arm.

The transform matrices correspond to either translations or rotations, but it is inconvenient in algorithms, code, and analysis to have to have different formats of the matrices depending on whether $\dot{\theta} = 0$ or not. There are also problems of numerical instability. What if $\dot{\theta}$ is very close to zero? This situation may arise if one wheel of a differential-drive is slightly larger than the other, or if the motors are mismatched. In this case, the rotation center is nearly at infinity, and the angle rotated through is near zero.

In this section, we suggest a trick that unifies rotation and translation matrices in the plane, by making use of well-behaved smooth functions. Compositions of these matrices describe the motion of the robot in a simple robust way.

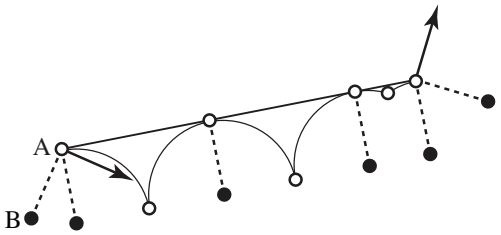

Fig. 2: Output of the universal planner for a robot with just two rotation controls centers (black and white circles). Start and goal configurations are given by arrows; the path of the white rotation center is shown.

A. Unified translation-rotation matrices

The well-known cardinal sine function is defined as:

$$\operatorname{sinc}(x) = \begin{cases} \frac{\sin x}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases}. \tag{4}$$

By analogy with the cardinal sine, and a function known as the versine, we define a similar function, the *cardinal versine*:

$$\operatorname{verc}(x) = \begin{cases} \frac{1-\cos x}{x}, & x \neq 0 \\ 0, & x = 0. \end{cases} \tag{5}$$

Using these functions, the unified translation-rotation matrix describing motion using constant control with index i for duration t is given by

$$T(i,t) = \begin{bmatrix} \cos\dot{\theta}_i t & -\sin\dot{\theta}_i t & \dot{x}_i t \operatorname{sinc}\dot{\theta}_i t - \dot{y}_i t \operatorname{verc}\dot{\theta}_i t \\ \sin\dot{\theta}_i t & \cos\dot{\theta}_i t & \dot{x}_i t \operatorname{verc}\dot{\theta}_i t + \dot{y}_i t \operatorname{sinc}\dot{\theta}_i t \\ 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

Given a sequence of control indices in a trajectory $i_1, \ldots i_j, \ldots i_n$, the transform matrix $T_{j-1,j}$ from equation 3 is given by

$$T_{j-1,j} = T(i_j, t_j). \tag{7}$$

The state at time t is:

$$T(t) = T_0 T(i_1, t_1) T(i_2, t_2) \ldots T(i_k, t_k) T(i_{k+1}, t'), \tag{8}$$

where k is the largest index such that $t' = t - \sum_{j=0}^{k} t_j \geq 0$.

III. Global controllability

In this section we sketch a simple and fast technique for always finding some trajectory to the goal; this will provide an upper bound on trajectory time that will be used in search algorithms later in the paper. A system is *controllable* over some space if a trajectory exists between any two states in that space. The basic geometry of this technique is described in the sketch of the proof of the following lemma.

Lemma 1: A rigid body, controlled by velocities chosen from a set U that is constant in the body's own frame of reference, is controllable in SE(2) if and only if U contains two or more distinct velocities, at least one of which is a rotation.

If one of the velocities is a translation, then the trajectory is easy to construct: choose the reference point of the robot to be centered on the rotation center, spin in place until the translation direction is lined up with the vector from the start to the goal, drive to the goal, and rotate to the required angle.

If both velocities are rotations, we replace the translation section of the trajectory with a sequence of rotations. Let A and B represent rotation centers (see fig. 2); choose the origin of the robot frame coincident with A. Let the distance between the two rotation centers be l. Rotate B around A until the line segment from A to B is perpendicular to the line from the start to the goal. Then repeat a series of segments, where each segment is of the form $B_{90^\circ} A_{180^\circ} B_{90^\circ}$, achieving a pure translation of distance $2l$ in the direction of the line segment from start to goal. If such a translation would overshoot the goal, adjust the angles in the BAB segment to exactly reach the goal (details left to reader); finally rotate about A to the goal angle.

By considering all possible rotation-rotation and rotation-translation pairs of controls and picking the fastest of the trajectories thus generated, this method can be used to quickly construct trajectories that are often not much slower than the optimal.

IV. Direct optimization using inverse kinematics

We now turn to the direct search method. Assume we know the sequence of controls that will be applied in a trajectory. Choosing the times of each segment $(t_1, t_2, \ldots t_n)$ changes the configuration that will be reached. On the other hand, if we are given the goal configuration q_g, computing the times is essentially an inverse kinematics problem.

If there are three segments in the trajectory, then we expect there to be at most two solutions for the segment times. If there are more switches, then the trajectory has extra degrees of freedom, and there may be an infinity of trajectories that reach the goal. This suggests a parametrization of the space of trajectories that reach the goal, similar to a parametrization often used for planar closed-chains. For an n-segment trajectory, where $n > 3$, use the first $n-3$ durations $t_1, t_2, \ldots, t_{n-3}$ as free parameters to reach some configuration. Then use inverse kinematics to compute the (up to two) possible trajectories that reach the goal from this intermediate configuration.

We use this parametrization to build a direct search method for optimal trajectories. We first decide on some number of trajectory segments – using more than five is typically computationally infeasible. Then iterate over all possible trajectory structures with this number of segments. For each trajectory structure, sample the $(n-3)$-dimensional space of trajectories that reach the goal, to find the minimum-time trajectory. This direct approach, though computationally expensive and limited in the complexity of trajectories that can be computed, is quite different from the indirect method described in the next sections, and forms a good basis for comparison. This method is also the simplest method for computing trajectories for systems with few switches in the optimal trajectories – for example, as pointed out by LaValle [15], the inverse kinematics method is probably the simplest way to implement Dubins curves.

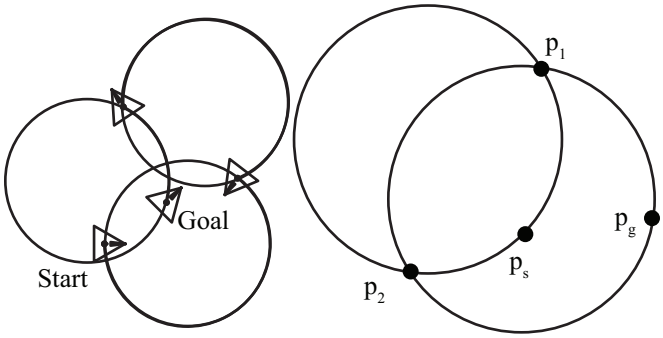

Fig. 3: CCC trajectory shown from two points of reference.

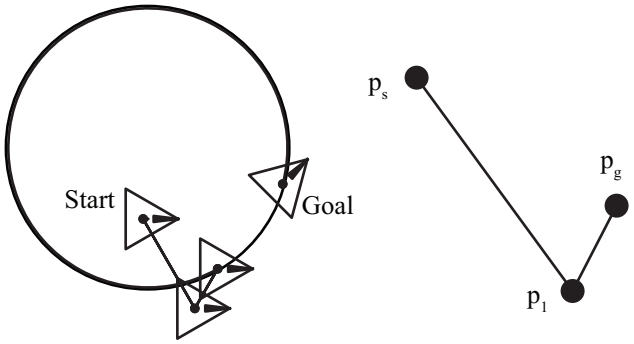

Fig. 4: LLC trajectory shown from two points of reference.

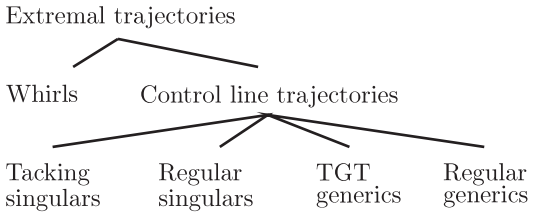

Fig. 5: Types of trajectories satisfying the Pontryagin Principle

As a first step, we must compute the three-segment optimal trajectory for a given set of controls using inverse kinematics. We denote a rotational action as 'C' and a translational action as 'L'. There are eight cases for a three-segment trajectory: CCC, CLC, CLL, LLC, LCL, LCC, CCL and LLL. Define the start configuration as q_s and goal configuration as q_g, and also define the start rotation matrix as R_s and the goal rotation matrix R_g; s and g are the coordinates of the start position and the goal position respectively. Without loss of generality, re-index the controls so that the first three controls in U are the controls for the trajectory structure currently being considered; for example, $u_1 = (\dot{x}_1, \dot{y}_1, \dot{\theta}_1)$ will be the first control applied, as well as the first control in the set U.

If the i^{th} control is a rotation, the rotation center in the car's local frame is:

$$c_i = \begin{bmatrix} -\dot{y}_i/\dot{\theta}_i \\ \dot{x}_i/\dot{\theta}_i \end{bmatrix} \tag{9}$$

A. CCC

Choose the second rotation center as the reference point; the trajectory structure is simplified to two circular arcs (figure 3). p_s is the start position of the chosen reference point and p_g is the goal position of the chosen reference point. The centers of the circles c_A and c_B are the locations of the first rotation center in the initial state, and the third rotation center in the final state, respectively. The radii are $r_A = ||c_1 - c_2||$, $r_B = ||c_2 - c_3||$. Compute the intersections of the circles, p_1 and p_2. For each intersection, compute the corresponding trajectory, and choose the faster. For p_1,

$$\cos(\Delta\theta_1) = \frac{(p_1 - c_A)^T(p_s - c_A)}{||(p_1 - c_A)||||(p_s - c_A)||} \tag{10}$$

$$t_1 = \Delta\theta_1/\dot{\theta}_1 \tag{11}$$

$$\cos(\Delta\theta_3) = \frac{(p_1 - c_B)^T(p_g - c_B)}{||(p_1 - c_B)||||(p_g - c_B)||} \tag{12}$$

$$t_3 = \Delta\theta_3/\dot{\theta}_3 \tag{13}$$

$$T(2, t_2) = (T_0 T(1, t_1))^{-1}(T(3, t_3)^{-1} T_g). \tag{14}$$

$\Delta\theta_2$ and t_2 may be computed using a two-argument arctangent of the first two elements in the first column of $T(2, t_2)$. Computation for the second intersection p_2 is analogous.

B. LLC,CLL and LCL

Choose the rotation center as the reference point; the simplified trajectory is composed of two line segments without any circular arcs. Figure 4 shows an example of an LLC trajectory, for which the solution is:

$$\begin{bmatrix} \dot{x}_1 & \dot{x}_2 \\ \dot{y}_1 & \dot{y}_2 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = p_g - p_s \tag{15}$$

$$T(3, t_3) = (T_0 T(1, t_1) T(2, t_2))^{-1} T_g \tag{16}$$

$\Delta\theta_3$ (and thus t_3) may be computed using a two-argument arctangent of the first two elements in the first column of $T(3, t_3)$.

C. LCC, CCL and CLC

For LCC and CCL, choose the second rotation center as the reference point and get a simplified trajectory which is composed of a line segment (the velocity vector in world frame) and a circular arc. The intersections of the line and the circle are possible switches. The solution for the CLC case has been discussed in section III.

D. LLL

LLL trajectories can only be applied when the start and goal configurations have the same orientation. Furthermore, although we omit the details, an optimal trajectory can always be constructed that uses at most two translation controls. This case therefore is a special case of an LLC trajectory with a final rotation of zero.

V. The Indirect Method: The Pontryagin Principle

In this section, we will describe the fundamentals of the indirect method for finding the fastest trajectories. The exposition is necessarily very brief, although it does address all

the main issues. For an easier and more detailed introduction, see [11].

The indirect method is based on Pontryagin's Maximum Principle, which places strong necessary conditions upon the structure of optimal trajectories. We call the class of trajectories that satisfy these necessary conditions extremal trajectories.

A. Types of extremal trajectories

As shown in [12], extremal trajectories can be divided into the following categories (see figure 5):

1) Generics: entirely determined by the position of the control line.
2) Singulars: may contain translations parallel to the control line. The duration of these translations is not constrained by the Maximum Principle.
3) Tacking trajectories: may contain translation-translation switches. The timing of translation-translation switches is not constrained by the Maximum Principle.
4) Whirls: only contain rotations that have the same angular velocity (either the maximum or the minimum angular velocity that the moving body is capable of). The switches on these trajectories are not constrained by the Maximum Principle.

Whirls have been completely solved in [12]. Tacking trajectories cannot occur for any existing vehicles that we are aware of, and will not be studied in this work. We concentrate upon generics and singulars.

B. The Maximum Principle

We will give a geometric interpretation of the Maximum Principle as applied to our problem. Let the control line be an arbitrary line in the world frame with heading (k_1, k_2) and signed distance k_3 from the origin. Then the frame of reference attached to the control line is

$$T_{LW} = \begin{bmatrix} k_1 & k_2 & 0 \\ -k_2 & k_1 & k_3 \\ 0 & 0 & 1 \end{bmatrix}. \tag{17}$$

Given a state T, define the Hamiltonian of control u_i as

$$H_i = (0, 1, 0) T_{LW} T c_i \tag{18}$$

where $c_i = (-\dot{y}_i, \dot{x}_i, \dot{\theta}_i)$ is the rotation center representation of control u_i. Restating theorem 1 from [12] with this notation, we obtain

Theorem 1: Consider a rigid body that is controlled by choosing velocities from a finite set U that is constant in the body's frame. Let $[(i_1, t_1), (i_2, t_2), \ldots (i_n, t_n)]$ be a time-optimal trajectory in the plane for this rigid body from initial state T_0 to final state T_f, such that not all $\dot{\theta}_i$ are the same. Then it is necessary that:

1) There exist, for this trajectory: a control line, fixed in the world frame, and a real constant λ_0.
2) At all times t along the trajectory, if u_k is the active control, then for any other control i

$$H_i(t) \leq H_k(t) = \lambda_0. \tag{19}$$

Control index i is a *sustainable control* at state T if H_i is maximal, and furthermore i does not cause any other control to overshoot λ_0. All extremals have at least one sustainable control at each time. Generics have exactly one sustainable control at each time. Singulars have at least one time when there are two sustainable controls.

VI. Generating extremal trajectories

Except in cases that can be well characterized, fixing the position of the control line exactly determines the trajectory structure and the time between switches. This section is concerned with describing this influence, and with showing how to build an extremal trajectory generator, i.e. an algorithm that generates extremal trajectory segments corresponding to a fixed position of the control line and some start configuration.

A. Control switches on extremal trajectories

Consider controls i, j at state T. Assume i is the currently maximizing control, and therefore active. How long will it take until we may switch to control j?

The Hamiltonians for the two controls are:

$$H_i = (0, 1, 0) T_{LW} T c_i \tag{20}$$

$$H_j = (0, 1, 0) T_{LW} T c_j \tag{21}$$

Subtracting:

$$H_i - H_j = (0, 1, 0) T_{LW} T (c_j - c_i) \tag{22}$$

We call the homogeneous point $s_{ij} = c_j - c_i$ a *switching point*: when control switches from i to j, s_{ij} is on the control line. In order to determine switching time, we need to solve the following equation for t:

$$0 = (0, 1, 0) T_{LW} T T(i, t) s_{ij} \tag{23}$$

If i is a translation, the solution is straightforward. For rotation, the time is calculated computing the intersection of a circle with a line.

B. Generic simulator

The generic simulator takes as inputs a control set U and an initial state T_0. It outputs an extremal trajectory. Depending on the termination condition, the generic simulator can run in several modes, e.g.

1) n switches: terminate after n control switches have occurred.
2) Generic period: terminate when a switch is encountered the second time. (All generic trajectories become periodic, if long enough [12].
3) Generic excursion: terminate when a switching state with more than one sustainable control is encountered. Generic excursions appear within singular trajectories.

The algorithm for the generic simulator is as follows:

1) Check termination condition.
2) Let i be the sustainable control at T.
3) Compute all switching times to all other controls in U.
4) Let t_{min} be the minimum switching time. Add (i, t_{min}) to the output trajectory.
5) $T = TT(i, t_{min})$. Go to 1).

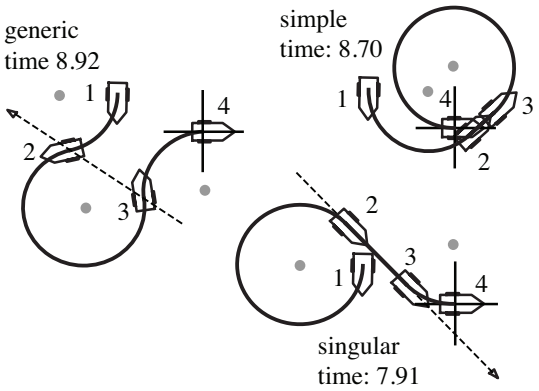

Fig. 6: Comparison of 'generic' and 'singular' (fastest) trajectories to the output of the simple universal planner, for Dubins cars with the same starting configuration and goal.

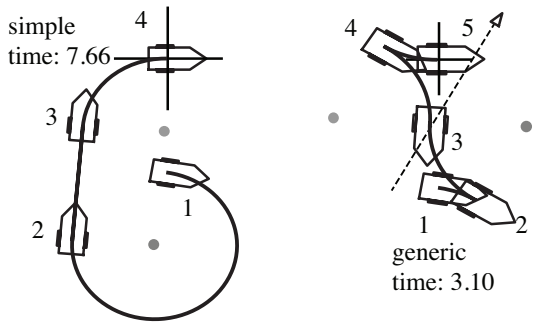

Fig. 7: Comparison of 'generic' and 'simple' trajectories for the Reeds-Shepp car.

VII. Finding all extremal trajectories

The extremal trajectory generator constructs a trajectory for a given position of the control line. In order to find the position of the control line corresponding to a given start and goal, we vary the position of the control line until the goal is attained with a sufficiently small error. First, we show how to reduce the problem of finding a suitable control line to one-dimensional search.

A. One-parameter search for the control line

The position of the control line is determined by three parameters: k_1, k_2 and k_3. Without loss of generality, we can restrict k_1 and k_2 such that $k_1^2 + k_2^2 = 1$. Further assume the initial and final controls, i_0 and i_g, are given. Then

$$0 = H_{i_g} - H_{i_0} = (0, 1, 0) T_{LW} (T_g c_{i_g} - T_0 c_{i_0}) \qquad (24)$$

Since $T_g c_{i_g} - T_0 c_{i_0}$ is a known quantity, this equation gives us a restriction on k_1, k_2, k_3. (This restriction is equivalent to knowing one point, possibly at infinity, on the control line.)

We can safely ignore the case when $T_g c_{i_g} - T_0 c_{i_0} = 0$. In this case, the initial and final control have the same center of rotation in the world frame (or are parallel translations). Since the controls need to have the same H, the angular velocities (or planar velocities, for translations) are also the same. Therefore, for any such trajectory, we can shift small amounts of movement from the initial to the final control, and generate an infinite number of trajectories that take the same time. At least one of these trajectories (zero time on the initial control) will be found by other means.

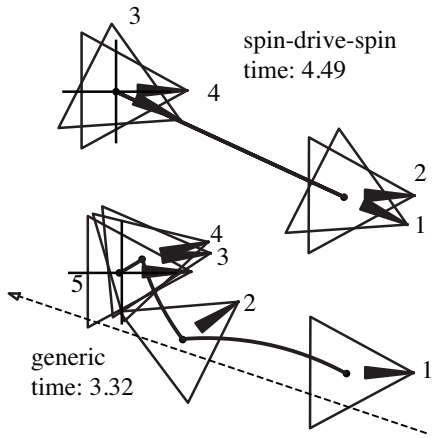

Fig. 8: Comparison between a spin-drive-spin trajectory and a 'generic' extremal for the omnidrive robot.

B. Searching for generic trajectories

Searching for generic trajectories involves two main issues: finding a convenient parameter for the one-dimensional search, and cutting the trajectory corresponding to a control line, in order to obtain a final state that we can compare with the goal state in terms of an error measuring metric.

1) TGT trajectories: If $\dot\theta_0 = \dot\theta_f = 0$ (i.e., both the initial and the final controls are translations) and furthermore (x_0, y_0) is not a multiple of (x_f, y_f) (which is ruled out by our earlier assumption that the initial and final controls are different motions in the world frame), then the corresponding trajectory begins and ends with non-parallel translations. In this case, the direction of the control line can be calculated, and consequently the length and shape of the subtrajectory between the initial and final translations are calculated exactly by placing the control line arbitrarily. This length further determines the exact position of the control line.

2) Parametrization of generic search: In cases where the initial and final motions are not translations, we use λ_0 as a parameter. A λ_0 parametrization is convenient for search, since the trajectory structure only changes at certain critical values of λ_0 (see [12], Theorem 3). We describe a procedure to obtain a function $f(\lambda_0)$ that returns a maximum of two (k_1, k_2, k_3) tuples characterizing the position of the control line.

Let $k_1 = \cos\varphi$ and $k_2 = \sin\varphi$ and let $c_{W0} = (x_0, y_0, \dot\theta_0)$ and $c_{Wg} = (x_g, y_g, \dot\theta_g)$ be the rotation centers, in the world frame, corresponding to the first control in the initial state, and the last control in the goal state, respectively. Since the Hamiltonians of the initial and final controls must be equal, we obtain the system

$$(-\sin\varphi, \cos\varphi, k_3)(x_0, y_0, \dot\theta_0)^T = \lambda_0 \qquad (25)$$

$$(-\sin\varphi, \cos\varphi, k_3)(x_g, y_g, \dot\theta_g)^T = \lambda_0 \qquad (26)$$

If $\dot\theta_0$ and $\dot\theta_f$ are not both zero, this system quickly yields two

values of (φ, k_3) as a function of λ_0, which is the sought-after parametrization.

3) Cutting a generic trajectory: The generic trajectory yielded by a known control line is of unbounded length, as the control law by itself does not specify a stopping point. In this section, we show how to create a finite trajectory, ending in a point that is closest to the goal by a certain metric. This metric is also a measure of how good the control line is, and is thus used to guide the search.

For any trajectory, the error in attaining the goal is a three-dimensional quantity in $SE(2)$. The previous section has shown how to restrict the position of the control line such that the y_L coordinate of the last rotation center is always attained exactly. We resolve the θ error by allowing the class of acceptable trajectories that are returned by the search to be slightly expanded. The last rotation movement is always applied until the goal θ coordinate is attained, even if this would make the trajectory non-extremal on that rotation segment. All extremal trajectories that reach the goal evidently belong to this slightly enlarged class.

These steps leave only one dimension in the error: the x_L coordinate. Measuring the x_L error gives us a metric for how close the control line is to attaining the goal. The x_L error is measured in the position of the last rotation center that is used on the trajectory. With each generated period, we keep track of the x_L coordinate of this rotation center when it is active. This quantity will keep increasing until the goal x_L is surpassed; at this point, the generator is stopped, and the attained x_L that is closest to the goal is used for both computing the metric and cutting the trajectory.

In the case in which the last control is a translation, we run the generator backwards for one generic segment. Since the H value is not critical, the previous control must have been a rotation. We use this rotation as the last control and the switch state as the target state, and we apply the method above for cutting the trajectory and generating the metric.

C. Searching for singulars

Singulars can only occur at critical values of the Hamiltonian ([12]). In conjunction with the known parameters of the control line, each critical value yields at most two positions for the control line. We assume for the rest of this section that the position of the control line is given.

Usually, most points on singular trajectories are generic. A continous segment of a singular trajectory is entirely composed of singular points only if the segment is a translation parallel to the control line. Sections of singular trajectories that are entirely composed of generic points are called *generic excursions*. Generic excursions fall into three categories:

1) type A: ends with singular point.
2) type B: starts with singular point.
3) type C (complete): starts and ends with singular points.

Singulars are composed of generic excursions and translations parallel to the control line. To find generic excursions, we use the generator with the appropriate termination condition, as follows.

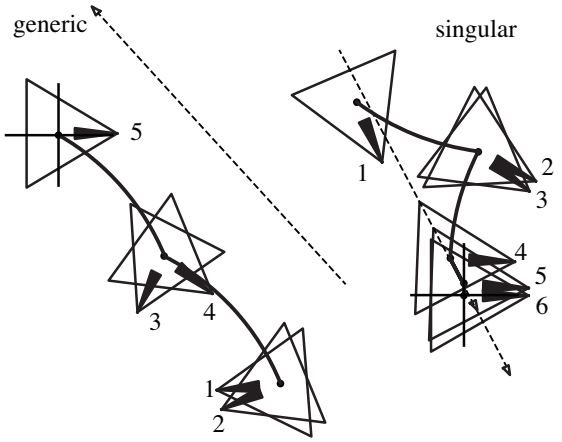

Fig. 9: Configurations of the omnidrive robot between which a 'generic' trajectory is fastest (left), and configurations between which a 'singular' trajectory is fastest (right).

Given T_0, T_g, we run the generator from T_0 to generate a type A excursion to state T_a, and run the generator in reverse from T_g to generate a type B excursion starting from T_b. T_a and T_b are now to be connected with type C generics and with translations parallel to the control line. There are usually multiple ways to effect such a connection.

The overall algorithm for finding all singulars takes as input a control set U, states T_a and T_b generated as above, and a maximum cut-off time t_{max} (found by the universal planner). The output is a list of all singulars connecting T_a to T_b.

1) If T_a and T_b have same y and θ coordinates and there exists a translation parallel to the control line at T_a, add this translation to result.
2) For each sustainable control j at T_a:
   a) Generate the type C generic excursion starting with j; let T_a' be the final state of this generic excursion and t' be its time.
   b) If $t' < t_{max}$, call current algorithm recursively with T_a', T_b, $t_{max} - t'$
   c) To each singular returned, prepend the type C generic excursion found in a), and add it to the output list.

VIII. Tests and discussion

We implemented the indirect and direct search methods described in C. The table below shows the total running time for the indirect method to find trajectories from 1000 starting conditions to the origin, for various vehicle models.

For the Dubins and Reeds-Shepp cars and the diff-drive, we generated q_0 from the range $[-3, 3] \times [-3, 3] \times [-\pi, \pi]$; for omni-directional vehicle we generated samples from the range $q_0 \in [-7, 7] \times [-7, 7] \times [-\pi, \pi]$.

model	controls	cases	total time
Dubins	3	1000	70s
Reed-Sheep	6	1000	287s
Diff-Drive	4	1000	90s
Omnidrive	8	1000	710s

With the exception of the Reeds-Shepp car (for which whirl trajectories may be optimal), for each starting configuration, either the generic or the singular was at least as fast as any trajectory found by the direct method.

model	generic optimal	singular optimal
Dubins	25.2%	74.8%
Reed-Sheep	21.7%	67.3%
Diff-Drive	64.5%	35.5%
Omnidrive	47.7%	52.3%

IX. Conclusion

Some aspects of our first attempts at searching for extremal and optimal trajectories are certainly naive. For steered cars and differential-drives, our approach computes trajectories more slowly and with less precision than by the exact procedures previously known, since we do not take advantage of special knowledge about the control set. However, the approach is general for all of the vehicle types, and we are also able to compute trajectories between configurations for the omnidirectional vehicle for the first time. Simply changing the input to the planner allows rapid study of any new robot design that fits the model. A particularly interesting set of robot designs inspired by this work is omnidirectional robots with different configurations of the driving omniwheels, allowing tradeoffs between speeds and torques the robot can achieve in different directions in the configuration space. From a practical perspective, the trajectories found by the planner are fast, reach the goal with very good precision, and many are interestingly-complicated enough that they have not previously been found by any other means.

References

[1] Pankaj K. Agarwal, Therese Biedl, Sylvain Lazard, Steve Robbins, Subhash Suri, and Sue Whitesides. Curvature-constrained shortest paths in a convex polygon. In *SIAM J. Comput*, pages 392–401. ACM Press, 1998.

[2] Ron Alterovitz, Michael Branicky, and Ken Goldberg. Motion planning under uncertainty for image-guided medical needle steering. *International Journal of Robotics Research*, 27(11-12):1361–1374, 2008.

[3] Devin J. Balkcom, Paritosh A. Kavathekar, and Matthew T. Mason. The minimum-time trajectories for an omni-directional vehicle. In *Workshop on the Algorithmic Foundations of Robotics*, 2006.

[4] Devin J. Balkcom and Matthew T. Mason. Time optimal trajectories for differential drive vehicles. *International Journal of Robotics Research*, 21(3):199–217, 2002.

[5] Jérôme Barraquand and Jean-Claude Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[6] Hamidreza Chitsaz, Steven M. LaValle, Devin J. Balkcom, and Matthew T. Mason. Minimum wheel-rotation paths for differential-drive mobile robots. In *IEEE International Conference on Robotics and Automation*, 2006.

[7] M. Chyba and T. Haberkorn. Designing efficient trajectories for underwater vehicles using geometric control theory. In *24rd International Conference on Offshore Mechanics and Artic Engineering*, Halkidiki, Greece, 2005.

[8] A. Theo Coombs and Andrew D. Lewis. Optimal control for a simplified hovercraft model. Preprint.

[9] Guy Desaulniers. On shortest paths for a car-like robot maneuvering around obstacles. *Robotics and Autonomous Systems*, 17:139–148, 1996.

[10] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[11] A. A. Furtuna. *Minimum time kinematic trajectories for self-propelled rigid bodies in the unobstructed plane*. PhD thesis, Dartmouth College, 2011.

[12] Andrei A. Furtuna and Devin J. Balkcom. Generalizing Dubins curves: minimum-time sequences of body-fixed rotations and translations in the plane. *International Journal of Robotics Research*, 29:703–726, 2010.

[13] Jean-Bernard Hayet, Claudia Esteves, and Rafael Murrieta-Cid. A motion planner for maintaining landmark visibility with a differential drive robot. In *Workshop on the Algorithmic Foundations of Robotics*, December 2008.

[14] Tamás Kalmár-Nagy, Raffaello D'Andrea, and Pritam Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46:47–64, 2004.

[15] Steven M. LaValle. *Planning Algorithms*. Cambridge Press, 2006. Also freely available online at `http://planning.cs.uiuc.edu`.

[16] Kevin M. Lynch and Matthew T. Mason. Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research*, 15(6):533–556, December 1996.

[17] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. John Wiley, 1962.

[18] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[19] David B. Reister and Francois G. Pin. Time-optimal trajectories for mobile robots with two independently driven wheels. *International Journal of Robotics Research*, 13(1):38–54, February 1994.

[20] Marc Renaud and Jean-Yves Fourquet. Minimum time motion of a mobile robot with two independent acceleration-driven wheels. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 2608–2613, 1997.

[21] Paolo Salaris, Felipe A. W. Belo, Daniele Fontanelli, Luca Greco, and Antonio Bicchi. Optimal paths in a constarained image plane for purely image-based parking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1673–1680, 2008.

[22] P. Souères and J.-D. Boissonnat. Optimal trajectories for nonholonomic mobile robots. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 93–170. Springer, 1998.

[23] Philippe Souères and Jean-Paul Laumond. Shortest paths synthesis for a car-like robot. *IEEE Transactions on Automatic Control*, 41(5):672–688, May 1996.

[24] Héctor Sussmann and Guoqing Tang. Shortest paths for the Reeds-Shepp car: a worked out example of the use of geometric techniques in nonlinear optimal control. SYCON 91-10, Department of Mathematics, Rutgers University, New Brunswick, NJ 08903, 1991.

[25] M. Vendittelli, J.P. Laumond, and C. Nissoux. Obstacle distance for car-like robots. *IEEE Transactions on Robotics and Automation*, 15(4):678–691, 1999.