Pergamon

PII: S0005-1098(97)00024-1

Automatica, Vol. 33, No. 7, pp. 1219–1233, 1997

Printed in Great Britain
0005-1098/97 $17.00 + 0.00

The Jogger's Problem: Control of Dynamics in Real-time Motion Planning*

ANDREI M. SHKEL† and VLADIMIR J. LUMELSKY†

An approach that incorporates the effects of body dynamics into the paradigm of sensor-based motion planning is proposed. Given the constraints on the robot's dynamics, sensing and control means, conditions are formulated for generating collision-free trajectories with guaranteed convergence.

Key Words—Robotics; real-time motion planning; dynamic behavior; intelligent machines.

Abstract—Existing approaches to sensor-based motion planning tend to deal solely with kinematic and geometric issues, and ignore the system dynamics. This work attempts to incorporate body dynamics into the paradigm of sensor-based motion planning. We consider the case of a point-mass mobile robot operating in a planar environment with unknown stationary obstacles of arbitrary shapes. Given the constraints on the robot's dynamics, sensing and control means, conditions are formulated for generating collision-free trajectories with guaranteed convergence. The approach calls for continuous computation, and is fast enough for real-time implementation. Based on its velocity and sensing data, the robot continuously plans its motion based on canonical solutions, each of which presents a time-optimal path within the robot's sensing range. For a special case of a sudden potential collision, an option of a safe emergency stopping path is always maintained. Simulated examples demonstrate the algorithm's performance. © 1997 Elsevier Science Ltd.

1. INTRODUCTION

This paper studies the effects of body dynamics on robot sensor-based motion planning, with the goal of designing exact (provably correct) algorithms for motion planning in an uncertain environment. We consider a mobile point robot operating in two-dimensional physical space filled with unknown stationary obstacles of arbitrary shapes. Planning is done in small steps (say, 30 or 50 times per second), resulting in a continuous motion. The robot is equipped with sensors, such as vision or rangefinders, which allow it to detect and measure distances to surrounding objects within its sensing range ('radius of vision'). This range covers a number of robot steps—say, 20 or 50 or more—so that, unless obstacles occlude one another, they can be sensed far enough to plan appropriate actions.

Besides the usual issues of step-by-step planning and convergence in view of incomplete information, an additional dynamic component of planning appears because of the effect of the robot's mass and velocity. A move reasonable from the standpoint of the global path—for example a sharp turn—may not be physically realizable because of the robot's inertia. The control and planning issues that appear in this model seem quite similar to those faced by a jogger in the city environment—hence the name the *Jogger's Problem.*

Interestingly, while the issues of dynamics and sensor-based motion control are tightly coupled, little attention has been paid to this connection in the literature. The existing approaches usually deal solely with the system kinematics and geometry, and ignore its dynamic properties. Consequently, only applications where the effect of speeds and masses is neglibible can benefit from these techniques. Perhaps one reason for this state of affairs is that the methods considered tend to depend strongly on tools from geometry and combinatorics, which are not easily connected to the tools common to control theory.

Most approaches to automatic motion planning adhere to one of two paradigms, which use different assumptions about the input information available. In the first paradigm, called

* Received 25 January 1995; revised 4 June 1996; received in final form 13 December 1996. This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor P. J. Gawthrop under the direction of Editor C. C. Hang.
Corresponding author Dr Andrei M. Shkel. Tel. +1 608 265 4010; Fax +1 608 265 2316; E-mail shkel@robios.me.wisc.edu.
† Mechanical Engineering Department, University of Wisconsin–Madison, Madison, WI 53706, USA.

motion planning with complete information (or the *Piano Mover's Problem*), one assumes perfect information about the robot and obstacles, and algebraic representation of objects; motion planning is a one-time off-line operation (see e.g. Schwartz and Sharir, 1983; Canny, 1987). Dynamics and control constraints can be incorporated into this model as well; for example, by dividing planning into two stages—first finding a path that satisfies geometric constraints and then modifying it to fit the dynamics constraints (Shiller and Lu, 1992), possibly in a time-optimal fashion (Donald and Xavier, 1989; Shiller and Dubowsky, 1991).

One of the first attempts to explicitly incorporate body dynamics into the planning process is the *kinodynamic planning* approach of Canny *et al.* (1990). It was shown that the algorithm requires space that is polynomial in the input and runs in exponential time. Aside from the fact that it operates in the context of the paradigm with complete information, the approach is somewhat akin to the approach in this paper, in that the control strategy adheres to the L_∞ norm—the velocity and acceleration components are assumed to be bounded with respect to a fixed (absolute) reference system. This allows one to decouple the equations of robot motion and thus treat the two-dimensional problem as two one-dimensional problems.†

This paper is concerned with the second paradigm, called *motion planning with incomplete information,* or *sensor-based motion planning.* In this model the objects in the environment can be of arbitrary shape, and the input information is typically of local character, such as from a rangefinder or a vision sensor (Lumelsky and Skewis, 1990). By making use of the notion of sensor feedback, this paradigm naturally fits the on-line character of the techniques of control theory.

Each of the two paradigms gives rise to two types of techniques: those that consider only kinematic and geometric issues—for brevity, they can be called *kinematic* approaches—as opposed to the *dynamic* approaches, which do take into account body dynamics. We now consider both briefly.

† One could make the following comparison. Using the L_∞ norm will obviously result, both in Canny *et al.* (1990) and in our strategy, in a less-efficient use of the control resources and in a 'less-optimal' time of path execution. Since planning with complete information is a one-time computation, this loss in time is likely to be significant owing to a large deviation of the robot's moving reference system from the absolute (world) system over the whole path. In contrast, in the sensor-based paradigm the decoupling of controls occurs again and again, with the reference system fixed only for the duration of one step, and so the resulting loss in efficiency should be less.

Kinematic algorithms

In turn, the existing kinematic techniques can be divided into two groups—those for *holonomic* systems and those for *nonholonomic* systems. In a holonomic system any desired direction of motion is realizable; it is not so in a nonholonomic system (Greenwood, 1965), where the number of control variables is less than the problem dimensionality (as, for example in the car parallel-parking problem).

A number of kinematic *holonomic* strategies make use of the *artificial potential field.* They usually require complete information and analytical presentation of obstacles; the robot's motion is affected by the repulsive forces created by a potential field associated with obstacles, and by the attractive force associated with the goal position (Khatib, 1986). A typical convergence issue here is how to avoid possible local minima in the potential field. Modifications that attempt to solve this problem include the use of repulsive fields with elliptic isocontours (Volpe and Khosla, 1987), introduction of special global fields (Koditschek, 1987) and generation of a numerical field (Barraquand *et al.,* 1992). The vortex-field method (De Medio and Oriolo, 1991) allows one to avoid undesirable attraction points, while using only local information; the repulsive actions are replaced by the velocity flows tangent to the isocontours, so that the robot is forced to move around obstacles.

On the incomplete-information side, a variety of kinematic holonomic techniques originate in maze-searching strategies (Lumelsky and Stepanov, 1987; Lumelsky and Skewis, 1990; Sankaranarayanan and Vidyasagar, 1990); when applicable, they are usually fast, can be used in real time and guarantee convergence; the obstacles may be of arbitrary shape.

The existing kinematic *nonholonomic* strategies also require analytical representation of obstacles, and assume complete (Jacobs *et al.,* 1991; Latombe, 1991; Barraquand and Latombe, 1991) or partial input information (De Luca and Oriolo, 1994). These schemes are essentially open-loop, do not guarantee convergence, and attempt to solve the planning problem by taking into account the effect of the nonholonomic constraints on the obstacle avoidance. In Jacobs *et al.* (1991) and Latombe (1991) a two-stage scheme is considered: first, a holonomic planner generates a complete collision-free path, and then this path is modified to account for nonholonomic constraints. In Barraquand and Latombe (1991) the problem is reduced to searching a graph representing the discretized configuration space. In De Luca and Oriolo (1994) planning is done incrementally, with

partial information: first a desirable path is defined, and then a control is found that minimizes the error in a least-squares sense.

Dynamic algorithms

To design an exact *dynamic* algorithm for sensor-based motion planning, one needs a single control mechanism—separating it into stages as above is likely to destroy convergence. Convergence has two faces—globally it is a guarantee of finding a path to the target if one exists, and locally it is an assurance of collision avoidance in view of the robot's inertia. The former can be borrowed from kinematic algorithms; the latter requires an explicit consideration of dynamics.

To illustrate our approach, consider an example. Imagine a jogger in the city environment. Taking a run will involve continuous on-line control and decision making. This will require a number of mechanisms: first, a global planning (convergence) mechanism is needed to assure arriving at the target location in spite of all deviations and detours that the changes in the environment may require. Second, since an instantaneous stop is impossible because of inertia, in order to maintain a reasonable speed the jogger needs at any moment an 'insurance' option of a safe *stopping path.* Third, when the jogger starts turning the street corner and suddenly sees a heavy large object right on the intended path, some quick local planning must occur to take care of collision avoidance. The jogger's speed may temporarily decrease, and the path will smoothly divert from the object. This path segment is likely to be locally optimized, to arrive at the street corner quicker or along a shorter path. Or, other options being infeasible, the jogger may 'brake' to a halt, and then start a detour path.

Note that sensing, local planning, global planning, and actual movement in this process are taking place simultaneously and continuously. Locally, unless the right relationship is maintained between the velocity at the time of noticing the object, the distance to it, and the jogger's mass, collision may occur: for example, a bigger mass may dictate better (further) sensing to maintain the same velocity. Globally, unless a 'grand plan' is followed, convergence may be lost.

Although at any moment the robot has knowledge about the obstacles within its sensing range, it would not be reasonable to address the problem as a sequence of smaller problems with complete information. For one, this would require computing the whole piece of the path inside the sensing range, which itself consists of many steps. Doing that at each step (say, at the rate of 30–50 steps per second) would be computationally very expensive. Note that only the first step of this path is likely to be executed—the new sensing data will require changes. Thus, computing the whole path within the sensing range would be largely a computational waste.

Our algorithm for step calculation will (a) place the step on a globally converging collision-free path, while (b) satisfying the robot dynamics constraints. Assume that the sensing range ('radius of vision') is r_v. The general strategy will be as follows: at the moment (step) i, the kinematic algorithm chosen identifies an intermediate target point T_i, which is the farthest visible point on a convergent path—normally at the boundary of the sensing range r_v. Then a single step is calculated and executed that lies on a time-optimal trajectory to T_i; the robot moves from its current position C_i to the next position C_{i+1}; then the process repeats.

The fact that no information is available beyond the sensing range dictates a number of requirements. The emergency *stopping path* must lie inside the current sensing area. Since some parts of the sensing range may be occupied or occluded by obstacles, the stopping path must lie in its visible part. Then, the robot needs a guarantee of stopping at the intermediate target T_i, even if it does not intend to do so. That is, each step is to be planned as the first step of a trajectory that, given the current position, velocity and control constraints, would bring the robot to a halt at T_i.

The step planning task is formulated as an optimality problem. At each step, a *canonical solution* is found that, if no obstacles were present, would bring the robot from C_i to T_i with zero velocity and in minimum time. If the canonical path crosses an obstacle and is thus not feasible, a *near-canonical solution* path is found that is collision-free and satisfies the control constraints. We show that in this case only a small number of options need be considered, at least one of which is guaranteed to be collision-free.

By making use of the L_∞ norm within the duration of a single step, we decouple the two-dimensional problem into two one-dimensional control problems and reduce the task to the bang–bang control strategy. This results in an extremely fast procedure for finding the (fairly complex) time-optimal path within the sensing range, easily implementable in real time. Only the first step of this path is actually executed; this decreases the error due to the control decoupling. At the next moment, new

sensing data arrive and the process repeats. A special case is also considered where the intermediate target goes out of the robot's sight either because of the robot's inertia or because of occluding obstacles.

The terminology and model used are introduced in Section 2, followed by a sketch of the suggested approach in Section 3. Analysis of the system dynamics in Section 4 results in decoupling of the control problem into two one-dimensional problems, each with a bang–bang control solution. Although obtaining such a solution is a relatively simple and known problem, we have not been able to find in the literature the derivation of the general-form solution, or the proof of its sufficiency. Consequently, those details are given in the Appendix. Procedures for finding the canonical and near-canonical solutions appear in Sections 5 and 6. The complete algorithm and its convergence and complexity properties are discussed in Sections 7 and 8, and simulated examples of the algorithm performance are given in Section 9.

2. THE MODEL

The environment (work space) is two-dimensional physical space $\mathcal{W} \equiv (x, y) \subset \mathbb{R}^2$; it may include a finite set of locally finite static obstacles $\mathcal{O} \in \mathcal{W}$. Each obstacle $\mathcal{O}_k \in \mathcal{O}$ is a simple closed curve of arbitrary shape and of finite length, such that a straight line will cross it in only a finite number of points. Obstacles do not touch each other (if they do, they are considered as one obstacle).

The robot is a *point mass*, of mass m. The robot is equipped with sensors that allow it, at its current location C_i, to detect any obstacles and the distance to them within its *sensing range*—a disc $D(C_i, r_v)$ of radius r_v ('radius of vision') centered at C_i. At moment t_i, the robot's input information includes its current velocity vector $\mathbf{V}_i$, and the coordinates of C_i and of its target location T.

The robot's means for motion control are two components of the acceleration vector $\mathbf{u} = \mathbf{f}/m = (p, q)$, where m is the robot's mass, and $\mathbf{f}$ is the force applied. Controls $\mathbf{u}$ come from a set $\mathbf{u}(\cdot) \in U$ of measurable, piecewise-continuous bounded functions in $\mathbb{R}^2$, $U = \{\mathbf{u}(\cdot) = (p(\cdot), q(\cdot)) \mid p \in [-p_{\max}, p_{\max}, q \in [-q_{\max}, q_{\max}]\}$. By taking the mass $m = 1$, we can refer to the components p and q as control forces, each within a fixed range $|p| \le p_{\max}$, $|q| \le q_{\max}$; $p_{\max}$, $q_{\max} > 0$. The force p controls the forward (or backward when braking) motion; its positive direction coincides with the robot's velocity vector $\mathbf{V}$. The force q, the steering control, is perpendicular to p, forming a right pair of vectors (see Fig. 2 below). There is no friction: for example, given the velocity $\mathbf{V}$, the control values $p = q = 0$ will result in a constant-velocity straight-line motion along vector $\mathbf{V}$.

Without loss of generality, we assume that no external forces except p and q act on the system. Note that, under this assumption, our model and approach can still handle other external forces and constraints, using, for example, the technique suggested in Fraichard and Scheuer (1994), whereby various dynamic constraints such as curvature, engine force, sliding and velocity appear in the inequality describing the limitations on the components of acceleration. The use of such inequalities defines a convex region of the $(\ddot{x}, \ddot{y})$ space. In our case the control forces act within the intersection of the box $[-p_{\max}, p_{\max}] \times [-q_{\max}, q_{\max}]$ with the half-planes defined by those inequalities.

The task is to move in $\mathcal{W}$ from point S (start) to point T (target); see Fig. 1. The control of robot motion is done in steps $i = 0.1.2, \ldots$. Each step i takes a time $\delta t = t_{i+1} - t_i = \text{const}$; the path length within the time interval δt depends on the robot's velocity $\mathbf{V}_i$. Steps i and $i+1$ start at times t_i and t_{i-1} respectively; $C_0 = S$. The control forces $\mathbf{u}(\cdot) = (p, q) \in U$ are constant within the step.

We define three coordinate systems (follow Fig. 2).

- The *world frame* $(\mathbf{x}, \mathbf{y})$ is fixed at the point S.
- The *primary path frame* $(\mathbf{t}, \mathbf{n})$ is a moving (inertial) coordinate frame. Its origin is attached to the robot; the axis $\mathbf{t}$ is aligned with the current velocity vector $\mathbf{V}$; the axis $\mathbf{n}$ is normal to $\mathbf{t}$. Together with the axis $\mathbf{b}$, which is the cross-product $\mathbf{b} = \mathbf{t} \times \mathbf{n}$, the triple $(\mathbf{t}, \mathbf{n}, \mathbf{b})$

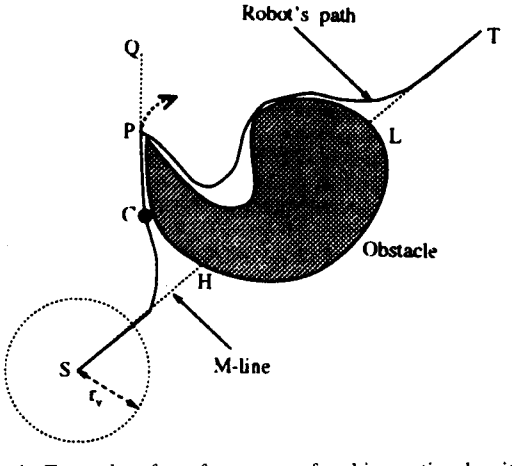

Fig. 1. Example of performance of a kinematic algorithm (Lumelsky and Skewis, 1990).

forms the *Frenet trihedron,* with the plane of **t** and **n** forming the *osculating plane* (Korn and Korn, 1968).

- The *secondary path frame,* (ξ_i, η_i), is a coordinate frame that is fixed during the time interval of the step i. The frame's origin is at the intermediate target T_i, with the axis ξ_i aligned with the velocity vector $\mathbf{V}_i$ at time t_i, and the axis η_i normal to ξ_i.

For convenience, we combine the requirements and constraints that affect the control strategy into a set, called Ω. A solution (a path, a step or a set of control values) is said to be Ω-*acceptable* if, given the current position C_i and velocity $\mathbf{V}_i$,

(i) it satisfies the constraints $|p| \leq p_{\max}$ and $|q| \leq q_{\max}$ on the control forces,

(ii) it guarantees a stopping path, and

(iii) it results in a collision-free motion.

3. THE APPROACH

The algorithm developed below is to be executed at each step of the robot's path. The procedure combines the convergence mechanism of a kinematic sensor-based motion-planning algorithm with a control mechanism for handling dynamics, resulting in a single operation. During the step time interval i, the robot will maintain within its sensing range an *intermediate target* point T_i, usually on an obstacle boundary or on the desired path—say, a straight line. At its current position C_i, the robot will plan and execute its next step towards T_i, then at C_{i+1} analyze new sensory data and define a new intermediate target T_{i+1}, and so on. At times the current T_i may go out of the robot's sight because of its inertia or due to occluding obstacles. In such cases the robot will rely on *temporary intermediate targets* until it can see the point T_i again.

The kinematic part

In principle, any maze-searching procedure can be utilized for the kinematic part of the algorithm, so long as it allows an extension to distant sensing. For the sake of specificity, we use here the VisBug algorithm (Lumelsky and Skewis, 1990). Below, the *M line* (*Main line*) is the straight line connecting the points S and T; it is the robot's desired path. When, while moving along the M line, the robot senses on its way an obstacle, this point on the obstacle boundary is called a *hit point, H.* The corresponding intersection point between the M line and the obstacle 'on the other side' of the obstacle is a *leave point, L.* Roughly, the algorithm revolves around two steps (see Fig. 1).

1. Walk from S toward T along the M line until, at some point detecting an obstacle crossing the M line, say at point H. Go to Step 2.

2. Define a farthest visible intermediate target T_i on the obstacle boundary in the direction of motion; make a step toward T_i. Iterate Step 2 until detecting the M line. Go to Step 1.

The actual algorithm includes other mechanisms, such as a finite-time target-reachability test and local path optimization. In the example shown in Fig. 1, note that, while trying to pass the obstacle from the left, at the point P the robot would make a sharp turn—the algorithm assumes holonomic motion. In our case, however, such motion will not be possible because of the robot's mass and velocity—the motion is non-holonomic (the dotted path beyond the point P).

The effect of dynamics

Dynamics affects three algorithmic issues: safety considerations, step planning and convergence. We consider these separately.

Safety considerations. These appear in a number of ways. Since no information about the obstacles is available beyond distance r_v from the robot, guaranteeing collision-free motion means assuring at any moment at least one 'last-resort' stopping path. Otherwise, at the next step new obstacles may appear in the sensing range, and collision will be imminent no matter what control is used. This dictates a certain relationship between the velocity $\mathbf{V}$, mass m, radius r_v and controls $\mathbf{u} = (p, q)$. Under a straight-line motion, the range of safe velocities must satisfy

$$V \leq \sqrt{2pd}, \tag{1}$$

where d is the distance from the robot to the stop point. That is, if the robot moves with the maximum velocity, the stop point of the *stopping path* must be no further than r_v from the current position C. In practice, (1) can be interpreted in a number of ways. Note that the maximum velocity is proportional to the acceleration due to control, which is in turn directly proportional to the force applied and inversely proportional to the robot's mass. For example, if the mass m becomes larger, while other parameters stay the same, the maximum velocity will decrease. Conversely, if the limits on (p, q) increase (say, because more powerful motors) then the maximum velocity will increase as well. Or, an increase in the radius r_v (say, due to better

sensors) allows one to increase the maximum velocity, by virtue of utilizing more information about the environment.

Consider the example in Fig. 1. When approaching the point P along its path, the robot will see it at distance r_v and will designate it as its next intermediate target T_i. Along this path segment, the point T_i happens to stay at P because no further point on the obstacle boundary will be visible until the robot arrives at P. Though there may be an obstacle right around the corner P, the robot need not slow down, since at any point of this segment there is a possibility of a stopping path ending somewhere around the point Q. That is, in order to proceed with maximum velocity, the availability of a stopping path has to be ascertained at every step i. Our stopping path will be a straight line path along the corresponding vector $\mathbf{V}_i$. If a candidate step cannot guarantee a stopping path, it is discarded. (Note that a deeper, multistep analysis would be hardly justifiable here because of high computational costs, though occasionally it could produce locally shorter paths.)

Step planning. Normally the stopping path is not used—it is only an 'insurance' option. The actual step is based on the *canonical solution,* a path that would bring the robot from C_i to T_i with zero velocity and in minimum time, assuming no obstacles. The optimization problem is set up based on Pontryagin's optimality principle. We assume that within a step—i.e. within the time interval $[t_i, t_{i+1})$—the system's controls (p, q) are bounded in the L_∞ norm, and apply it with respect to the secondary coordinate frame (ξ_i, η_i). The result is a fast computational scheme that is easily implementable in real time. Only the very first step of the solution is used for the actual motion; then, another solution is calculated using new sensory information, and so on. With such a step-by-step execution of the optimization scheme, on the one hand, a good approximation of the globally time-optimal path from C_i to T_i is produced, and, on the other hand, little computation is wasted on the part of the path solution that will not be utilized.

If the step suggested by the canonical solution is not feasible because of obstacles, a close approximation, called the *near-canonical solution,* is found that is both feasible and Ω-acceptable. For this case, we show, first, that only a finite number of path options need be considered, and, second, that there exists at least one path solution that is Ω-acceptable. A special case here is when the intermediate target goes out of the robot's sight either because of the robot's inertia or because of occluding obstacles.

Convergence. Once a step has been physically executed, new sensing information appears and the process repeats. The procedure for a detour around a sudden obstacle is tied to the issue of convergence, and is constructed similarly to the case of normal motion. Because of the effect of dynamics, the convergence mechanism borrowed from a kinematic algorithm—here VisBug—will need some modification. The intermediate target points T_i produced by VisBug lie either on the boundaries of obstacles or on the M line, and are visible from the corresponding robot positions. However, the robot's inertia may cause it to move so that T_i will become invisible, either because it goes outside the sensing range r_v (as after the point P on Fig. 1), or because of occluding obstacles (as in Fig. 5 below). This may endanger path convergence. A safe but inefficient solution would be to slow down or to keep the speed small at all times to avoid such overhsoots. The solution chosen (see Section 7) is to keep the velocity high and, if the intermediate target T_i does go out of sight, to modify the motion locally until T_i is found again.

4. DYNAMICS AND COLLISION AVOIDANCE

Consider a time sequence $\sigma_t = \{t_0, t_1, t_2, \ldots\}$ of the starting moments of steps. Step i takes place within the interval $[t_i, t_{i+1})$, $t_{i+1} - t_i = \delta t$. At the moment t_i the robot is at the position C_i, with the velocity vector $\mathbf{V}_i$. Within this interval, based on the sensing data, the intermediate target T_i (supplied by the kinematic planning algorithm) and the vector $\mathbf{V}_i$, the control system will calculate the values of the control forces p and q. The forces are then applied to the robot, and the robot executes step i, finishing it at the point C_{i+1} at the moment t_{i+1}, with the velocity vector $\mathbf{V}_{i+1}$; then the process repeats.

The analysis that leads to the procedure for handling dynamics consists of three parts. First, in the remainder of this section we incorporate the control constraints into the robot model and develop the transformations between the primary path frame and world frame, and between the secondary path frame and world frame. Then, in Section 5 we develop the *canonical solution,* and, finally, in Section 6 the *near-canonical solution,* for the case when the canonical solution would result in a collision. The resulting algorithm operates incrementally, and computes the forces p and q at each step. The remainder of this section refers to the time interval $[t_i, t_{i+1})$ and its intermediate target T_i, and so the the index i is often dropped.

Denote by $(x, y) \in \mathbb{R}^2$ the robot's position in the world frame, and by θ the (slope) angle between the velocity vector $\mathbf{V} = (V_x, V_y) = (\dot{x}, \dot{y})$

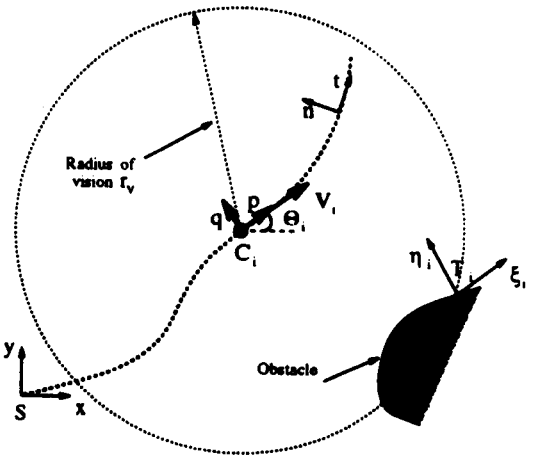

Fig. 2. The coordinate frame (**x**, **y**) is the *world frame* with its origin at S; (**t**, **n**) is the *primary path frame*, and (ξ_i, η_i) is the *secondary path frame* for the current robot position C_i.

and the **x** axis of the world frame (Fig. 2). The planning process involves computation of the controls $\mathbf{u} = (p, q)$, which for every step define the velocity vector and eventually the path $(x(t), y(t))$ as a function of time. Taking the mass $m = 1$, the equations of motion become

$$\ddot{x} = p\cos\theta - q\sin\theta,$$

$$\ddot{y} = p\sin\theta + q\cos\theta.$$

The angle θ between the vector $\mathbf{V} = (V_x, V_y)$ and the **x** axis of the world frame is found as

$$\theta = \begin{cases} \arctan(V_y/V_x) & (V_x \geq 0), \\ \arctan(V_y/V_x) + \pi, & (V_x < 0). \end{cases}$$

The transformations between the world frame and the secondary path frame, from (x, y) to (ξ, η) and from (ξ, η) to (x, y), are given by

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = R\begin{pmatrix} x - x_T \\ y - y_T \end{pmatrix}, \tag{2}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = R'\begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} x_T \\ y_T \end{pmatrix}, \tag{3}$$

where

$$R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix},$$

R' is the transpose of the matrix of the rotation matrix between the frames ($\boldsymbol{\xi}$, $\boldsymbol{\eta}$) and (**x**, **y**), and (x_T, y_T) the coordinates of the (intermediate) target in the world frame (**x**, **y**).

To define the transformations between the world frame (**x**, **y**) and the primary path frame (**t**, **n**), write the velocity in the primary path frame as $\mathbf{V} = V\mathbf{t}$. To find the time derivative of the vector **V** with respect to the world frame (**x**, **y**), note that the time derivative of the vector **t** in the primary path frame (see Section 2) it not equal to zero, and can be defined as the cross-product of the angular velocity $\boldsymbol{\omega} = \dot{\theta}\mathbf{b}$ of the primary path frame and the vector **t** itself: $\dot{\mathbf{t}} = \boldsymbol{\omega} \times \mathbf{t}$, where the angle θ is between the unit vector **t** and the positive direction of the **x** axis. Given that the control forces p and q act along the **t** and **n** directions respectively, the equations of motion with respect to the primary path frame are

$$\dot{V} = p,$$

$$\dot{\theta} = q/V.$$

Since p and q are constant over the time interval $t \in [t_i, t_{i+1})$, the solution for $V(t)$ and $\theta(t)$ within the interval becomes

$$\begin{aligned} \mathrm{V}(\mathrm{t}) &= pt + V_0, \\ \theta(t) &= \theta_0 + \frac{q\log(1 + tp/V_i)}{p} \end{aligned} \tag{4}$$

where θ_0 and V_0 are constants of integration and are equal to the values of $\theta(t_i)$ and $V(t_i)$ respectively. By parameterizing the path by the value and direction of the velocity vector, the path can be mapped into the world frame (**x**, **y**) using the vector integral equation

$$\mathbf{r}(t) = \int_{t_i}^{t_{i+1}} V \cdot \mathbf{t}\, dt. \tag{5}$$

Here $\mathbf{r}(t) = (x(t), y(t))$, and **t** is a unit vector in the direction of **V**, with the projections $\mathbf{t} = (\cos\theta, \sin\theta)$ onto the world frame (**x**, **y**). Integration (5) gives the set of solutions,

$$\begin{aligned} x(t) &= \frac{2p\cos\theta(t) + q\sin\theta(t)}{4p^2 + q^2} V^2(t) + A, \\ y(t) &= -\frac{q\cos\theta(t) - 2p\sin\theta(t)}{4p^2 + q^2} V^2(t) + B, \end{aligned} \tag{6}$$

where

$$A = x_0 - \frac{V_0^2(2p\cos\theta_0 + q\sin\theta_0)}{4p^2 + q^2}$$

$$B = y_0 + \frac{V_0^2(q\cos\theta_0) - 2p\sin\theta_0)}{4p^2 + q^2},$$

and $V(t)$ and $\theta(t)$ are given by (4).

Equations (6) describe a spiral curve. Note two special cases: when $p \neq 0$ and $q = 0$, (6) describe straight-line motion under the force along the velocity vector; when $p = 0$, $q \neq 0$, the force acts perpendicularly to the vector of velocity, and (6) produce a circle of radius $V_0^2/|q|$ centered at the point (A, B).

5. THE CANONICAL SOLUTION

Given the current position $C_i = (x_i, y_i)$, the intermediate target T_i and the velocity vector $\mathbf{V}_i = (\dot{x}_i, \dot{y}_i)$, the canonical solution presents a path that, assuming no obstacles are present, will bring the robot from C_i to T_i with zero velocity and in minimum time. The L_∞-norm assumption allows us to decouple the bounds on the accelerations in the ξ and η directions, and thus treat the two-dimensional problem as a set of two one-dimensional problems, one for control p and the other for control q. For details on the solution of a one-dimensional control problem, refer to the Appendix.

The optimization problem is formulated based on Pontryagin's (1962) optimality principle with respect to the secondary frame (ξ, η). We seek to optimize a criterion F that signifies time. Assume the trajectory being sought starts at time $t = 0$ and ends at time $t = t_f$ (for 'final'). Then the problem in hand is

$$F(\xi(\cdot), \eta(\cdot), t_f) = t_f \to \infty,$$
$$\ddot{\xi} = p, \quad \|p\| \le p_{\max},$$
$$\ddot{\eta} = q, \quad \|q\| \le q_{\max},$$
$$\xi(0) = \xi_0, \quad \eta(0) = \eta_0, \quad \dot{\xi}(0) = \dot{\xi}_0, \quad \dot{\eta}(0) = \dot{\eta}_0,$$
$$\eta(t_f) = \eta(t_f) = \dot{\xi}(t_f) = \dot{\eta}(t_f) = 0.$$

Analysis shows (see the details in the Appendix) that the optimal solution of each one-dimensional problem corresponds to 'bang–bang' control, with at most one switching along each of the directions ξ and η, at the times $t_{s,\xi}$ and $t_{s,\eta}$ ('s' for 'switching') respectively.

The switching curves for control switchings are two connected parabolas in the phase space $(\xi, \dot{\xi})$,

$$\xi = \begin{cases} -\dfrac{\dot{\xi}^2}{2p_{\max}} & (\dot{\xi} > 0), \\ \dfrac{\dot{\xi}^2}{2p_{\max}} & (\dot{\xi} < 0) \end{cases} \tag{7}$$

and in the phase space $(\eta, \dot{\eta})$

$$\eta = \begin{cases} -\dfrac{\dot{\eta}^2}{2q_{\max}} & (\dot{\eta} > 0), \\ \dfrac{\dot{\eta}^2}{2q_{\max}} & (\dot{\eta} < 0) \end{cases} \tag{8}$$

(see Fig. 3). The time-optimal solution is then obtained using the bang–bang strategy for ξ and η, depending on whether the starting points, $(\xi, \dot{\xi})$ and $(\eta, \dot{\eta})$ are above or below their corresponding switch curves, as follows:

$$\hat{p}(t) = \begin{cases} \alpha_1 p_{\max} & (0 \le t \le t_{s,\xi}), \\ \alpha_2 p_{\max} & (t_{s,\xi} < t \le t_f), \end{cases}$$
$$\hat{q}(t) = \begin{cases} \alpha_1 q_{\max} & (0 \le t \le t_{s,\eta}), \\ \alpha_2 q_{\max} & (t_{s,\eta} < t < t_f), \end{cases} \tag{9}$$

where $\alpha_1 = -1$ and $\alpha_2 = 1$ if the starting point, $(\xi, \dot{\xi})_s$ or $(\eta, \dot{\eta})_s$ respectively, is above its switch curves, and $\alpha_1 = 1$ and $\alpha_2 = -1$ if the starting point is below its switch curves. For example, if the initial conditions for ξ and η are as shown in Fig. 3 then

$$\hat{p}(t) = \begin{cases} -p_{\max} & (0 \le t \le t_{s,\xi}), \\ +p_{\max} & (t_{s,\xi} < t \le t_f), \end{cases}$$
$$\hat{q}(t) = \begin{cases} +q_{\max} & (0 \le t \le t_{s,\eta}), \\ -q_{\max} & (t_{s,\eta} < t \le t_f), \end{cases} \tag{10}$$

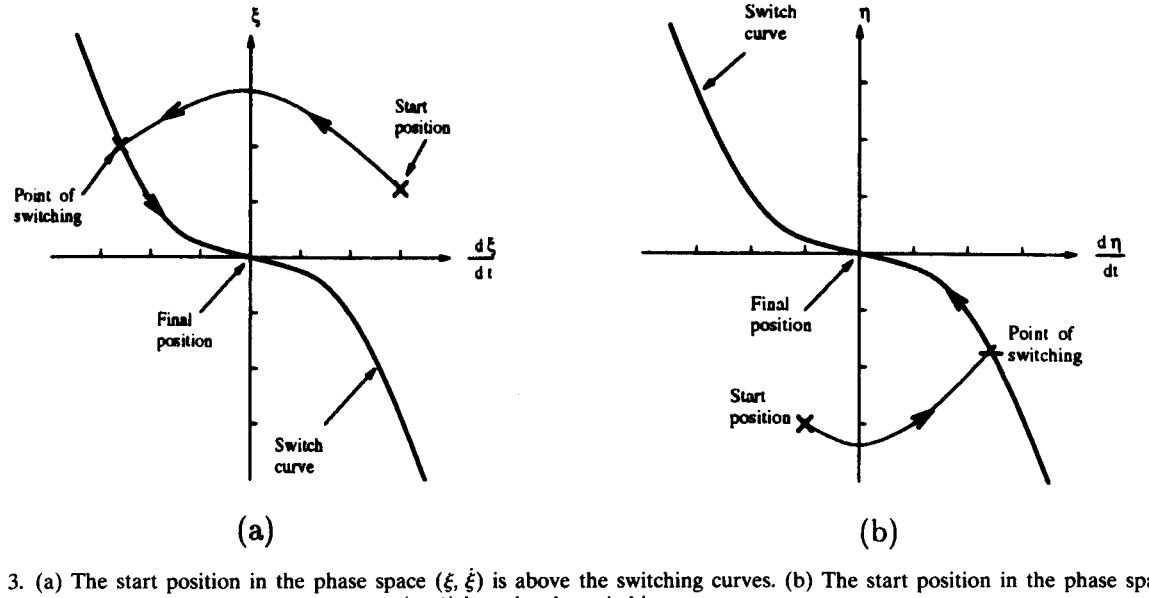

Fig. 3. (a) The start position in the phase space $(\xi, \dot{\xi})$ is above the switching curves. (b) The start position in the phase space $(\eta, \dot{\eta})$ is under the switching curves.

where the 'hat' indicates the parameters under optimal control. The time, position and velocity of the control switching for the ξ components are given by

$$t_{s,\xi} = \frac{\sqrt{\frac{1}{2}(\dot{\xi}_0)^2 + \xi_0 p_{max}} + \dot{\xi}_0}{p_{max}},$$

$$\xi_s = \frac{\dot{\xi}_0^2}{4p_{max}} + \frac{\xi_0}{2},$$

$$\dot{\xi}_s = -\sqrt{\frac{\dot{\xi}_0^2}{2p_{max}} + \xi_0 p_{max}},$$

and those for the η components are given by

$$t_{s,\eta} = \frac{\sqrt{\frac{1}{2}(\dot{\eta}_0)^2 - \eta_0 q_{max}} - \dot{\eta}_0}{q_{max}},$$

$$\eta_s = -\frac{\dot{\eta}_0^2}{4q_{max}} + \frac{\eta_0}{2}$$

$$\dot{\eta}_s = \sqrt{\frac{\dot{\eta}_0^2}{2q_{max}} - \eta_0 q_{max}}.$$

The number, time and locations of switchings can be uniquely defined from the initial and final conditions. It can be shown (see the Appendix) that for every position of the robot in $\mathbb{R}^4(\xi, \eta, \dot{\xi}, \dot{\eta})$, the control law obtained guarantees time-optimal motion in both the $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ directions, as long as the time interval considered is sufficiently small. Substituting this control law into the equations of motion (6) produces the canonical solution.

To summarize, the procedure for obtaining the first step of the canonical solution is as follows.

1. Substitute the current position/velocity $(\xi, \eta, \dot{\xi}, \dot{\eta})$ into (7) and (8) and see if the starting point is above or below the switching curves.

2. Depending on the above/below information, take one of the four possible bang–bang control pairs p, q from (9).

3. With this pair (p, q), find from (6) the position C_{i+1} and from (4) the velocity $\mathbf{V}_{i+1}$ and angle θ_{i+1} at the end of the step. If this step to C_{i+1} crosses no obstacles, and if there exists a stopping path in the direction $\mathbf{V}_{i-1}$, the step is accepted; otherwise, a near-canonical solution is sought (see Section 6).

Note that though the canonical solution defines a fairly complex multistep path from C_i to T_i, only one—the very first—step of that path is calculated explicitly. The switching curves (7) and (8) and the position and velocity equations (6) and (4) are quite simple. The whole computation is therefore very fast.

6. NEAR-CANONICAL SOLUTION

As discussed above, unless a step being considered for the next moment guarantees a stopping path along its velocity vector, it will be rejected. This step will be always the very first step of the canonical solution. Therefore, if the stopping path of this candidate step happens to cross an obstacle within the distance found from (1), the controls are modified into a *near-canonical solution* that is both Ω-acceptable and reasonably close to the canonical solution. The near-canonical solution is one of the nine possible combinations of the bang–bang control pairs $(k_1 p_{max}, k_2 q_{max})$, where k_1 and k_2 are chosen from the set $\{-1, 0, 1\}$ (see Fig. 4). Since the canonical solution takes one of those nine control pairs, the near-canonical solution is to be chosen from the remaining eight pairs. This set is guaranteed to contain an Ω-acceptable solution: since the current position has been chosen so as to guarantee a stopping path, such a last-resort path—for example, under the control $(-p_{max}, 0)$—always exists. Further, the position of the intermediate target T_i relative to the vector $\mathbf{V}_i$—in its left or right half-plane—suggests an ordered and thus shorter search among the control pairs. For step i, denote the nine control pairs $\mathbf{u}_i^j$, $j = 0, 1, 2, \ldots, 8$, as shown in Fig. 4. If, for example, the canonical solution is $\mathbf{u}_i^2$ then the near-canonical solution will be the first Ω-acceptable control pair $\mathbf{u}^j = (p, q)$ from

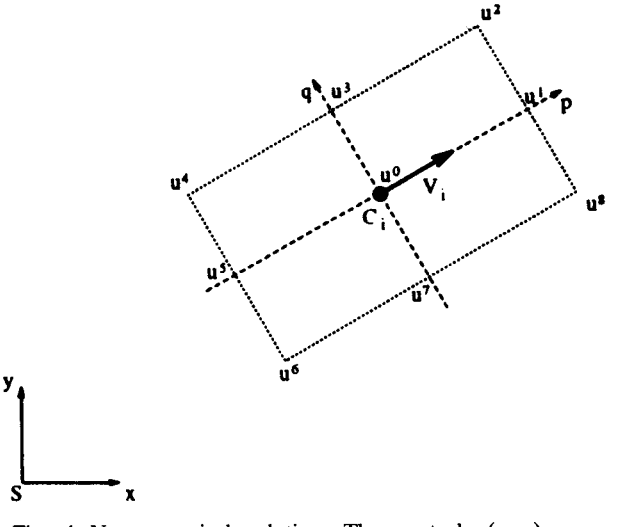

Fig. 4. Near-canonical solution. The controls (p, q) are assumed to be L_∞-norm bounded on the small interval of time. The choice of (p, q) is among eight 'bang–bang' solutions shown.

the sequence ($\mathbf{u}^3$, $\mathbf{u}^1$, $\mathbf{u}^4$, $\mathbf{u}^0$, $\mathbf{u}^8$, $\mathbf{u}^5$, $\mathbf{u}^7$, $\mathbf{u}^6$); note that $\mathbf{u}^5$ is always Ω-acceptable.

7. THE ALGORITHM

The complete motion-planning algorithm is executed continuously at every step of the path, and generates motion by computing canonical or near-canonical solutions at each step. It includes four procedures.

(i) The *Main Body* procedure monitors the general control of motion towards the intermediate target T_i.

In turn, *Main Body* makes use of three procedures.

(ii) Procedure *Define Next Step* chooses the controls (p, q) for the next step.

(iii) Procedure *Find Lost Target* deals with the special case when the intermediate target T_i goes out of the robot's sight.

(iv) *Main Body* also uses the procedure called *Compute* T_i, taken directly from the kinematic algorithm (Lumelsky and Skewis, 1990), which computes the next intermediate target T_{i+1} so as to guarantee the global convergence, and also performs the test for target reachability.

As before, S is the starting point, T is the robot's target; at step i, C_i is the robot's current position and the vector $\mathbf{V}_i$ is the current velocity vector. Initially, $i = 0$, $C_i = T_i = S$.

Procedure Main Body. At each step i:
If $C_i = T$, stop.
Find T_i from *Compute* T_i.
If T is found unreachable, stop.
If T_i is visible, find C_{i+1} from *Define Next Step*; make a step towards C_{i+1}; iterate; else.
Use *Find Lost Target* to produce T_i visible; iterate.

Procedure Define Next Step. At step i:

S1. Find the canonical solution (the switch curves and controls (p, q)) using (7)–(9). If it is Ω-acceptable, exit; else go to S2.

S2. Find the near-canonical solution as in Section 6; exit.

Procedure Find Lost Target. This is executed when T_i becomes invisible. The last position C_i where T_i was visible is then stored until T_i becomes visible again. Various heuristics can be used here, as long as convergence is preserved. One simple strategy would be to come to a halt using a stopping path, then come back to C_i with zero velocity, and then move directly toward T_i. The procedure chosen below is a bit smarter: if the robot loses T_i, it keeps moving ahead while defining temporary intermediate targets T_i^t on the visible part of the line segment (C_i, T_i), and continuing looking for T_i. Then, if it finds T_i, it moves directly toward it (Fig. 5a) otherwise, if the whole segment (C_i, T_i) becomes invisible, the robot brakes to a stop, returns to C_i with zero velocity, and then moves directly toward T_i (Fig. 5b) as follows.

S1. While at C_k, $k > i$, find on the segment (C_i, T_i) the visible point closest to T_i; denote it by T_k^t. If there is no such point (i.e. if the whole segment (C_i, T_i) is not visible), go to S2. Else, using *Define Next Step,* compute and execute the next step using T_k^t as the temporary intermediate target; iterate.

S2. Initiate a stopping path, then go back to C_i; exit.

8. CONVERGENCE. COMPUTATIONAL COMPLEXITY

Convergence

The collision-free motion along the path is guaranteed by the design of the canonical and near-canonical solutions. To prove convergence, we need to show that the algorithm will find a path to the target position T if one exists, or it will infer in finite time the nonreachability of T if true. This is guaranteed by the convergence properties of the kinematic algorithm (Lumelsky and Skewis, 1990). The following statements hold.

Claim 1. Under the algorithm, assuming zero velocity $\mathbf{V}_S = 0$ at the start position S, at each step of the path there exists at least one stopping path.

To see this, recall that, according to our model, the stopping path lies along a straight line. Guaranteeing a stopping path implies two requirements: a *safe direction* and the velocity value such as to allow a stop within the visible area. Because the latter is assured by the choice of the system parameters in (1), we focus now on the existence of safe directions. We proceed by induction: we have to show that if a safe direction exists at the start point and on an arbitrary step i then there is a safe direction on step $i + 1$.

Since at the start point S the velocity is zero, any direction of motion at S will be a safe direction—this gives the basis of induction. The induction step is as follows. Under the algorithm,

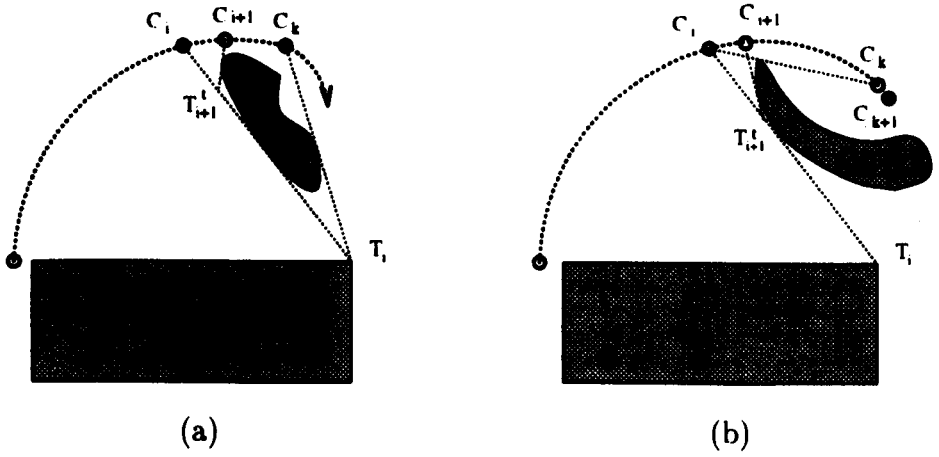

Fig. 5. In these examples, because of the system inertia, the robot temporarily 'loses' the intermediate target point T_i. In (a), it keeps moving forward until at C_k it sees T_i. In (b), at C_k it sees T_i. In (b), at C_k the robot initiates a stopping path, stops at C_{k+1}, returns back to C_i and moves toward T_i along the line (C_i, T_i).

a candidate step is accepted for execution only if its direction guarantees a safe stop for the robot if needed. Namely, at the point C_i, step i is executed only if the resulting vector $\mathbf{V}_{i+1}$ at C_{i+1} will point in a safe direction. Therefore, at step $i+1$, at the least this very direction can be used for a stopping path.

Remark. Claim 1 will hold for $\mathbf{V}_S \neq 0$ as well if there exists at least one stopping path originating at the start point S.

Claim 2. The algorithm guarantees convergence.

To see this, note that at each step i at its current position C_i the robot uses its sensing to generate the next intermediate target point T_i. That point T_i is known to lie on a convergent path of the kinematic algorithm (Lumelsky and Skewis, 1990). At the moment when T_i is generated, it is visible (see above). The only way the robot can get lost then is if at the next step(s) C_{i+1} the point T_i becomes invisible because of the robot's inertia or an obstacle occlusion: this would make it impossible to generate the intermediate target T_{i+1} as required by the kinematic algorithm. But, if indeed point T_i becomes invisible, the *Find Lost Target* procedure (above) is invoked, and a set of temporary intermediate targets T^t_{i+1} and associated steps are executed until point T_i again becomes visible (see Fig. 5a). Thus the robot is always moving toward a point which lies on a convergent path and itself converges to the target T.

Computational complexity

As with other on-line sensor-based algorithms, it is not very informative to assess the algorithm complexity in the way it is usually done for algorithms with complete information (Canny *et al.*, 1990). This is because in the latter one deals with one-time computation, whereas in the former the important complexity measure is the amount of computations at each step; the total computation time is simply a linear function of the path length. (Recall that with a sampling rate or, say, 50 per second, each step calculation must be done within the 20 ms interval.)

As shown in Section 5, though the canonical solution found by the algorithm at each path step is the solution of a fairly complex time-optimal problem, its computational cost is remarkably low, thanks to the (optimal) bang–bang control. This computation includes substituting the initial conditions $(\xi, \eta, \dot{\xi}, \dot{\eta})$ into the equations for parabolas (7), and (8) to see if the start point is above or below the corresponding parabola, and then simply taking the corresponding control pair $(\hat{p}, \hat{q})$ from the four choices in (9). The parabola equations themselves are found beforehand, only once. The near-canonical solution, when needed, is similar and as fast. Note that a single-step computation is of constant time: though the canonical solution represents the whole multistep trajectory within the sensing range of radius r_v, the computational time is independent of the value r_v and the length of path within the sensing range.

9. EXAMPLES

Figure 6 illustrates the performance of the proposed algorithm in simulated examples. The robot's mass m and the constraints on the control parameters p and q are the same for all examples: $p_{max} = q_{max}$ and $p_{max}/m = 1$. The generated paths are shown by thicker lines. For comparison, also shown as thin lines are the

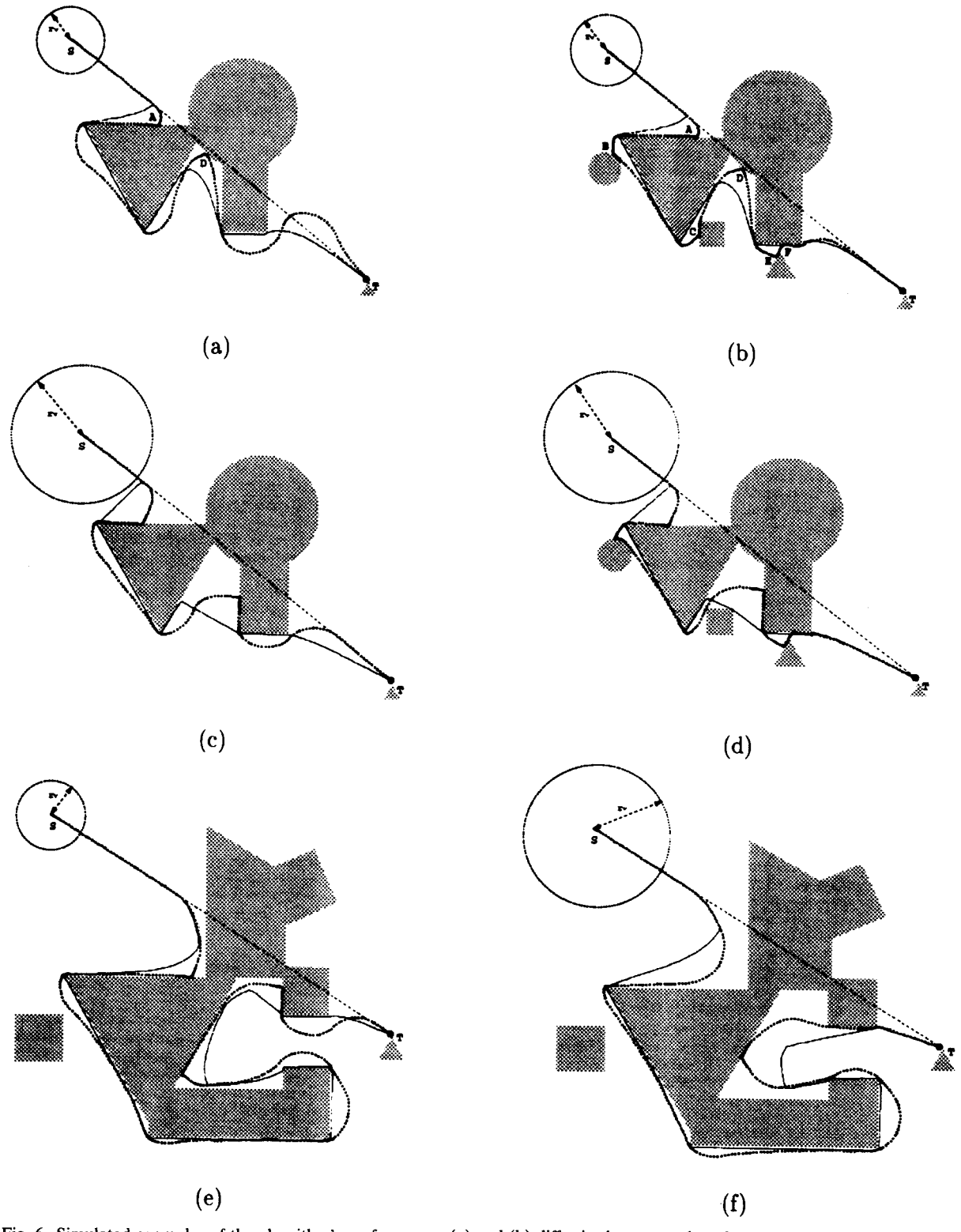

Fig. 6. Simulated examples of the algorithm's performance. (a) and (b) differ in that more obstacles are added in (b). (c) and (d) relate to the same respective scenes, and have larger radius of vision r_v. (e) and (f) relate to a different scene. Note the stopping points along the paths.

corresponding paths produced under the same conditions by the kinematic algorithm VisBug (Lumelsky and Skewis, 1990).

In both Figs. 6 (a) and (b) the radius of vision r_v is the same; the difference is in the environment: in Fig. 6(b) there are additional obstacles that the robot suddenly uncovers at a close distance when turning around a corner. Note that in Fig. 6(b) the path becomes tighter and shorter, though it takes longer: measured in the number of steps, the path in Fig. 6(a) takes 242 steps, and in Fig. 6(b) 278 steps; one might

say the robot becomes more cautious in Fig. 6(b). A similar pair of examples shown in Figs 6(c, d) illustrates the effect of the radius of vision: in (c) and (d) r_v is twice that of Figs 6(a, b). The times to execute the path are 214 and 244 steps respectively—shorter than in the corresponding examples in Figs 6(a, b). That is, in these examples better sensing (larger r_v) results in shorter time to complete the task; more crowded space results in longer time (though perhaps in shorter paths).

Note that at a few points the system found it necessary to make use of the stopping path—those points are usually easy to recognize from the sharp turns in the path. For example, in Fig. 6(a) the robot came to a halt at the points *A* and *D*, and in Fig. 6(b) it stopped at the points *A*, *B*, *C*, *D*, *E* and *F*. The algorithm's performance in a more complicated environment is shown in Figs 6(e, f): in (f) the radius of vision r_v is 1.4 times that of (e). Note again that richer input information (further sensing range) is likely to translate into shorter paths.

10. CONCLUSIONS

As the existing approaches to real-time (sensor-based) motion planning tend to deal solely with kinematic and geometric issues, one has difficulty using them in situations where the system dynamics cannot be ignored. This work has attempted to incorporate body dynamics into the paradigm of sensor-based motion planning. While the accepted assumption of a point-mass mobile robot is still a simplification of some real-world problems, it does catch the basic relation between dynamics and motion planning, and gives a basic paradigm for other, perhaps more complex, problems of this kind. Other properties of the accepted model are quite realistic; the robot is assumed to operate in an environment with unknown stationary obstacles of arbitrary shapes. Given the constraints on the robot's dynamics, sensing and control means, conditions have been formulated for generating locally optimal collision-free trajectories with guaranteed convergence. The approach calls for continuous computation, and is fast enough for real-time implementation.

Based on its velocity and sensing data, the robot continuously plans its motion based on canonical solutions, each of which presents a time-optimal path within the robot's sensing range. For a special case of a sudden potential collision, an option of a safe emergency stopping path is always maintained. As the convergence properties come from an appropriate static procedure of motion planning, the resulting strategy guarantees that, in spite of various constraints on velocity, mass, etc., our 'jogger' will always arrive at the destination if a path exists, or infer in finite time that the destination is not reachable if such is the case. Simulated examples demonstrate good performance of the algorithm.

Acknowledgements—This work is supported in part by the U.S. Sea Grant R/NI-20 and DOE (Sandia Labs) Grant 18-4379C.

REFERENCES

Barraquand, J. and Latombe, J. C. (1991) Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *Proc. IEEE International Conf. on Robotics and Automation,* Sacramento, CA, pp. 2328–2335.

Barraquand, J., Langlois, B. and Latombe, J. C. (1992) Numerical potential fields techniques for robot path planning. *IEEE Trans. Syst., Man, Cyber.* **SMC-22,** 224–241.

Canny, J. (1987) A new algebraic method for robot motion planning and real geometry. In *Proc. 28th IEEE Symp. on Foundations of Computer Science,* Los Angeles, CA, pp. 39–48.

Canny, J., Rege, A. and Reif, J. (1990) An exact algorithm for kinodynamic planning in the plane. In *Proc. 6th Annual Symp. on Computational Geometry,* Berkeley, CA, pp. 271–280.

Donald, B. and Xavier, P. (1989) A provably good approximation algorithm for optimal-time trajectory planning. In *Proc. IEEE International Conf. on Robotics and Automation,* Scottsdale, AZ, pp. 958–964.

De Luca, A. and Oriolo, G. (1994) Local incremental planning for nonholonomic mobile robots. In *Proc. IEEE International Conf. on Robotics and Automation,* San Diego, CA, pp. 104–110.

De Medio, C. and Oriolo, G. (1991) Robot obstacle avoidance using vortex fields. In *Advances in Robot Kinematics,* ed. S. Stifter and J. Lenarcic, pp. 227–235. Springer-Verlag, New York.

Fraichard, T. and Scheuer, A. (1994) Car-like robots and moving obstacles. In *Proc. IEEE International Conf. on Robotics and Automation,* San Diego, CA, pp. 64–69.

Greenwood, D. T. (1965) *Principles of Dynamics.* Prentice-Hall, New York.

Hocking, L. (1991) *Optimal Control.* Clarendon Press, Oxford.

Jacobs, P., Laumond, J. P. and Taix, M. (1991) Efficient motion planners for nonholonomic mobile robots. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS),* Osaka, Japan, pp. 1229–1235.

Khatib, O. (1986) Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robotics Res.* **5,** 90–99.

Koditschek, D. (1987) Exact robot navigation by means of potential functions: Some topological considerations. In *Proc. IEEE International Conf. on Robotics and Automation,* Raleigh, NC, pp. 1–6.

Korn, G. and Korn, T. (1968) *Mathematical Handbook.* McGraw-Hill, New York.

Latombe, J. C. (1991) A fast path planner for a car-like indoor mobile robot. In *Proc. 9th National Conf. on Artificial Intelligence,* Anaheim, CA, pp. 659–665.

Lumelsky, V. and Skewis, T. (1990) Incorporating range sensing in the robot navigation function. *IEEE Trans. Syst., Man, Cyber.* **SMC-20,** 1058–1069.

Lumelsky, V. and Stepanov, A. (1987) Path planning strategies for a point mobile automation moving amidst unknown obstacles of arbitrary shape. *Algorithmica* **2,** 403–430.

Pontryagin, L. S. (1962) *The Mathematical Theory of Optimal Processes.* Interscience, New York.

Sankaranarayanan, A. and Vidyasagar, M. (1990) Path planning for moving a point object amidst unknown

obstacles in a plane. In *Proc. 29th IEEE Conf. on Decisions and Control,* Honolulu, HI, pp. 1111–1120.
Schwartz, J. and Sharir, M. (1983) On the 'Piano Movers' problem. II. General techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.* **4,** 298–351.
Shiller, Z. and Dubowsky, S. (1991) On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Trans. Robotics Automation,* **RA7,** 785–797.
Shiller, Z. and Lu, H. H. (1992) Computation of path constrained time optimal motions along specified paths. *ASME J. Dyn. Syst. Meas., Control* **114,** 34–40.
Volpe, R. and Khosla, P. (1987) Artificial potential with elliptical isopotential contours for obstacle avoidance. In *Proc. 26th IEEE Conf. on Decision and Control,* Los Algeles, CA, pp. 180–185.

APPENDIX—TIME-OPTIMAL SOLUTION FOR A SINGLE-DEGREE-OF-FREEDOM SYSTEM

Consider a single-degree-of-freedom system described by a linear second-order differential equation $\ddot{x} = u(t)$, where $\|u(t)\| \le u_{\max}$ and $u(t)$ is a scalar control function. By introducing new variables $x_1 = x$ and $x_2 = \dot{x}$, the system's motion can be rewritten as $\dot{x}_1 = x_2$ and $\dot{x}_2 = u(t)$, and its behavior described in the *phase space* (x_1, x_2).

The problem to address is as follows: find a piecewise control $\hat{u}(\cdot) \in \{u(\cdot) \mid \|u(\cdot)\| \le u_{\max}\}$ such that the system can be steered from its initial position $(x_1(t_0), x_2(t_0)) = (x_1^0, x_2^0)$ to the origin $(x_1(t_f), x_2(t_f)) = (0, 0)$ in the minimal time t_f. This can be written as

$$\tau(x_1(\cdot), x_2(\cdot), t_f) = t_f \to \infty,$$

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = u(t), \quad \|u(t)\| \le u_{\max};$$

$$x_1(t_0) = x_1^0, \quad x_2(t_0) = x_2^0,$$

$$x_1(t_f) = x_1^f = 0, \quad x_2(t_f) = x_2^f = 0.$$

Below, the optimal solution quantities are indicated by a hat. The following optimal control strategy can be obtained using Pontryagin's maximum principle, which presents a necessary condition for optimality (see e.g. Hocking, 1991). The resulting bang–bang control strategy has just one switching, and thus two options:

$$\hat{u}^+(t) = \begin{cases} -u_{\max} & (t_0 \le t < t_s), \\ +u_{\max} & (t_s < t \le t_f), \end{cases}$$

$$\hat{u}^-(t) = \begin{cases} +u_{\max} & (t_0 \le t < t_s), \\ -u_{\max} & (t_s < t \le t_f), \end{cases}$$

where t_s is the time of switching of the control. To understand the behavior of this system, note that for $u(t) = -u_{\max}$, the solution for the dynamic system $\dot{x}_1 = x_2$ and $\dot{x}_2 = u(t)$ with initial conditions $x_1(t_0) = x_1^0$ and $x_2(t_0) = x_2^0$ is

$$x_1 = -\frac{(x_2)^2}{2u_{\max}} + \frac{(x_2^0)^2}{2u_{\max}} + x_1^0. \tag{A.1}$$

Equation (A.1) indicates that, under the control $u(t) = -u_{\max}$, the motion in phase space is along a parabola of the family (A.1), in the direction of decreasing x_2; hence, from (A.1), $\dot{x}_2 = -u_{\max}$. Similarly, for our control system, the motion along the parabola

$$x_1 = \frac{(x_2)^2}{2u_{\max}} - \frac{(x_2^0)^2}{2u_{\max}} + x_1^0 \tag{A.2}$$

will be defined by $u(t) = u_{\max}$. The motion along the parabolas (A.1) and (A.2) will correspond to increasing values of x_2. As shown above, the origin of the phase space can be reached only by moving either along a half of the parabola $x_1 = (x_2)^2/2u_{\max}$ with control $u(t) = -u_{\max}$, or along $x_1 = (x_2)^2/2u_{\max}$ with control $u(t) = u_{\max}$. The parabolas themselves define the switch curve of the control function $\hat{u}(t)$:

$$x_1 = \begin{cases} -\dfrac{(x_2)^2}{2u_{\max}} & (x_2 \ge 0), \qquad \text{(A.3)} \\ \dfrac{(x_2)^2}{2u_{\max}} & (x_2 \le 0). \qquad \text{(A.4)} \end{cases}$$

The resulting control strategy is therefore as follows.

The optimal control strategy. If in the phase space the initial position is above the switch curve (A.3), (A.4), first move along the parabola (A.1) with control $\hat{u} = -u_{\max}$, until hitting the switching curve, then move with control $\hat{u} = u_{\max}$ to the origin along the switching curve (A.3). If the initial position is below the switching curve, first move with control $\hat{u} = u_{\max}$ toward the switching curve, and, after meeting it, move with control $\hat{u} = -u_{\max}$ to the origin along the switching curve (A.4).

With the optimal control strategy established, we can now find the coordinates (x_1^s, x_2^s) of the switching point, the time t_s of arrival at the switching point, and the optimal time $\hat{t}_f$. There are two possibilities to consider, depending on whether the initial position lies on or off the switching curve.

(i) If the initial position is on the switching curve then the time of arrival at the phase-space origin is $t_f = |x_2^0|/u_{\max}$; the optimal control is $\hat{u} = u_{\max} \operatorname{sign}(x_2^0)$.

(ii) If the initial position is off the switching curve then the time of switching will be the solution of one of the set of equations (A.1), (A.4) or (A.2), (A.3). In particular, if the initial position is above the switching curve, the optimal control is

$$\hat{u}^+(t) = \begin{cases} -u_{\max} & (0 \le t < t_s), \\ +u_{\max} & (t_s < t \le t_f), \end{cases}$$

the switching point is

$$x_1^s = \frac{(x_2^0)^2}{4u_{\max}} + \frac{x_1^0}{2}, \qquad x_2^s = -\sqrt{\frac{(x_2^0)^2}{2u_{\max}} + x_1^0 u_{\max}}, \tag{A.5}$$

and, using the optimal control strategy and the switching points (A.5), the switching time is

$$t_s = \frac{\sqrt{(x_2^0)^2/2u_{\max} + x_1^0 u_{\max}} + x_2^0}{u_{\max}}. \tag{A.6}$$

If the initial position is below the switching curve, the corresponding optimal control, the switching point and the switching time are

$$\hat{u}^-(t) = \begin{cases} +u_{\max} & (0 \le t < t_s), \\ -u_{\max} & (t_s < t \le t_f), \end{cases}$$

$$x_1^s = -\frac{(x_2^0)^2}{4u_{\max}} + \frac{x_1^0}{2}, \tag{A.7}$$

$$x_2^s = \sqrt{\frac{(x_2^0)^2}{2u_{\max}} - x_1^0 u_{\max}}, \tag{A.8}$$

$$t_s = \frac{\sqrt{(x_2^0)^2/2u_{\max} - x_1^0 u_{\max}} - x_2^0}{u_{\max}}. \tag{A.9}$$

Pontryagin's optimality principle is only a necessary condition. For the system at hand, it is also possible to show that the derived control law is sufficient as well—that is, that it indeed produces the absolute minimum time. To prove this fact, we have to show that there is no other control process that would generate motion from the position (x_1^0, x_2^0) to the position $(0, 0)$ in time less than $\hat{t}_f$.

Proceeding by contradiction, assume there is another control process, $(x(\cdot), y(\cdot), t_f)$, with time $t_f < \hat{t}_f$. Define a function $x(\cdot)$ on the interval $[t_f, \hat{t}_f]$ as $x(\cdot) \equiv 0$. Without loss of generality, assume that the initial position is above the

switching curve. Integrating by parts and taking into account the initial conditions gives

$$\int_{t_0}^{t_s} (\tau - s)\ddot{x}(s)\,ds = -t_s\dot{x}(t_0) + x(t_s) - x(t_0).$$

Therefore the function $x(\cdot)$ at the switching time t_s can be presented in the form

$$x(t_s) = \int_{t_0}^{t_s} (t_s - s)\ddot{x}(s)\,ds + \tau\dot{x}(0) + x(t_0).$$

Since $\ddot{\hat{x}}(s) = -u_{max} \le \ddot{x}(s)\ \forall s \in [t_0, t_s]$,

$$\hat{x}(t_s) - x(t_s) = \int_{t_0}^{t_s} (t_s - s)[-u_{max} - \ddot{x}(s)]\,ds \le 0. \quad \text{(A.10)}$$

By analogy, considering the final conditions, the function $x(\cdot)$ at the time t_s is given by

$$x(t_s) = \int_{t_s}^{\hat{t}_f} (s - t_s)\ddot{x}(s)\,ds,$$

and since $\ddot{\hat{x}}(s) = u_{max} \ge \ddot{x}(s)\ \forall s \in [t_s, \hat{t}_f]$,

$$\hat{x}(t_s) - x(t_s) = \int_{t_0}^{t_s} (t_s - s)[-u_{max} - \ddot{x}(s)]\,ds \ge 0. \quad \text{(A.11)}$$

The conditions (A.10) and (A.11) are compatible if $\hat{x}(\tau) - x(\tau) = 0$. From the continuity principle, the condition (A.10) holds only if $\ddot{x}(s) = -u_{max}$, and then $\hat{x}(t) = x(t)$ $\forall t \in [0, t_s]$. By analogy, the equality (A.11) is possible only if $\ddot{\hat{x}}(s) = u_{max}$; then $\hat{x}(t) = x(t)$ $\forall t \in [t_s, \hat{t}_f]$. That is a contradiction, which was obtained from the assumption that there exists an optimal control process with a completion time less than our $\hat{t}_f$. This completes the proof of sufficiency of our optimal control strategy.