# Incorporating Body Dynamics into the Sensor-Based Motion Planning Paradigm. The Maximum Turn Strategy.*

Vladimir J. Lumelsky and Andrei M. Shkel
University of Wisconsin-Madison
Madison, Wisconsin 53706, USA

*Abstract*—**The 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 mass point robot operating in a planar environment with unknown arbitrary stationary obstacles. Given the constraints on the robot's dynamics, sensing, and control means, conditions are formulated for generating trajectories which guarantee convergence and the robot's safety at all times. The approach calls for continuous computation and is fast enough for real time implementation. The robot plans its motion based on its velocity, control means, and sensing information about the surrounding obstacles, and such that in case of a sudden potential collision it can always resort to a safe emergency stopping path. Simulated examples demonstrate the algorithm's performance.**

## I. Introduction

This work studies the effect of body dynamics on robot sensor-based motion planning, with the goal of designing provably correct algorithms for motion planning in an uncertain environment. We consider a mobile robot operating in a planar environment filled with unknown arbitrary stationary obstacles. 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 range finders, 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, 5 or 50 or 1000 – so that normally the robot would see an obstacle far enough to plan a collision-avoiding maneuver.

Besides the usual planning problems of "where to go" and how to provide convergence in view of incomplete information, an additional, dynamic component of planning appears because of the robot's mass and velocity. A step reasonable from the standpoint of reaching the target position – for example, a sharp turn – may not be physically realizable because of the robot's inertia (as at point $P$, Fig. 1). The existing strategies for sensor-based planning usually deal solely with the system kinematics and geometry and ignore its dynamic properties.

Most approaches to automatic motion planning adhere to one of two paradigms which differ in the assumptions about the information available for planning. In the first paradigm, called *motion planning with complete information* (or the *Piano Mover's problem*), perfect information about the robot and the obstacles is assumed, their shapes are presented algebraically, and the motion planning is a one-time off-line operation (see, e.g., [1, 2]). Dynamics and control constraints can be incorporated into this model as well [3], for example by introducing a two-stage planning process, time-optimal trajectories [4, 5].

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 range finder or a vision sensor [6]. By making use of the notion of sensor feedback, this paradigm naturally fits the methodology of control theory, in particular techniques for continuous dynamic on-line processes.

Techniques that ignore the dynamic issues can be called *kinematic*, as opposed to the *dynamic* techniques which take into account the body dynamics. The existing kinematic techniques can be divided into two groups – those for *holonomic* systems and for *nonholonomic* systems [7]. A number of kinematic strategies for holonomic systems originate in maze-searching algorithms [6, 8]; when applicable, they are usually fast, can be used for real time control, and guarantee convergence; the obstacles can be of arbitrary shape. Below we make of use of such algorithms.

To design a provably-correct *dynamic algorithm* for sensor-based motion planning, one needs a single control mechanism – separating it into stages is likely to destroy

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

IEEE International Conference
on Robotics and Automation
0-7803-1965-6/95 $4.00 ©1995 IEEE

convergence. Convergence has two faces – globally it's a guarantee of finding a path to the target if one exists, and locally it's a guarantee 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.

Imagine a runner who starts turning the corner and suddenly sees a heavy large object right on the intended path. Clearly, some quick replanning will take place, almost simultaneous with further sensing and motion execution. The runner's velocity may temporarily decrease and the path will smoothly divert from the object. Or, he might "brake" to a halt along the initial path, and then start a detour path. Note that sensing, planning, and movement are taking place simultaneously and continuously. Unless a right relationship is maintained between the velocity at the time of noticing the object, the distance to it, and the runner's mass, collision may occur. For a bigger mass, for example, better (further) sensing is needed to maintain the same velocity. Also, if obstacles can appear at any time and distance, and if higher velocities are essential, the control algorithm must provide an "insurance" option of a safe stopping at all times.

Although the robot has complete knowledge of the obstacles within its sensing range, because of the computational and convergence problems it would be difficult to address the problem as a sequence of smaller problems with complete information. Our step-by-step calculation algorithm thus operates as a single procedure which (a) places each step on a globally converging collision-free path, while (b) satisfying the robot dynamics constraints.

Assume that the sensing range ("radius of vision") is $r_v$, Fig. 1. The general strategy is as follows: at the moment (step) $i$, the kinematic algorithm identifies an intermediate target point, $T_i$, which lies on a convergent path and, for local path optimization, is far enough from the robot – normally at the boundary of the sensing range. Then, a step is made in the direction of $T_i$, and the process repeats. Because of the dynamics, this step toward $T_i$ may not be possible and needs to be modified.

As mentioned above, sometimes stopping may be the only way to avoid collision, and an assurance is required that at any moment the robot could execute a last resort *stopping path* if needed. Since no information is available beyond the sensing range, the whole stopping path, as computed at a given moment, must lie within that sensing area. Further, since part of the sensing range may be invisible because of obstacles, the stopping path must lie in the visible part of the sensing range. Similarly, if the intermediate target $T_i$ lies on the boundary of the sensing range, the robot needs a guarantee of stopping at $T_i$, even if at the moment it does not intend to do so. Thus, each step is to be planned as the first step of a trajectory which, given the initial position, velocity and control constraints, would bring the robot to a halt at $T_i$.

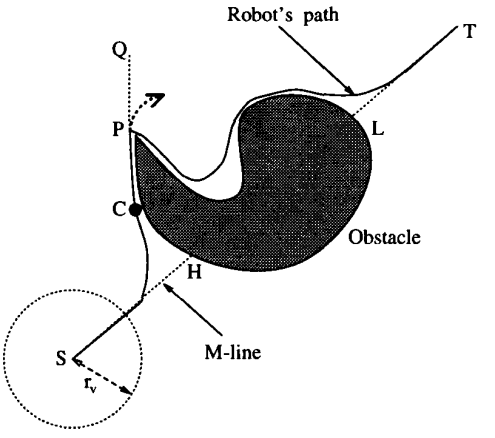


Fig. 1. Example of performance of the (kinematic) VisBug algorithm.

In case of a curved path segment, the control system will attempt a reasonably fast convergence to $T_i$ by trying to align the direction of the robot's motion with the direction toward $T_i$ as soon as possible. That is, if the angle between the current velocity vector and the direction toward $T_i$ is larger than the maximum turn the robot can make in one step, the robot will keep turning in the maximum way possible until the directions align (hence the name *Maximum Turn Strategy*). Once a step is physically executed, new sensing information appears and the process repeats. The procedure also covers a special case where the intermediate target goes out of the robot's sight because of the robot's inertia or because of occluding obstacles.

The model assumed is introduced in Section II, followed by the sketch of the suggested approach in Sec. III. Analysis of the system dynamics appears in Sec. IV; the algorithm and its convergence properties are discussed in Sec. V, and simulated examples of the algorithm performance in Sec. VI. Some details and proofs are omitted due to space limitations.

## II. The model

The environment (the scene) is a plane; it may include a locally finite number of static obstacles. Each obstacle is bounded by a simple closed curve of arbitrary shape and of finite length, such that a straight line will cross it only in a finite number of points. Obstacles do not touch each other (if they do, they are considered one obstacle). Note that the model does not require the number of obstacles be finite.

The robot is a *mass point*, of mass $m$. Its sensors allow it, at its current location $C_i$, to detect any obstacles and the distance to them within its *sensing range* – a disc 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

$\mathbf{V}_i$, coordinates of $C_i$ and of the target point $T$ it is trying to reach, and perhaps of few other points of interest, such as an intermediate target.

The task is to move from point $S$ (start) in the scene to point $T$ (target), see Fig. 1. The robot's means for motion control are two components of the acceleration vector $\mathbf{u} = \frac{f}{m} = (p, q)$, where $m$ is the robot's mass, and $f$ is the force applied. By taking, without loss of generality, mass $m = 1$, we can refer to the components $p$ and $q$ as if they represent control forces, each within its fixed range $|p| \leq p_{max}$, $|q| \leq q_{max}$; $p_{max}, q_{max} > 0$. Force $p$ controls the forward (or backward when braking) motion; its positive direction coincides with the robot's velocity vector $\mathbf{V}$. Force $q$ is perpendicular to $p$ forming a right pair of vectors, and is equivalent to the steering control (rotation of vector $\mathbf{V}$), Fig. 2.

The *M-line* (*Main line*) is the straight line connecting points $S$ and $T$, and 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 M-line point "on the other side" of the obstacle is a *leave point*, $L$.

The control of robot motion is done in *steps* $i = 0, 1, 2, \ldots$. Each step $i$ takes the same time $\tau = t_{i+1} - t_i = const$; its length thus depends on the robot's velocity within the step. Steps $i$ and $i + 1$ start at times $t_i$ and $t_{i+1}$, respectively; $C_0 = S$. We define two coordinate systems (follow Fig. 2): (i) The *world coordinate frame*, $(x, y)$, fixed at point $S$. (ii) The *path coordinate frame*, $(t, n)$, which describes the motion of the mass point at any moment $\tau \in [t_i, t_{i+1})$ within the step $i$. Its origin is attached to the robot; axis $t$ is aligned with the current velocity vector $\mathbf{V}$, axis $\mathbf{n}$ is normal to $\mathbf{t}$. Together with axis $\mathbf{b}$, which is a cross product $\mathbf{b} = \mathbf{t} \times \mathbf{n}$, the triple $(\mathbf{t}, \mathbf{n}, \mathbf{b})$ forms the Frenet trihedron, with the plane of $\mathbf{t}$ and $\mathbf{n}$ being the osculating plane.

## III. The approach

The algorithm will be executed at each step of the robot's path, as a single operation. We take the convergence mechanism of a kinematic sensor-based motion planning algorithm, and add to it controls for handling dynamics. At any moment $t_i$ the robot will maintain an *intermediate target* point $T_i$ within its sensing range, usually on an obstacle boundary or on the M-line. At its current position $C_i$, the robot will plan and execute its next step in the direction $T_i$, then, at $C_{i+1}$ analyze new sensory information and perhaps 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 be designate *temporary intermediate targets* and use them until it can see point $T_i$ again.

In principle, any maze-searching algorithm can be utilized for the kinematic part, so long as it allows an extension to distant sensing. For the sake of specificity, we use here the VisBug algorithm [6], which is based on these two steps (as before, $S$ and $T$ are starting and target positions, $r_v$ the radius of the sensing range, Fig. 1): (1) Walk from $S$ toward $T$ along the M-line until, at some point $C$, detect an obstacle crossing the M-line, say at point $H$; go to Step 2. (2) Using sensors, define the farthest visible intermediate target $T_i$ on the obstacle boundary; make a step toward $T_i$; iterate Step 2 until detect M-line; go to Step 1. In Fig. 1, note that while trying to pass the obstacle from the left, at point $P$ the robot will make a sharp turn.

Safety considerations due to dynamics 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" path that would bring the robot to a halt within the radius $r_v$ if needed. If this rule is violated, at the next step new obstacles may appear in the sensing range, such that collision will become imminent no matter what control is used. This dictates a certain relation between the velocity $\mathbf{V}$, mass $m$, and controls $\mathbf{u} = (p, q)$: if the robot moves with the maximum velocity, the *stop point* of the *stopping path* must be no further that at distance $r_v$ from $C$, $V = \sqrt{2pr_v}$.

In Fig. 1, when approaching point $P$, the robot will designate it as its next intermediate target $T_i$. For awhile $T_i$ will stay at $P$ because no other visible point on the obstacle boundary appears until the robot arrives at $P$. During this time, unless a stopping path is possible at any time, the robot would have to plan to stop at $P$. Otherwise, it may arrive at $P$ with a non-zero velocity, start turning around the corner, and suddenly uncover an obstacle invisible so far, making a collision unavoidable.

Convergence. Because of dynamics, the convergence mechanism borrowed from a kinematic algorithm needs some modification. For example, the robot's inertia may cause it to move so that the intermediate target $T_i$ will become invisible, either because it goes outside the sensing range $r_v$ (as after point $P$, Fig. 1), or due to occluding obstacles (Fig. 3), and so the robot may lose it (and the path convergence with it). The solution chosen is to keep the velocity high and, if the intermediate target $T_i$ does go out of sight, modify the motion locally until $T_i$ is found again.

## IV. Dynamics and collision avoidance

Consider a time sequence $\sigma_t = \{t_0, t_1, t_2, \ldots\}$. Step $i$ corresponds to the interval $[t_i, t_{i+1})$. At 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 VisBug procedure), and vector $\mathbf{V}_i$, the control system calculates the values of control forces $p$ and $q$, applies them to the robot, and the robot executes step $i$, finishing it at point $C_{i+1}$ at moment $t_{i+1}$, with the velocity vector $\mathbf{V}_{i+1}$; then the process repeats.

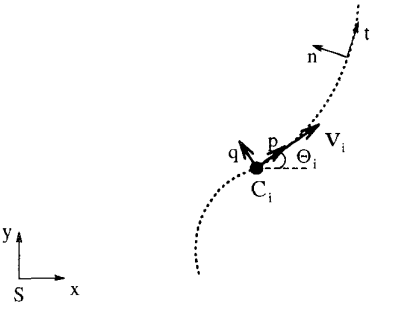


Fig. 2. The dynamic effects of the robot motion are presented with respect to the *path coordinate frame* $(\mathbf{t}, \mathbf{n})$. Obstacle detection analysis and path planning are done with respect to the *world frame* $(x, y)$; its origin is in the starting position $S$.

The following analysis consists of two parts. First, we incorporate the control constraints into the mobile robot's model, and develop the transformation from the moving path coordinate frame, see Section II, to the world frame. Then, the Maximum Turn Strategy is described, the incremental decision making mechanism which controls the computation of forces $p$ and $q$ at each step.

**Transformation from path frame to world frame.** The remainder of this section refers to the time interval $[t_i, t_{i+1})$, and so the index $i$ can be dropped. Denote $(x, y) \in \mathbf{R}^2$ the position of the robot in the world frame, and $\theta$ the (slope) angle between the velocity vector $\mathbf{V} = (V_x, V_y) = (\dot{x}, \dot{y})$ and $x$-axis of the world frame (see Fig. 2). The planning process involves computation of the controls $\mathbf{u} = (p, q)$, which for every step defines the velocity vector and eventually the path, $\mathbf{x} = (x, y)$, as a function of time. The angle $\theta$ between vector $\mathbf{V} = (V_x, V_y)$ and $x$-axis of the world frame is found as

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

Given that the control forces $p$ and $q$ act along the $\mathbf{t}$ and $\mathbf{n}$ directions, respectively, the equations of motion with respect to the path frame are $\dot{V} = p, \quad \dot{\theta} = q/V$. Since the control forces are constant over time interval $[t_i, t_{i+1})$, within this interval the solution for $V(t)$ and $\theta(t)$ becomes

$$\begin{aligned} V(t) &= pt + V^0, \\ \theta(t) &= \theta^0 + \frac{q \log(1 + \frac{tp}{V_i})}{p} \end{aligned} \tag{1}$$

where $\theta^0$ 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 onto the world frame $(x, y)$ using the following vector integral equation:

$$\mathbf{r}(t) = \int_{t_i}^{t_{i+1}} V \mathbf{t} dt \tag{2}$$

Here $\mathbf{r}(t) = (x(t), y(t))$, and $\mathbf{t} = (\cos(\theta), \sin(\theta))$ is a projection of unit vector along the $\mathbf{V}$ direction. After integrating equation (2), we obtain the set of solutions of the form:

$$\begin{aligned} x(t) &= \frac{2p\cos\theta(t) + q\sin\theta(t)}{4p^2 + q^2} V^2(t) + A \\ y(t) &= -\frac{q\cos\theta(t) - 2p\sin\theta(t)}{4p^2 + q^2} V^2(t) + B \end{aligned} \tag{3}$$

where terms $A$ and $B$ are

$$\begin{aligned} A &= x_0 - \frac{{V_0}^2 \, (2\,p\,\cos(\theta_0) + q\,\sin(\theta_0))}{4\,p^2 + q^2}, \\ B &= y_0 + \frac{{V_0}^2 \, (q\,\cos(\theta_0) - 2\,p\,\sin(\theta_0))}{4\,p^2 + q^2} \end{aligned}$$

$V(t)$ and $\theta(t)$ in equations (3), which are directly controlled by the variables $p$ and $q$, are given by equation (1).

In the general case, equations (3) describe a spiral curve. Note two special cases: when $q = 0, p \neq 0$, equations (3) describe a straight line motion along the vector of velocity; when $p = 0, q \neq 0$, they produce a circle of radius $\frac{V_0^2}{|q|}$ centered at the point $(A, B)$.

**The Maximum Turn Strategy .** We now turn to the control law which guides the selection of forces $p$ and $q$ at each step $i$, for the time interval $[t_i, t_{i+1})$. To assure a reasonably fast convergence to the intermediate target $T_i$, those forces are chosen such as to align the direction of the robot's motion with that toward $T_i$ as soon as possible. First, find a solution among the controls $(p, q)$ such that

$$(p, q) \in \{(p, q) : p \in [0, \; -p_{max}], \quad q = \pm q_{max}\} \tag{4}$$

where $q = +q_{max}$ if the intermediate target $T_i$ lies in the left semiplane, and $q = -q_{max}$ if it lies in the right semiplane with respect to the vector of velocity. That is, force $p$ is chosen so as to keep the maximum velocity, and $q$ is chosen on the boundary to produce maximum turn in the right direction.

If, on the other hand, no controls in the range (4) can be chosen, this means that the maximum braking should be applied and that the turning angle should be limited. Then the controls are chosen from the set:

$$(p, q) \in \{(p, q) : p = -p_{max}, \quad q \in [\pm q_{max}, \; 0]\} \tag{5}$$

where the sign in front of $q_{max}$ is chosen as above. Note that the sets (4) and (5) always include at least one safe solution – due to the algorithm's design, the straight-line motion with maximum braking, $(p, q) = (-p_{max}, 0)$ is always safe.

## V. The Algorithm

The algorithm includes three procedures: the *Main body* analyzes the path towards the intermediate target $T_i$; *Define Next Step* chooses the forces $p$ and $q$; *Find Lost Target* deals with the case when $T_i$ goes out of the robot's sight. Also used is a procedure called *Compute* $T_i$, from the VisBug algorithm [6], for computing the next intermediate target $T_{i+1}$ and analyzing the target reachability. Vector $\mathbf{V}_i$ is the current vector of velocity, $T$ is the robot's target.

*Main Body:* The procedure is executed at each step, and makes use of two procedures, *Define Next Step* and *Find Lost Target* (below). It includes the following steps:

- Step 1: Move in the direction specified by *Define Next Step*, while executing *Compute* $T_i$. If $T_i$ is visible do: if $C_i = T$ the procedure stops; else if $T$ is unreachable the procedure stops; else if $C_i = T_i$ go to Step 2. Otherwise, use *Find Lost Target* to make $T_i$ visible. Iterate Step 1.
- Step 2: Make a step along vector $\mathbf{V}_i$ while executing *Compute* $T_i$: if $C_i = T$ the procedure stops; else if the target is unreachable the procedure stops; else if $C_i \neq T_i$ go to Step 1.

*Define Next Step*: the steps below correspond to different cases: Step 1 handles the motion along M-line and a simple one-step turn; Step 2 handles more complex cases of turning:

- Step 1: If vector $\mathbf{V}_i$ coincides with the direction toward $T_i$, do: if $T_i = T$ make a step toward $T$; else make a step toward $T_i$. Otherwise, do: if the directions of $\mathbf{V}_{i+1}$ and $(C_i, T_i)$ can be aligned within one step, choose this step. Else go to Step 2.
- Step 2: If a step with a maximum turn toward $T_i$ and maximum velocity is safe, choose it. Else, if a step with maximum turn toward $T_i$ and some braking is possible, choose it. Else, choose a step along $\mathbf{V}_i$, with no turn and maximum braking, $p = -p_{max}, p = 0$.

*Find Lost Target* is executed when $T_i$ becomes invisible. The last position $C_i$ where $T_i$ was visible is stored until $T_i$ becomes visible again. After losing $T_i$, the robot keeps moving ahead while defining temporary intermediate targets on the visible part of the line segment $(C_i, T_i)$, and continuing looking for $T_i$. If it finds $T_i$, it moves directly toward it, Fig. 3a; otherwise, if the whole segment $(C_i, T_i)$ becomes invisible, the robot brakes to a stop and returns to $C_i$ etc., Fig. 3b. The procedure includes these steps:

- Step 1: If segment $(C_i, T_i)$ is visible, define on it and move toward temporary intermediate targets $T_i^t$, while looking for $T_i$. If current position $C_j = T$, exit; else if $C_i$ lies in the segment $(C_i, T_i)$, exit. Else go to Step 2.

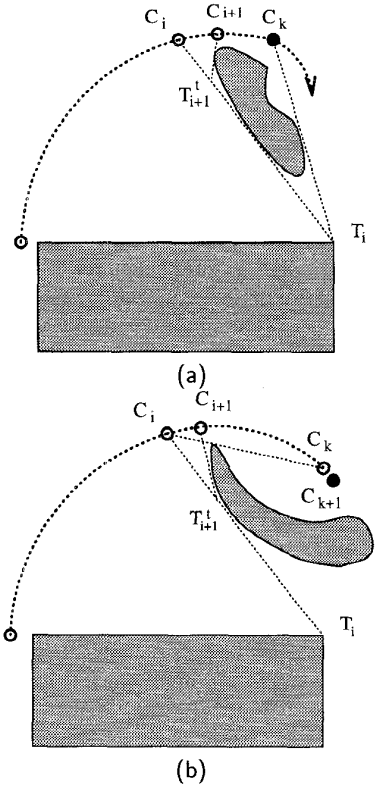


Fig. 3. In this example, because of the system inertia the robot temporarily "loses" the intermediate target point $T_i$.

- Step 2: If segment $(C_i, T_i)$ is invisible, initiate a stopping path and then go back to $C_i$; exit.

**Convergence** To prove convergence, we need to show that (i) at every step the algorithm guarantees collision-free motion and (ii) a path to the target position $T$ will be found if one exists, or the nonreachability of $T$ will be inferred in finite time. Condition (i) can be shown by induction; condition (ii) is assured by the VisBug mechanism [6]. The following statements hold:

**Claim 1** *Under the Maximum Turn Strategy algorithm, assuming zero velocity at the start point, $\mathbf{V}_S = 0$, at every step of the path there exists at least one stopping path.*

**Claim 2** *The Maximum Turn Strategy algorithm guarantees convergence.*

## VI. Examples

Fig. 4a-d illustrate the algorithm's operation in simulated examples. The robot's mass $m$ and controls $p, q$ are the same throughout. Thicker lines show paths generated by the Maximum Turn Strategy Algorithm; thin lines show the corresponding paths produced by the VisBug algorithm.

Examples in Fig. 4a,b correspond to the same radius of vision $r_v$; in Fig. 4b there are additional obstacles which the robot suddenly uncovers at a close distance when turning around corner. Note that in (b) the path becomes tighter, the robot becomes more cautious. A similar pair of examples shown in Fig. 4c,d illustrates the effect of radius of vision: in (c) and (d), $r_v$ is twice that of (a) and (b).

It is interesting to compare the time (the number of steps) the motion took. In Fig. 4a-d thepaths take 221, 232, 193, and 209 steps, respectively. That is, here better sensing (larger $r_v$) results in shorter time to complete the task; more crowded space requires longer time (though resulting perhaps in shorter paths).

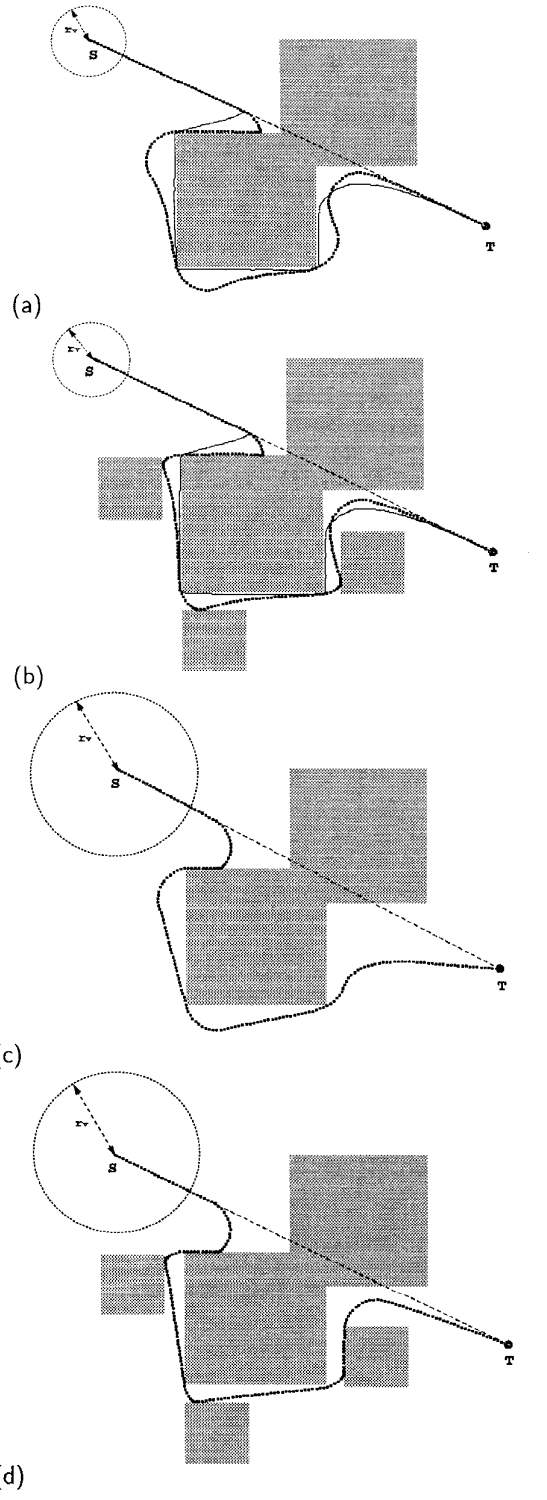


Fig. 4. Simulated examples of the algorithm's performance.

## References


[1] J. Schwartz and M. Sharir. On the "Piano Movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

[2] J. Canny. A new algebraic method for robot motion planning and real geometry. *Proc. 28th IEEE Symposium on Foundations of Computer Science*, 1987. Los Angeles, CA.

[3] Z. Shiller and H.H. Lu. Computation of path constrained time optimal motions along specified paths. *ASME Journal of Dynamic Systems, Measurement and Control*, 114(3):34–40, 1992.

[4] B. Donald and P. Xavier. A provably good approximation algorithm for optimal-time trajectory planning. *Proc. IEEE Intern. Conf. on Robotics and Automation*, May 1989. Scottsdale, AZ.

[5] Z. Shiller and S. Dubowsky. On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Trans. on Robotics and Automation*, 7(6):785–797, 1991.

[6] V. Lumelsky and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(5):1058–1069, September 1990.

[7] D.T. Greenwood. *"Principles of Dynamics"*. Prentice-Hall, New York, 1965.

[8] A. Sankaranarayanan and M. Vidyasagar. Path planning for moving a point object amidst unknown obstacles in a plane: A new algorithm and a general theory for algorithm development. *Proc. 29th IEEE Intern. Conf. on Decision and Control*, 1990. Honolulu, HI.