Numerical Determination of Optimal Non-Holonomic Paths in the Presence of Obstacles

Sudhaker Samuel and S.Sathiya Keerthi
Department of Computer Science and Automation
Indian Institute of Science, Bangalore 560 012, India

Abstract

This paper addresses the problem of numerically finding an optimal path for a robot with non-holonomic constraints. A car like robot, whose turning radius is lower bounded is considered as an example, where the arc length and the change in steering angle are optimized. The car like robot is kinematically constrained and is modelled as a 2 D object translating and rotating in the horizontal plane in the midst of well defined static obstacles. Given the initial and final configurations of the car and a complete description of the obstacles, our procedure directly generates a non-holonomic path as a function of the control variables in an environment of reasonable obstacle clutter. Non-holonomic paths in the midst of more complex obstacle clutter have been generated by identifying grid points on a geometric road map and by applying our procedure between successive grid points.

1 Introduction

This paper concerns path planning for a robot with non-holonomic constraints. These constraints arise due to one or more rolling contacts between rigid bodies and it turns out that the relative velocity of the two points in contact is zero. These constraints render the dimension of the space of achievable velocities smaller than the dimension of the configuration space of the robot.

The motion planning problem without non-holonomic constraints, also referred to as the piano-mover problem, consists of finding a feasible motion for a connected set of rigid bodies amidst obstacles in three dimensional Euclidean space. This problem can be reformulated into the problem of motion of a point in configuration space. A comprehensive overview of the various approaches for solving this problem can be found in [1]. This simple formulation in configuration space does not hold when planning constrained motions, where constraints are nonholonomic in nature.

A car is a typical example of a mechanical non-holonomic system. In the absence of obstacles, it can attain any position and orientation in the plane, rendering the configuration space three dimensional. However, assuming no slip, the velocity of the mid-point between the two rear wheels is always tangential to the car orientation. The space of achievable velocities at any configuration is clearly two dimensional. A successful path planning procedure should construct a collision free path connecting the source and goal configurations in the subset of configurations such that the robot does not graze or intersect obstacles. Important results have been recently reported using tools from differential geometric control theory. Laumond, Canny, Murray, Sastry, Latombe, Lafferiere and Sussmann have initiated investigations in this direction.

Laumond [2, 3] was one of the early investigators of motion planning with non-holonomic constraints in the area of robotics. Later, Fortune and Wilfong [4] developed a decision based algorithm for finding a feasible path satisfying the constraints. Barraquand and Latombe [5] have presented a non-holonomic motion planner which finds a path using potential field methods to systematically search the configuration space, with minimum number of manœuvers. Jacobs and Canny [6, 7] have given a path planning procedure based on canonical trajectories, constructed from Dubin's paths. Murray and Sastry [8] have shown that systems with non-holonomic constraints can be steered between arbitrary states using sinusoids, in the absence of obstacles.

Jacobs, Laumond, and Taïx [9] have developed a two stage planner for a car like robot. The first stage finds a path without accounting for the non-holonomic constraints. In the second stage, this path is approximated by feasible path segments. Reference [10] gives an exhaustive presentation of this procedure. Mirtich and Canny [11] have presented an algorithm which is based on building a one dimensional maximal clearance skeleton to be used as a road map. The metric used to determine the clearance captures information about the non-holonomy of the robot and the robot navigates from start to goal configurations by loosely following the skeleton. An example of a planar two axle car has been cited. Ideas from Reeds and Shepp characterization of shortest paths between any two configurations of the car have been used to design the local planner.

Most of these approaches appear to begin with a holonomic or geometric path and arrive at a non-holonomic path. The numerical approach adopted by us directly arrives at a non-holonomic path. The path finding problem is converted to a finite dimensional non-linear optimization problem. Piecewise constant control inputs serve as optimization parameters. For low and medium complexity obstacle clutter,

1050-4729/93 $3.00

our planner accepts initial estimates of the optimization parameters, descriptions of source and goal configurations of the robot along with the non-holonomic differential constraints and obstacle avoidance state constraints to generate a global collision free non-holonomic path. In case of high complexity obstacle clutter, where the optimization process faces conflicting terms within the objective function, a geometric road map is first drawn. Convenient grid points or sub-goal configurations are identified on the road map. Our planner is used to generate feasible paths between successive grid points. A global non-holonomic path can thus be constructed by putting together these paths between successive grid points. Our procedure has been implemented for a car like mobile robot with two front wheels and two rear wheels navigated by means of two control inputs, the drive velocity and the steer velocity. The car like robot is an example where the kinematic constraints do not restrict the reachable configurations. The kinematics of the car like robot are constrained because the front and rear wheels are only allowed to roll and spin but not slide sideways. Consequently the robot cannot slide sideways or rotate in place. In spite of this, we know that in the real world, a car can be moved to any arbitrary configuration. This controllability property has also been established with rigorous mathematical proofs [13]. Hence the car like robot serves as a good example for implementing non-holonomic motion planning. See [12] for further insight into non-holonomic motion planning.

2 Problem Formulation

2.1 Modelling

Figure 1 shows a four wheeled car, $\mathcal{C}$, modelled as a two dimensional object translating and rotating in the plane. The rear wheels are aligned with the car while the front wheels are allowed to spin about their respective vertical axes. The front and rear pairs of wheels are modelled as single wheels at the mid point of the respective axles. The constraints on the system arise by allowing the wheels to roll and spin but not slip.

The configuration space of the robot is $\mathcal{D} \times \mathcal{S}^1$, where $\mathcal{D}$ is a compact domain of $\mathcal{R}^2$. $\mathcal{D}$ is compact since the range of positions reachable by the robot is bounded. The robot configuration is parameterized by the co-ordinates x and y of the mid point between the two rear wheels and the angle θ between the X-axis of the cartesian frame embedded in the plane and the main axis of the car. The steering angle ϕ measures the orientation of the front wheels with reference to the main axis of the car. The control inputs of the car are the velocity $U_1^* \epsilon \mathcal{R}$ of the front wheels in the direction in which the front wheels are pointing, and the steering velocity $U_2^* \epsilon \mathcal{R}$.

Assuming that there is no slipping, the velocity of the point (x, y) at the midpoint of the rear wheel axle is always parallel to the main axis of the car. Hence the resultant sideways velocity of the wheels is zero. The constraints for the front and rear wheels are formed by writing the expression for the sideways velocity of the wheels and setting it equal to zero. For the rear wheels, this can be written as

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0 \tag{1}$$

For the front wheels, this can be written as

$$\frac{d}{dt}(x + l\cos\theta)\cdot\sin(\theta+\phi) - \frac{d}{dt}(y + l\sin\theta)\cdot\cos(\theta+\phi) = 0 \tag{2}$$

From (1) and (2)

$$\sin\theta\, dx - \cos\theta\, dy = 0 \tag{3}$$

$$\sin(\theta+\phi)\, dx - \cos(\theta+\phi)\, dy - l\cos(\phi)\, d\theta = 0 \tag{4}$$

The system equations may be easily written in a traditional control-theoretic form as

$$\begin{aligned} \dot{x} &= \cos\theta\, U_1 && (a)\\ \dot{y} &= \sin\theta\, U_1 && (b)\\ \dot{\theta} &= \frac{1}{l}\tan\phi\, U_1 && (c)\\ \dot{\phi} &= U_2 && (d) \end{aligned} \tag{5}$$

In this paper, U_1 will be referred to as the drive velocity and U_2 as the steer velocity. The obstacles are assumed to be finite in number. Each obstacle is modelled as a finite union of convex polygons. Each convex polygon is represented by its vertex list given in order. The work space of the robot is also bounded by a finite union of convex polygons.

2.2 Optimal Control Formulation

The problem is to obtain the drive velocity and steer velocity functions so as to realize a connected path between the source and goal, or intermediate goal configurations satisfying the non-holonomic differential constraints defined by the equations in (5) and the obstacle avoidance state constraints imposed by the presence of obstacles. Obstacles in the vicinity are considered in case of path planning between intermediate goal states.

The solution of the above problem is transposed as the solution of the same problem cast as an optimal control problem. It seems a good idea to minimize path length and 'integrated change in the steering angle, ϕ.' The arc length component is represented as

$$J_1 = \int_0^1 \sqrt{\dot{x}^2 + \dot{y}^2}\, dt = \int_0^1 abs(U_1)\, dt \tag{6}$$

The 'change in steering angle' component is written as

$$J_2 = \int_0^1 abs(\frac{d\phi}{dt})\, dt = \int_0^1 abs(U_2)\, dt \tag{7}$$

The problem is to minimize $J_1 + J_2$ subject to satisfying the differential constraints, the state constraints, and the initial and goal states.

Next, the state constraints have to be represented mathematically in the optimization. In order to express the obstacle avoidance condition mathematically

we need to quantify the proximity of a pair of objects represented as convex polygons.

Let P and Q be two convex polygons in $\mathcal{R}^2$. Let $\{p_1 \cdots p_m\}$ and $\{q_1 \cdots q_n\}$ be the ordered vertex lists of polygons P and Q respectively. Let $\bar{p}$ and $\bar{q}$, $\bar{p} \neq \bar{q}$ be specified reference points in the interiors of P and Q respectively. We define the 'expansive distance between P and Q,' $d(P,Q)$ [14] as:

$$d(P,Q) = \min\{\lambda : \bar{p} + \lambda P \ \cap \ \bar{q} + \lambda Q \neq \emptyset\} - 1 \quad (8)$$

It is easy to see that:

(i) $d(P,Q) + 1$ denotes the least expansion (contraction is taken as negative expansion) of P and Q about their reference points, so as to reach a 'just touching' position.
(ii) $P \cap Q = \emptyset$ iff $d(P,Q) > 0$
(iii) int $P \cap$ int $Q \neq \emptyset$ iff $d(P,Q) < 0$
(iv) P and Q are 'just touching' iff $d(P,Q) = 0$.

Thus positive $d(P,Q)$ implies that the polygons do not intersect, and negative $d(P,Q)$ implies that the polygons intersect. The condition, 'P and Q avoid collision,' can now be expressed as $d(P,Q) \geq 0$. If P and Q are represented as convex hulls of points and n denotes the total number of points then $d(P,Q)$ can be computed in time $O(n)$ using a linear programming formulation.

Let: $X = (x, y, \theta, \phi)$ denote the configuration of the car; $\mathcal{O}_1, \ldots, \mathcal{O}_l$ denote the convex polygons that represent the obstacles; $\mathcal{C}(X)$ denote the space occupied by the car while it is in configuration X; $X_o = (x_o, y_o, \theta_o, \phi_o)$ denote the initial configuration of the car; and, $X_f = (x_f, y_f, \theta_f, \phi_f)$ denote the final configuration of the car. Then the optimal control problem is to find $U_1(\cdot)$, $U_2(\cdot)$ and $X(\cdot)$ that solve the following

$$\text{minimize} \ \ J_1 + J_2 \quad (9)$$

subject to:

$$(5) \ \forall t \in [0,1]; X(0) = X_o; \quad (10)$$

$$X(1) = X_f; \ d(\mathcal{C}(X(t)), \mathcal{O}_i) \geq 0 \ \forall i, \ t \quad (11)$$

Given $U_1(\cdot)$ and $U_2(\cdot)$, let $X(t; U_1, U_2)$ denote the solution at t of (5) with $X(0) = X_o$. In this way it is easy to get rid of the unknown, $X(\cdot)$, as well as the constraints in (10).

However, (11), which contains the boundary condition, $X(1) = X_f$ and the point-wise state constraints corresponding to obstacle avoidance are hard constraints on U_1 and U_2, and are difficult to deal with. We handle these constraints by using a penalty approach. Let $p(d)$ denote a penalty function of the type shown in Figure 2. We choose two 'large' constants c_1 and c_2 and replace (9)–(11) by the approximate problem,

$$\min J = \ J_1 + J_2 + c_1 \|X(1; U_1, U_2) - X_f\|^2 + c_2 \sum_{i=1}^{l} \int_0^1 p(d(C(X(t; U_1, U_2)), \mathcal{O}_i)) dt \quad (12)$$

If we select a sequence of (c_1, c_2) values going to infinity and sequentially solve the corresponding problems in (12) then, under reasonable conditions we will arrive at a solution of (11). However, usually a single good choice of (c_1, c_2) and the corresponding solution of (12) leads to a solution that satisfies (10) and (11), and has a good small value for $J_1 + J_2$.

3 Numerical Solution

Application of computer based methods for the solution of the unconstrained optimization problem formulated above requires finite dimensional approximations. The optimization parameters U_1 and U_2 representing the drive and steer velocities have to be appropriately discretized, with trade-offs between complexity, accuracy and computation time. One way is to discretize the entire path into a number of segments. In each segment, U_1 and U_2 can be represented by nth order splines, typically cubic splines, where the spline coefficients become the optimization parameters.

Alternatively, the coefficients of the truncated Fourier series expansion of U_1 and U_2 could be used as optimization parameters. We have chosen U_1 and U_2 to be piecewise constant in each path segment to limit the number of optimization parameters. Varying the number of segments as a trade-off between accuracy and complexity has an effect on computation time. If there are n segments in the path, there will be $2n$ optimization parameters. The problem then reduces to an unconstrained nonlinear minimization problem involving a finite number of variables.

The important task in any optimization routine is the function evaluation. Closed form expressions can be written to evaluate the integrals involved in the definition of J_1 and J_2. The evaluation of the third component in (12) requires the solution of (5). The θ and ϕ trajectories are obtained by integration of 5(c) and 5(d) and are available as closed form expressions. Then x and y are obtained by integrating 5(a) and 5(b) using Simpson's rule.

The fourth component in (12) relates to obstacle avoidance. For non-intersecting configurations, the value of d is immaterial since $p(d)$ is zero. For intersecting configurations, at each t, $X(t; U_1, U_2)$ can be obtained from the interpolants associated with the (x, y, θ, ϕ) trajectories obtained during the solution of (5) mentioned above and d can be obtained by solving a linear program. Polygon P is considered as the obstacle polygon and polygon Q is considered as the robot for applying the obstacle avoidance quantification procedure discussed in section 2. The $\bar{p}$ and $\bar{q}$ are the respective centroids. Given this, the integral can then be numerically computed using Simpson's rule. Computing this fourth component is the most expensive part of computing the objective function J.

Several optimization procedures such as Shanno's extended conjugate gradient(CG) [15], BFGS quasi-Newton(QN) [15], Principal axis(PA) [16] and Nelder and Mead's simplex(SM) [17] were implemented. For conjugate gradient and quasi-Newton, we used an efficient implementation by Shanno called CONMIN. For Principal axis method, we used Brent's PRAXIS, and for simplex, we used Nelder's implementation. Quasi-

	QN	CG	PA	SM
No obstacles	1	1	1	1
Few obstacles < 3	1	3	1	6
Many obstacles $\gg 3$	1	6	10	Failed

Table 1: Relative computation efforts (rough) of various optimization methods.

Newton BFGS proved to be the least expensive in terms of computation time, number of iterations and the number of objective function evaluations. Table 1 gives rough estimate of the relative performance (computational effort) of the various methods, taking quasi-Newton BFGS to be 1. Further discussion and results in this paper refer to BFGS quasi-Newton method.

Choice of initial values for the optimization parameters does have a significant effect on convergence. Suppose $\mathcal{B}$ is the largest circular disc around X_0 such that $\mathcal{B}$ is in the free work space. It was observed that initial estimates of $U_1(\cdot)$ and $U_2(\cdot)$ such that

$$X(t) \in \mathcal{B} \qquad \forall t \in [0,1]$$

resulted in quick convergence. It can be seen that such a path is very unlikely to intersect any of the obstacles. Convergence was much faster with such an initial estimate rather than to start with a path which cuts through obstacles even if the endpoint of that path is close to X_f. The graphic screen output in the former case showed the path winding through from source state to goal state avoiding obstacles at every iteration. But in the latter it proved more expensive to bring a path out of intersection with the obstacles.

Our procedure was found to be successful with a variety of state constraints and in fairly tricky situations in which there were about sixteen cluttered obstacles. Parallel parking and low and medium complexity mazes were some of the problems solved successfully.

Problems with highly cluttered obstacle space were solved by constructing a rough road map such as a Voronoi diagram and identifying convenient intermediate goal states. Mirtich and Canny's [11] road map, which captures the non-holonomy of the problem, can be very useful for generating intermediate goal states. However, the local planner that they use between intermediate points on their maximal clearance skeleton may result in a global path with a large number of cusps. On a global basis, the path reported by us is likely to be much smoother as state continuity and control continuity can be ensured at each intermediate goal state. Further, in our procedure a non-holonomic map or a highly accurate Voronoi diagram was not found necessary as the procedure performed successfully even when successive intermediate points were far off. Hence successful performance as a local planner is well guaranteed.

4 Examples

Our procedure was implemented on a Personal IRIS 4D/20 work station with interactive graphics showing the moving object, the obstacles and the path at the end of each iteration. We tested the planner on a variety of examples using a simulated car like robot with a length to width ratio of 4:1.

Figures 3 shows the solution of a parallel parking problem without state constraints for two different initial estimates of the control parameters; Figure 4 shows the parallel parking problem with state constraints. Figure 5 shows the progress of the algorithm displaying the path returned at intermediate iterations. This is a simple case, with few obstacles where convergence is reached within five quasi-Newton iterations. Figure 6 shows an example with a more cluttered workspace. Here our planner was used to generate a non-holonomic path on a global basis given the source and the goal states. Compare this with Figure 7 which shows the same example as in Figure 6 but in this case our planner is being used as a local planner. Several intermediate goal points were identified on a rough road map and the planner was used to generate non-holonomic paths between these intermediate goal points. The global non-holonomic path was then obtained by putting together these paths. On an aggregate, roughly a 4:1 reduction in computation time was observed when such a road map method was adopted, depending on the nature of the workspace.

5 Conclusion

In this paper we have presented a numerical path planner which directly generates a non-holonomic path. For the car like robot example, the paths generated by the planner have minimal length, minimal change in steering angle and avoid obstacles. For implementing numerical methods to solve the problem, both the work space and the configuration space of the robot was discretized. Further, the discretized control inputs served as the optimization parameters. We have also reported success through the presentation of several illustrative examples. The approach is expected to work for any non-holonomic system. We hope to extend the application to trailer like robots and other higher etage controllable systems.

References

[1] J.C.Latombe, *Robot Motion Planning*, Kluwer Academic Pub., 1990.

[2] J.P.Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," International Conference on Intelligent Autonomous Systems, Amsterdam, pp. 346–354, 1986.

[3] J.P.Laumond, "Finding collision free smooth trajectories for a non-holonomic mobile robot," International Joint Conference on Artificial Intelligence, pp. 1120–1123, 1987.

[4] S.Fortune and G.Wilfong, "Planning constrained motion," STOCS, pp. 445–459, Chicago, 1988.

[5] J.Barraquand and J.C.Latombe, "On non-holonomic mobile robots and optimal maneuvering," 4th International Symposium on Intelligent Control, Albany, 1989.

[6] P.Jacobs and J.Canny, "Planning smooth paths for mobile robots," International Conference on Robotics and Automation, pp. 2–7, 1989.

[7] P.Jacobs and J.Canny, "Robust motion planning for mobile robots," International Conference on Robotics and Automation, 1990.

[8] R.M.Murray and S.S.Sastry, "Steering non-holonomic systems using sinusoids," IEEE Control and Decision Conference, 1990.

[9] P.Jacobs, J.P.Laumond and M.Taïx, "A complete iterative motion planner for a car like robot," Journées de Géométrie Algorithmique, INRIA, June 1990.

[10] P.Jacobs, J.P.Laumond, M.Taïx and R.Murray, "Fast and exact trajectory planning for mobile robots and other systems with non-holonomic constraints," Technical Report # 90318, LAAS, CNRS, Toulouse, France, September 1990.

[11] B.Mirtich and J.Canny, "Using skeletons for non-holonomic path planning among obstacles," International Conference on Robotics and Automation, pp. 2533–2540, May 1992.

[12] Z.Li, R.M.Murray and S.S.Sastry, *Robotics: Manipulation and Planning*, Preprint, December 1990.

[13] A.Bellaïche, J.P.Laumond and P.Jacobs, "Controllability of Car like robots and complexity of the motion planning problem with non-holonomic constraints," The International Symposium on Intelligent Robotics, Bangalore, India, pp. 322–337, 1991.

[14] E.G.Gilbert, Personal communication.

[15] D.F.Shanno and K.H.Phua, "Minimization of unconstrained multivariate Functions," ACM Transactions on Mathematical Software, pp. 618-622, Dec. 1980.

[16] R.P.Brent, *Algorithms for Minimization without derivatives*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1973.

[17] J.L.Kuester and J.H.Mize, *Optimization Techniques with Fortran*, McGraw-Hill Book Co., New York, 1973.

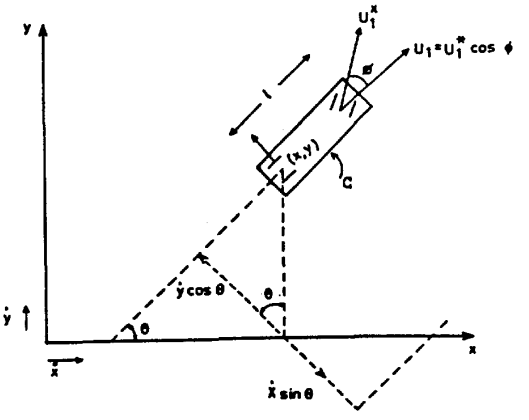


Figure 1: Four wheeled car like robot

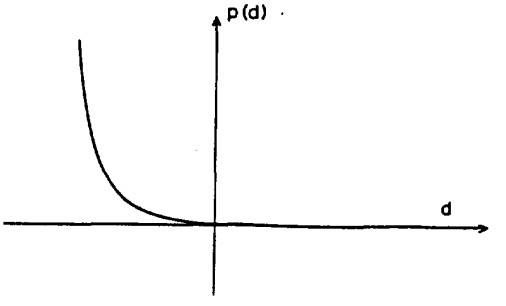
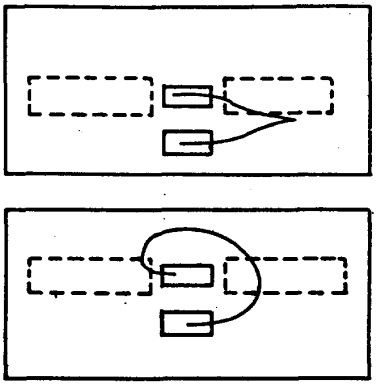
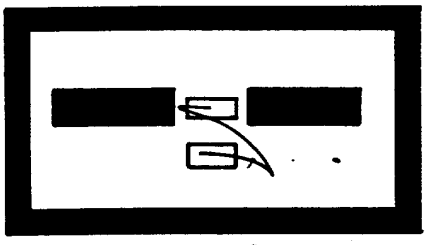


Figure 2: A penalty function for obstacle-avoidance



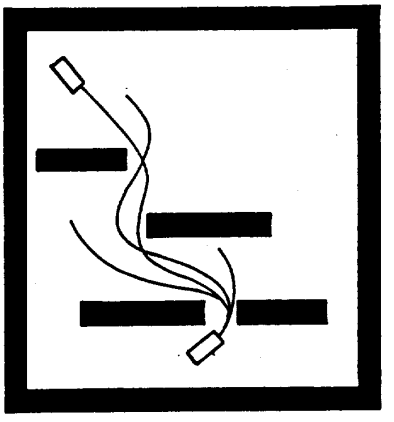
CPU TIME: 4s

Fig 3: Parallel Parking without state constraints.



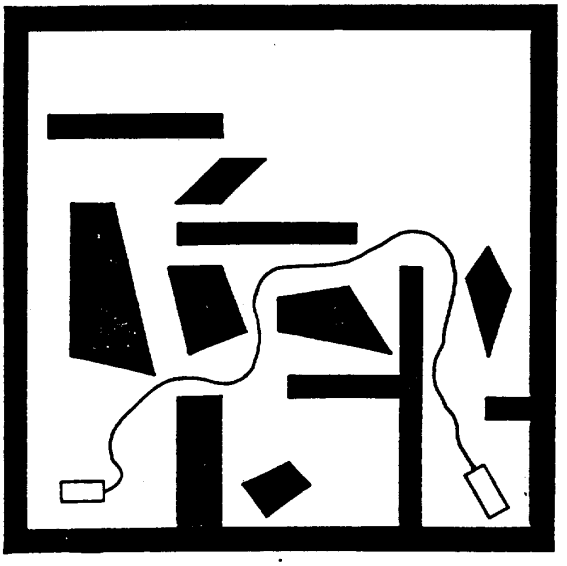
CPU TIME: 8s

Fig. 4. Parallel Parking with state constraints.



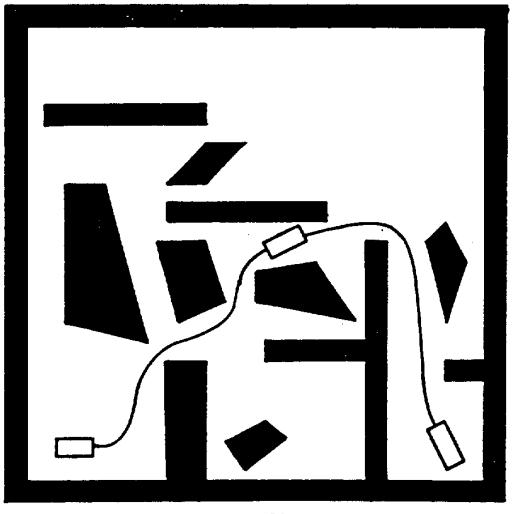
CPU TIME: 14s

Fig. 5. Progress of the algorithm for a simple example



CPU TIME: 885s

FIG. 6: Global path for a more complex example.



CPU TIME: 185s

FIG. 7. Example of figure 6 with one intermediate goal point