

## NAVIGATION OF A CAR-LIKE MOBILE ROBOT USING A DECOMPOSITION OF THE ENVIRONMENT IN CONVEX CELLS

Hubert A. Vasseur, François G. Pin, and Jack R. Taylor<sup>†</sup>

Center for Engineering Systems Advanced Research  
Oak Ridge National Laboratory  
Oak Ridge, TN 37831-6364

### Abstract

Due to their kinematics, car-like mobile robots cannot follow an arbitrary path. Besides obstacle avoidance, the path planning problem for such platforms has to satisfy two additional constraints: a lower bounded radius of turn, and a non-holonomic constraint. When the robot is not circular, precise maneuvering always implies working in the configuration space of the vehicle which, due to the complexity of the problem, involves computer intensive methods, and rarely allows for real time applications. This paper presents a method for rapidly computing the possible maneuvers of car-like robots within convex polygonal cells. In a convex polygonal cell, maneuvering can be completely handled with geometric reasoning and, since only a few boundary configurations have to be checked to avoid collision, the method allows precise computation of the maneuvers without using the whole configuration space. General environments are then described by means of a graph connecting overlapping convex cells. To find a path to a goal, the graph is searched to determine the cells that have to be traversed. Intermediate configurations are then computed inside the intersection of each pair of adjacent cells. Finally, the trajectories generated inside each cell are assembled to produce global collision-free paths in complex environments.

### 1. Introduction

Path planning for an autonomous mobile robot often leads to working in the admissible configuration space, in other words, in the set of collision-free configurations of the robot. However, the method consisting of growing the obstacles by a fixed value so that any arc within the remaining space is a collision-free trajectory, can obstruct some passages, and does not allow for precise maneuvering. In constrained areas, an approach which takes into account the shape and orientation of the robot is the only possible one,

even if it involves much more computer intensive methods, due to the difficulty of representing and searching the configuration space.

Demonstration robots or surveillance robots often have holonomic motion system, i.e., any infinitesimal variation of their configurations is achievable. Therefore, they can perform any trajectory in their admissible configuration space. Sturdy mobile robots, however, often feature kinematic constraints that further complicate the path planning problem. This kind of platforms can only achieve a subset of the set of arcs included in their admissible configuration space.

J. P. Laumond proved that a mobile robot with a non-holonomic constraint such as a car like-robot, remains fully controllable [1]. This means that it is possible to join two configurations located in the same connected component of the admissible configuration space by means of a collision-free trajectory respecting the car-like robot kinematics. Although it is an important theoretical result, the proof given by Laumond does not provide a realistic trajectory because of the number of maneuvers involved.

The aim of this paper is to present a new navigation algorithm for car-like robots. The novelty of the method is that it allows for accurate and efficient maneuvering, while remaining very fast and suitable for real time applications.

The first part of this paper discusses a set of hypotheses that enable us to design trajectories in the admissible configuration space of the robot without requiring prohibitive computations. In the second part of the paper we describe an algorithm generating collision-free trajectories and maneuvers in a convex polygonal cell. Using the previous results, we present in the last section of the paper a global navigation algorithm together with its computer simulation.

### 2. Statement of the Problem

We address the problem of path planning for car-like robots. Among the numerous papers that have investigated the path planning problem, few mentioned the type of robots they consider from a kinematic point of view. This feature can be disregarded for holonomic robots, but many actual robots have a motion system that turns them into non-holonomic vehicles. The usual paradigm of path

<sup>†</sup> Engineering Application Graphics Laboratory,  
Application Support Department, Central  
Engineering, Y-12, Oak Ridge, TN 37831-8201

planning which is the “piano movers problem”, is irrelevant in this case since we have constraints on the type of movements which can be made.

A car-like mobile robot is a front-wheel-steer four-wheel vehicle. We denote by  $L$  the distance between the 2 axes of the wheels, by  $\theta$  the angle between the major axis of the robot and the  $x$  axis of the absolute reference frame, by  $\phi$  the steering angle (i.e. the orientation of the front wheels with respect to the major axis of the robot), by  $M$  the middle of the axle of the rear wheels, and by  $N$  the middle of the front wheels axle (see Fig. 1).

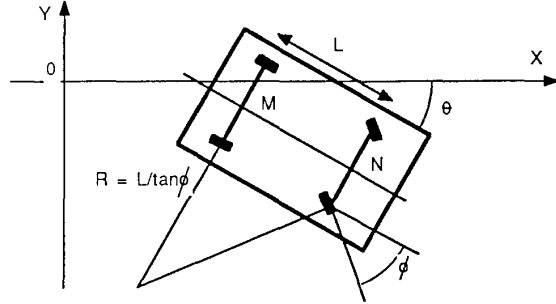


Fig. 1. Radius of curvature of the trajectory of M.

There are two main kinematic constraints for this type of robots. The first one comes from the fact that we assume no slippage of the wheels. Therefore the direction of the velocity of the rear axle is that of the vehicle:

$$\frac{dx_M}{dt} \sin \theta - \frac{dy_M}{dt} \cos \theta = 0 \quad (1)$$

Since

$$\begin{cases} x_N = x_M + L \cos \theta \\ y_N = y_M + L \sin \theta \end{cases} \quad (2)$$

we have

$$\frac{-dx_N}{dt} \sin \theta + \frac{dy_N}{dt} \cos \theta - L\dot{\theta} = 0 \quad (3)$$

Equation (3) involves the velocity of the robot, but since it is not integrable, it does not affect the coordinates of the robot themselves [2]. This constraint is therefore a non-holonomic constraint.

The second constraint concerns the steering angle of the front wheels. If the steering angle is constant, the vehicle moves along a circle. Since the steering angle of a car-like robot is limited, the radius of the circle is always greater than a certain value  $R_{min}$  which is the minimum radius of curvature of any achievable trajectory. Figure 1 provides the expression of the value of  $R$ , the radius of curvature of the trajectory along which the vehicle is moving, as a function of the steering angle. Therefore,

$$|R| \geq R_{min} = \frac{L}{\sin \phi_{max}} \quad (4)$$

As well as the kinematics, the size and shape of the vehicle are also often ignored in many path planning algorithms. The most common approach is to grow the obstacles and to design a trajectory as if the robot were a point. For the case of a circular autonomous robot this method is quite accurate [3]. The accuracy is due to the fact that the map of the free space that is obtained by an isotropic growth of the obstacles by the radius of the robot coincides with the admissible configuration space of this particular robot. However, few actual platforms are circular; the only assumption that will be made in this paper is that the robot can be modeled by a polygon, the vertices of which we denote by  $(R_i)_{1 \leq i \leq n}$  so that  $[R_i, R_{i+1}]$  is an edge of the robot for every  $i$  in  $\{1..n\}$  and  $(R_{n+1} = R_1)$ . Building the configuration space is much more difficult with a polygon shaped vehicle than with a circular shaped one and the representation of the admissible configuration space becomes much more complex [4].

In this paper, we assume the environment to be known and described by polygons. The navigation algorithm requires a decomposition of the environment that will be explained in the third section. Additionally, dynamics and control are not explicitly addressed as the basis of our approach lies in the geometric and kinematic aspect of the problem.

### 3. Car-Like Robot Maneuvering in a Convex Cell

As mentioned in the previous section, working in the admissible configuration space of a non circular robot is extremely numerically intensive and rarely allows for real time applications. However, different attempts have been made in order to simplify the problem. Lozano-Perez proposed to discretize the orientation of the robot, and to compute the configuration space corresponding to each orientation [5]. Once the configuration space has been built this way, it still must be searched in order to find a possible path.

Another approach described by Barraquand and Latombe [6] consists of discretizing the whole configuration space of the robot (i.e. orientation and position), and performing a dynamic search in the discretized configuration space, with the number of maneuvers as a cost function. Both methods have the common drawback of trading off accuracy and computational cost.

We describe below a new method generating the maneuvers required for a car-like robot to join two configurations by a collision-free trajectory, respecting the kinematic constraints of the platform. Our method takes explicit advantage of (i) the partitioning of the environment in convex cells and (ii) the use of predefined trajectories, and thus avoids the time consuming search usually required to compute a path. The difference between the previous methods

mentioned and ours is that by partitioning the environment in convex cells, we can globalize the search for trajectory, and realize the trajectory with large segments and arcs instead of minute increments. This is an important time saving feature. For this first step, the environment is simplified; namely, we assume that the robot has to move inside a polygonal convex cell. We explain in the next paragraph how this result can be generalized to a polygonal environment.

A convex cell is a region of a two-dimensional plane bounded by line segments such that the angle between each adjacent pair of lines is less than  $180^\circ$  (as measured from inside the polygon). Convex cells have the property that any interior point can be reached from any other interior point by a straight line which is entirely in the cell. Thus, inside a convex cell, a mobile robot can travel between any two configurations on a same straight line and with the same orientation by a single straight line motion. This property makes convex cells an interesting representation of the environment for path planning purpose. It can be noticed that in a convex cell, the polygonal description of the robot can be replaced by its convex envelope (the smallest convex polygon containing the robot), without affecting the generality of the algorithm. This may simplify the description of the vehicle, and reduce the execution time of the algorithm.

Let us denote by  $C$  the convex cell in which the robot has to move. The cell  $C$  can be a room or simply an area limited by obstacles.  $C$  is assumed to be convex.

We denote by  $(C_i)_{1 \leq i \leq n}$  the extremities of the segments limiting the cell, so that  $[C_i, C_{i+1}]$  is an edge of the cell for every  $i$  in  $\{1..n\}$  and  $(C_{n+1} = C_1)$ .

The particular geometry of the environment provides the following straightforward result that is used in the design of the trajectory.

*Proposition:*

– A configuration is admissible (i.e. collision-free) if and only if all the vertices of the robot are inside the cell.

*Proof:*

If

$$\forall i \in \{1..m\}, R_i \in C,$$

then

$$\forall i \in \{1..m\}, [R_i, R_{i+1}] \in C \quad (5)$$

because  $C$  is convex. Therefore the configuration is admissible. The reciprocal is obvious.

Because of this result, we can tell if a path is collision-free only by checking that at any moment, the vertices of the robot remain inside the cell.

In such a simple environment, the only problem that can occur is when there is not enough space for the robot to reach its desired orientation. Maneuvering and backing up must then be considered. We described in a previous paper a method to join

two given configurations of a car-like robot [2]. No obstacle was taken into account; therefore, when there is enough space in the cell, this method can be readily used. If the vehicle moves with a constant steering angle, the trajectory performed is a circle. For a given configuration of the vehicle, a circle can be drawn on both sides of the platform, corresponding to the trajectory achieved when the steering angle of the front wheels is maximum. The radius of these circles is  $R_{min}$ , the minimum radius of curvature allowed on any achievable trajectory. Therefore, an optimal trajectory must avoid these circles to avoid additional maneuvering. In order to find a short path providing a continuous orientation for the vehicle, a circle related to the initial configuration and a circle related to the final configuration must be connected by a straight line segment, tangent to the circles. Generally, since there are four pairs of circles, there are sixteen such tangents. Only eight correspond to compatibly oriented circles. This method is particularly interesting in our case because it provides a family of possible trajectories. The shortest one may not be collision-free but another one may be available that remains within the admissible configuration space (see Fig. 2). However, this is the easy case, in which at least one of the eight trajectories is collision-free.

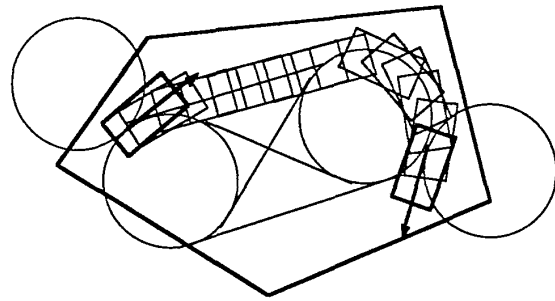


Fig. 2. One of the eight basic trajectories is collision-free.

The method can be adapted to the case in which maneuvering is required, i.e., when it is impossible to find a common tangent to two circles inside the cell. In this case, the circles of minimum radius of curvature are not entirely inside the cell. The vehicle can move only along arcs of circles before colliding with an edge of the cell. The length of the arcs is determined by checking the trajectories of all the vertices of the robot, since it is only with a vertex that the collision can occur. When the steering angle of the wheels is maximum, all the vertices move along circles that have the same center but different radii. Therefore, it is easy to compute the boundary configurations on the circles of minimum radius of curvature (see Fig. 3).

Once the vehicle has reached the extremity of the arc, it has to turn the front wheels to the opposite steering angle and reverse its velocity. Then, it moves along another arc of circle, tangent to the first one, inside the cell (see Fig. 4). The boundary

configuration on this new arc is computed using the same method as for the first arc. Once the extremity of the new arc is reached, the same procedure can be repeated. The method consists of finding a common tangent to an arc related to the initial configuration and an arc related to the final configuration. If a tangent is found that provides the right orientation, a collision-free trajectory has been designed (see Fig. 5): the arcs of circle have been checked for collision, and the segment joins two collision-free configurations on a same straight line. The simplification of the environment enables us to avoid the computation of the whole admissible configuration space. Only the computation of a few admissible configurations is necessary. The search of the path is done by checking first the potential paths requiring maneuvers at the end of the trajectory. This is done because the inaccuracy on the final configuration involved by the maneuvers is increased if the maneuvers occur at the beginning of the trajectory.

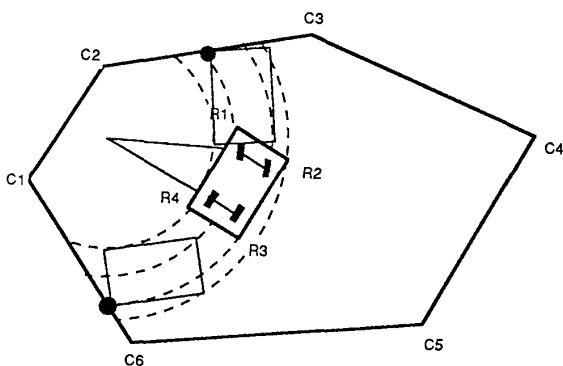


Fig. 3. Computation of the boundary configurations.

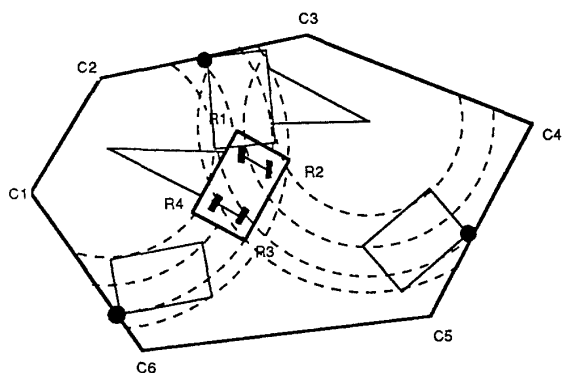


Fig. 4. Computation of the boundary configuration after maneuvering.

This algorithm is particularly suitable for mobile manipulators. Indeed, the problem for these vehicles is that their geometry is variable: the manipulator can be extended outside the frame of the platform. Unlike methods in which the configuration space is completely computed beforehand, changes in the

vehicle geometry can be taken into account. All we have to check is the vertices of the vehicle. By checking also the vertices of the manipulator, we can handle the problem of variable geometry. One can also notice that the algorithm can be parallelized. Sixteen computations can be run simultaneously corresponding to the pairs of circles considered, and the way the robot moves along them (backward or forward). However, the implementation we made of this algorithm on a Silicon Graphics machine proved that the computation of a very complex trajectory (20 maneuvers) takes a few milliseconds; the parallelization is therefore irrelevant except for more complex situations such as combined motion and manipulation.

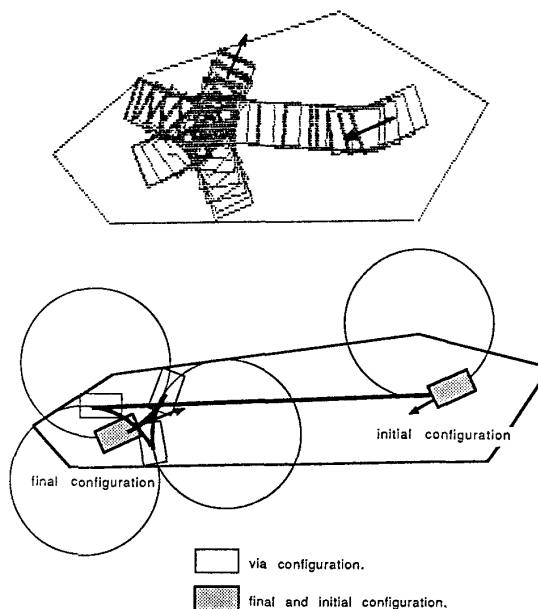


Fig. 5. Trajectory involving maneuvers related to the final configuration.

#### 4. Path Planning for a Car-Like Robot in a General Polygonal Environment

In this section we extend the results obtained for one polygonal cell to a more general environment. Namely, the environment is now assumed to be a polygon containing polygonal obstacles. Such an environment can always be partitioned (not uniquely!) in convex polygonal cells. The partitioning itself may be relevant for some particular applications. Choosing one partitioning over another will then have to do with satisfying additional constraints or finding the extremum of certain functionals related to the problem at hand. We chose the method developed by Crowley [7]. The reason of this choice is that this method features some results of optimality concerning the size of the cells. It provides a partitioning with a reasonable number of cells with reasonable sizes. This

is of great interest since in order to be useful, a cell must be large enough to accommodate the vehicle. The convex cells are defined by an algorithm that divides the navigable space with segments originating at each vertex which is convex with regard to the navigable space. At each convex vertex, two segments are computed by projecting the segments which make up the vertex. A third segment is projected to the nearest visible vertex. The shortest of these three segments is selected as an edge of a cell. The algorithm continues until no convex vertex remains without being associated to a cell boundary (see Fig. 6).

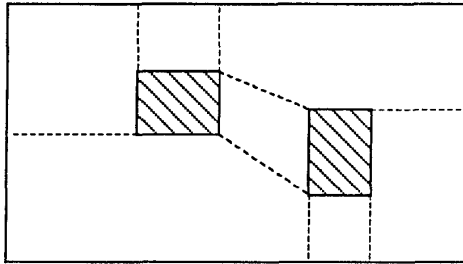


Fig. 6. Example of a decomposition of the environment into maximum convex cells.

Then one can build a graph connecting the adjacent cells through which a path is possible. According to our previous analysis, a convex cell is the most suitable environment to check collisions. Outside of a convex cell, it requires time consuming algorithms, hence the idea of always remaining in a convex cell by using overlapping cells. The problem is to find a collision-free trajectory crossing the boundary edge between two adjacent cells.

Given two connected cells  $C_1$  and  $C_2$  the intersection of which is empty, let us denote by  $C'_1$  (resp  $C'_2$ ) the biggest convex cell included in  $C_1 \cup C_2$  and containing  $C_1$  (resp  $C_2$ ). It is very convenient to place a subgoal in  $C'_1 \cap C'_2$ . We notice that this decomposition allows for and requires nonvoid intersections of adjacent cells. Moreover, the intersections are supposed to be large enough to accommodate the robot (see Fig. 7).

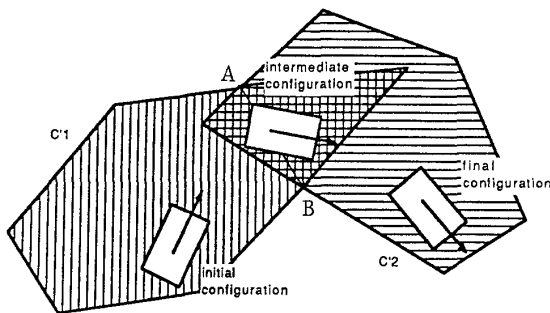


Fig. 7. Intermediate configuration in  $C'_1 \cap C'_2$ .

If this is the case, then the algorithm described in the previous section will move the robot between the initial configuration and the intermediate configuration within the cell  $C'_1$ . The intermediate configuration in this cell is then taken as initial position in the cell  $C'_2$  and the trajectory is continued until the final configuration is reached.

The position of the intermediate configuration is chosen on the line joining the two vertices defining the intersection (A and B on Fig. 7). On this line and within the intersection, the nearest position from either the next boundary edge to be crossed or the final configuration if the trajectory comes to its end, is determined. The chosen orientation is either the direction of the next boundary edge to be crossed or the direction of the final configuration. When the intersection between two adjacent cells is not large enough to accommodate the vehicle, an artificial cell is used. It consists of the intersection between  $C'_1 \cup C'_2$  and the region determined by the perpendiculars to the bisectrices of the two angles generating  $C'_1 \cap C'_2$  (see Fig. 8). The same method to generate intermediate configurations is used to define two more subgoals, corresponding to the intersection between  $C'_1$  and the artificial cell, and between the artificial cell and  $C'_2$ .

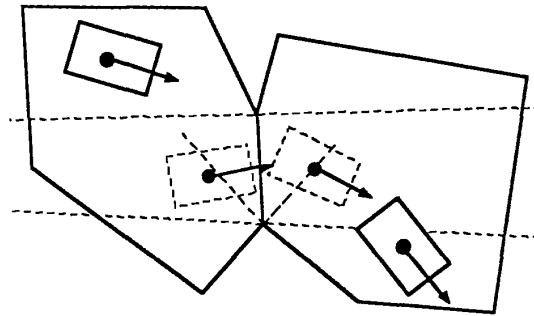


Fig. 8. Use of an artificial cell.

Several remarks are in order:

1) The cells can be shrunk by a tolerance factor in order to make up for unavoidable problems of self location errors of the platform.

2) Given a certain partitioning procedure (like the one described above), one can execute a search of trajectories in order to eliminate the ones that pass through "uncomfortable" cells.

3) If all the trajectories have to pass through "uncomfortable" cells, then more refined ways of assigning the trajectories are necessary, such as creating several artificial cells overlapping two "uncomfortable" cells.

4) Global sufficient conditions (of geometric nature) can be given that ensure passage of the robot through straits. Such a condition requires that the distance between obstacles be larger than the longest diagonal of the vehicle. This condition can obviously be further refined and is currently being investigated.

5) The set of canonical trajectories used in our algorithm (two arcs of circle connected to a straight line) has some interesting properties of optimality regarding the length of the path [8], and the space required. It is particularly suitable for very constrained environments. In a more spacious environment, and without sacrificing the geometric simplicity of the algorithm, it is possible to design trajectories with a continuous curvature [2], that enable the vehicle to have a smoother motion.

The algorithm has been implemented in C in a 3-D simulation on a Silicon Graphics work station. We used a graphics simulation package called IGRIP to simulate a forklift vehicle operations and movements. It is a four-wheel platform equipped with a manipulator. The kinematics of the platform allows a choice between several modes of motion,

including one featuring the characteristics of a car-like platform. The implementation of the algorithm has been organized as follows. The algorithm consists of two different parts: the first one lists the cells that have to be crossed. In order to search the graph we used the Dijkstra algorithm [9]. The second part is a loop on the selected cells; each step calls a recursive procedure that computes the trajectory in the current cell by adding maneuvers as long as no path is found. We carried out a simulation of an autonomous mode of motion. An operator indicates a goal and the vehicle reaches this goal using the trajectory generation algorithm described in this paper. Figure 9 exemplifies the maneuvers inside a convex cell. Figure 10 shows maneuvers using a three cell decomposition of the environment to generate a trajectory.

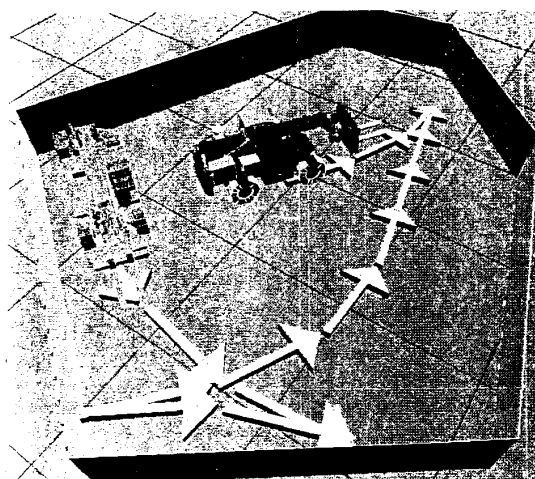
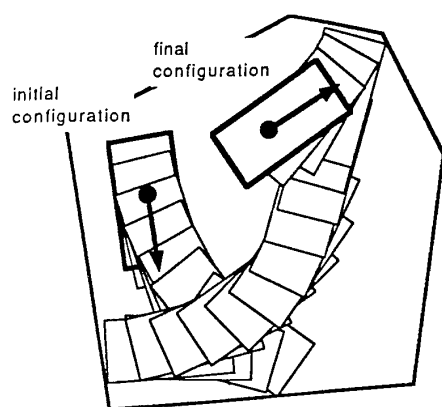


Fig. 9. Forklift vehicle maneuvering in a convex cell.

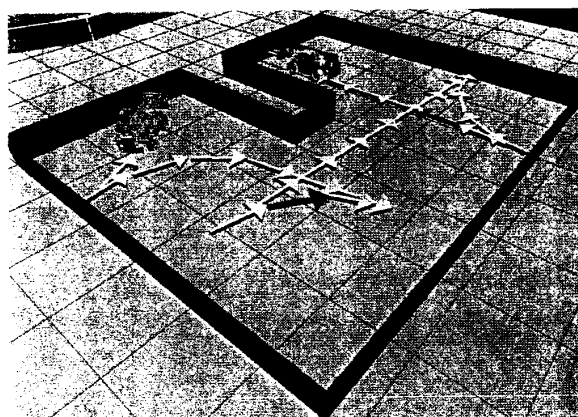
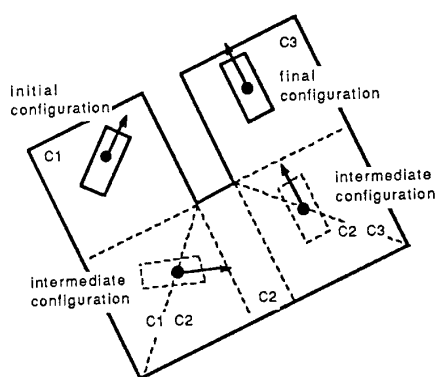


Fig. 10. Example of a trajectory in a 3 cell environment.

## 5. Conclusion

We have developed a new method to provide a fast algorithm for the navigation of car-like robots in a general polygonal environment. The method is based on (i) the decomposition of the environment in convex polygonal cells and (ii) the navigation in such convex polygonal cells by using simple preassigned trajectories. Since these trajectories are formed by long line segments and circle arcs, the preassignment task requires much less time than in previous approaches where the trajectories were a reunion of minute increments.

The method is very consistently geometrical, simple to understand and to implement, and lends itself to various generalizations and refinements: we are currently working on a sensor based navigation version of this method. The fact that our algorithm is not specific to a given vehicle and can be used in different situations (teleoperated mode, autonomous mode in known environment, autonomous mode in unknown environment) makes it even more attractive.

## References

- [1] J. P. Laumond, "Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints," pp. 346-354 in *Proceedings of the International Conference on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, 1986.
- [2] F. G. Pin and H. A. Vasseur, "Autonomous Trajectory Generation for Mobile Robots with Non-Holonomic and Steering Angle Constraints," in *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, Istanbul, Turkey, 1990.
- [3] J. P. Laumond, "Finding Collision-Free Smooth Trajectories for a Non-Holonomic Mobile Robot," in *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milano, Italy, 1987.
- [4] F. Avnaim, J. D. Boissonnat and B. Faverjon, "A Practical Exact Planning Algorithm for Polygonal Object Amidst Polygonal Obstacles," pp. 1656-1661 in *Proceedings of the IEEE International Conference on Robotics and Automation*, Philadelphia, Penn., 1988.
- [5] T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Trans. Comp.*, C-32, 108-120 (1983).
- [6] J. Barraquand, J. C. Latombe, "On Non-Holonomic Mobile Robots and Optimal Maneuvering," in *Proceedings of the 4th IEEE International Symposium on Intelligent Control*, Albany, N.Y., 1989.
- [7] J. L. Crowley, "Navigation of an Intelligent Mobile Robot," *IEEE J. Rob. Auto.* (March 1985).
- [8] P. Jacobs and J. Canny, "Planning Smooth Paths for Mobile Robots," pp. 2-7 in *Proceedings of the IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, 1989.
- [9] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley Publishing Co., 1983.