A collision checker for car-like robots coordination

Thierry Simeon, Stéphane Leroy, Jean-Paul Laumond

IEEE Int. Conf. on Robotics and Automation
Leuven, May 1998

A collision checker for car-like robots coordination

T. Siméon S. Leroy J.P. Laumond

LAAS-CNRS
7, avenue du Colonel-Roche
31077 Toulouse Cedex - France
{nic,sleroy,jpl}@laas.fr

Abstract: *This paper presents a geometric algorithm dealing with collision checking in the framework of multiple mobile robot coordination. We consider that several mobile robots have planned their own collision-free path by taking into account the obstacles, but ignoring the presence of other robots. We first compute the domain swept by each robot when moving along its path; such a domain is called a* trace. *Then the algorithm computes the coordination configurations for one robot with respect to the others, i.e. the configurations along the path where the robot enters the traces of the other robots or exits from them. This information may be exploited to coordinate the motions of all the robots.*

1 Introduction

This paper presents a geometric algorithm dealing with collision checking in the framework of multiple mobile robot coordination. Path planning for multiple robots has been addressed along two main axis: centralized and decentralized approaches.

In the centralized approaches the search is performed within the Cartesian product of the configuration spaces of all the robots. While the problem is PSPACE-complete [3], recent results by Svestka and Overmars show that it is possible to design planners which are efficient in practice (up to four mobile robots) while being probabilistically complete [10]: the underlying idea of the algorithm is to compute a probabilistic roadmap constituted by elementary (nonholonomic) paths admissible for all the robots considered separately; then the coordination of the robots is performed by exploring the Cartesian product of the roadmaps.

In [1], Alami reports experiments involving ten mobile robots on the basis of a fully decentralized approach: each robot builds and executes its own plan by merging it into a set of already coordinated plans involving other robots. In such context, planning is performed in parallel with plan execution. At any time, robots exchange information about their current state and their current paths. Geometric computations provide the required synchronization along the paths. If the approach is not complete (as a decentralized scheme), it is sufficiently well grounded to detect deadlocks. Such deadlocks usually involve only few robots among the fleet; then they may be overcome by applying a centralized approach locally.

In this paper we propose an algorithm to solve the following problem. Several mobile robots plan their own collision-free path by taking into account the obstacles and ignoring the presence of other robots. The domain of the plane swept by each robot when moving along its path called a *trace*. The objective is to compute the coordination configurations for one robot with respect to the others, i.e., the configurations along the path where the robot enters the traces of the other robots or exits from them. This information may be exploited to coordinate the motions of all the robots.

Swept volume computation has already been proposed for collision detection along a given path (see [4] for a survey). For instance, in [2] the authors propose an approximated method based on an approximation of the trace by bounding boxes.

The algorithm presented in this paper does not make any approximation of the traces. It is dedicated to the following case: the paths of the robots are sequences of straight line segments and circular arcs of angle lesser than π (e.g., Reeds and Shepp paths [9]). This hypothesis is realistic from a path planning point of view: it holds for any mobile robot (the existence of a collision-free feasible path is equivalent to the existence of such a special sequence); moreover numerous existing nonholonomic path planners compute solu-

tions of this type (eg. [6, 5, 7, 8]).

For a polygonal robot, the trace is a domain bounded by arcs of a circle and straight line segments, i.e. a so-called *generalized polygon*. In the following developments we restrict ourselves to a rectangular robot; the extension to a polygonal robot would need technical and tedious developments in the computation of the traces that would not add any value to the approach.

In the following section we present the geometric structure of the traces. Then, we give an algorithm to compute the collision subpaths with a generalized polygon. The last section presents experimental results and we conclude on the interest of the algorithm with respect to coordinated motion planning.

2 Computing robot traces

Computing the trace swept by a rectangular robot along a path segment is obvious; it simply corresponds to the rectangle elongated by the length of the path segment.

Therefore, this section only concerns the case of arc paths. Given a rectangular robot, a rotation center c and a rotation angle θ, we want to compute the generalized polygon swept during the motion.

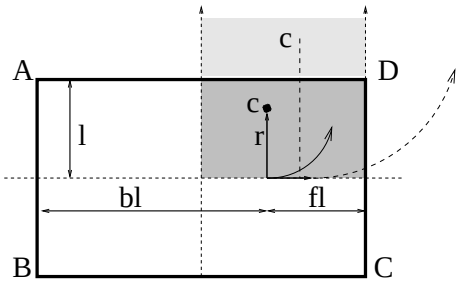

Figure 1: Canonical case: the rotation center c belongs to one of the two grey regions

For symmetry reasons, the presentation is limited to the following canonical case: the robot moves along a *forward left* motion and $bl > fl$ with fl (forward length) and bl (backward length) as defined by Figure 1 (also remind that the angle θ is smaller than π). Also note onto the figure the two situations that may occur depending whether the rotation center c is located inside or outside the robot.

2.1 Rotation center inside the robot

For a small θ the trace contains four circular arcs (centered at c) and eight segments (see Fig. 2-a). Each arc goes from a vertex of the rectangle to the same vertex rotated by θ. Consecutive arcs (eg. $A \xrightarrow{a} A^\theta$ and

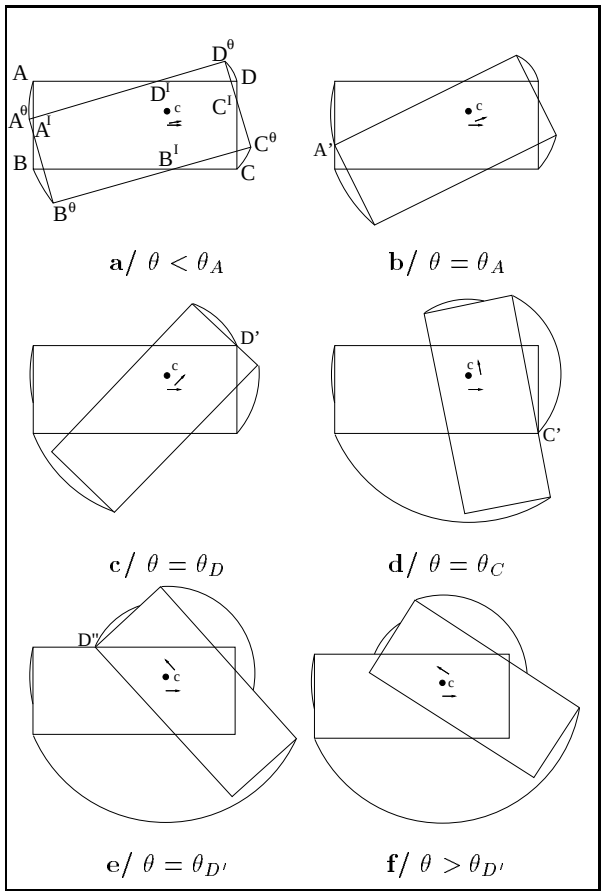

Figure 2: Trace evolution and critical angles

$B \xrightarrow{a} B^\theta$) are connected by two segments (eg. $A^\theta \xrightarrow{s} A^I \xrightarrow{s} B$). The full description of the trace obtained in this case, is given by the outside sequence of Figure 3.

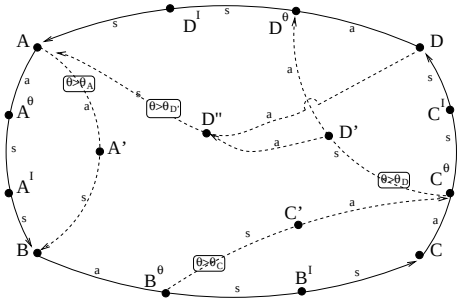

Figure 3: The trace is obtained by following the sequences associated to the critical values

Let us now increase the angle θ. Figures 2-b to 2-f show the modifications onto the trace: the arcs increase up to some critical θ-value. The first critical value θ_A (Fig. 2-b) occurs when A^θ crosses the edge

AB (at position $A' = A^\theta = A^I$). For θ greater than θ_A, the sequence $A \xrightarrow{a} A^\theta \xrightarrow{s} A^I \xrightarrow{s} B$ which initially connected the trace's vertices A and B is replaced by the shorter sequence $A \xrightarrow{a} A' \xrightarrow{s} B$. The rest of the trace remains unchanged until the next critical value θ_D is reached (when D crosses the edge $C^\theta D^\theta$ at position D'), reducing in a similar way the sequence between vertices C^θ and D^θ. The next encountered critical value θ_C (Fig. 2-d) modifies the sequence between vertices B^θ and C^θ and the last critical value $\theta_{D'}$ (Fig. 2-e) occurs when D^θ crosses the edge DA at position noted D''. After this fourth critical value, the shape of the trace remains unchanged with only eight parts $(A \xrightarrow{a} A' \xrightarrow{a} B \xrightarrow{a} B^\theta \xrightarrow{s} C' \xrightarrow{a} C^\theta \xrightarrow{s} D' \xrightarrow{a} D'' \xrightarrow{s} A)$.

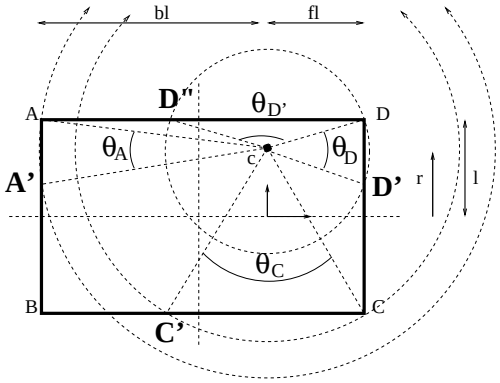

Figure 4: The critical angles $(\theta_A, \theta_C, \theta_D, \theta'_D)$ and the associated critical points (A', C', D', D'')

The four critical values only depend on the robot geometry. Figure 4 explains how their expression can be easily deduced from the parameters r, l, bl and bl. Note however that their order may change depending on the location of the rotation center c. More precisely, one can establish that the order only depends on the relative situation of the two intervals $[fl, bl]$ and $[r\text{-}l, r\text{+}l]$. Figure 5 resumes the order associated to each such case.

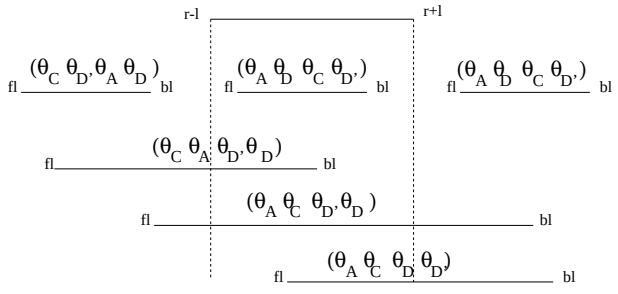

Figure 5: The sorted critical values

This analysis allows us to derive a very simple algorithm for the trace computation: once the ordered set of critical θ-values is determined, the trace is directly obtained from the diagram of Figure 3. The choice of the relevant sequences is made by comparing θ to each critical value.

2.2 Rotation center outside the robot

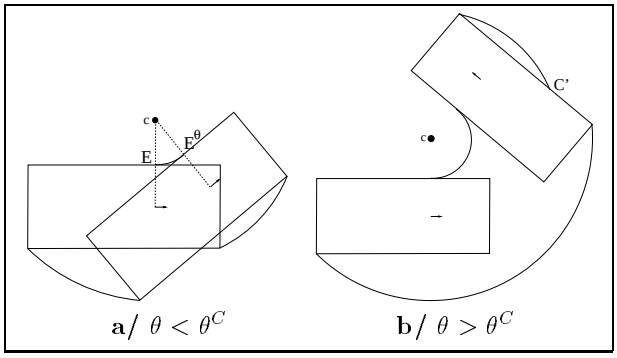

Figure 6: Only two different traces

As illustrated by Figure 6, this case introduces a concave arc (issued from the trace swept by the edge DA) which connects the two points $E = (0, l)$ and E^θ (point E rotated by θ). This case is however much simpler than the previous one since the arcs swept by the vertices A and D always remain interior to the trace. Therefore, only the critical value θ_C may occur in this case.

3 Collision subpaths

In this section, we present an algorithm which will be used in section 4 to compute the coordination configurations for a robot with respect to the paths of other ones.

The inputs of the algorithm are: an edge $e = [A, B]$ moving along a path γ (straight-line segment or circular arc) and a generalized polygon PG.

The output is the set of *collision subpaths* for which the moving edge e collides with PG. Let $s \in [0, 1]$ be the curve length along path γ and $e(s)$ be the edge placed at position s along γ. The collision subpaths γ_{coll} are represented by the ordered set of *collision intervals* S_i defined by the $s-$values at which the i^{st} collision either begins or stops between $e(s)$ and PG.

3.1 Case of segment subpaths

Let us illustrate the algorithm on the canonical example of figure 7. The figure shows the collision subpaths that should be produced by the algorithm. The bolded curves of PG's contour represent the only

curves that need to be considered for this computation.

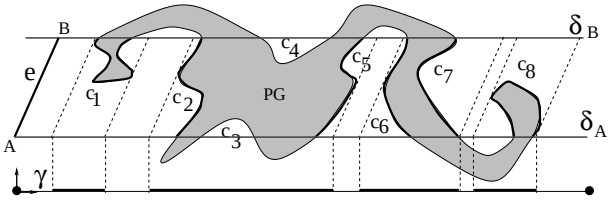

Figure 7: Collision subpaths of a generalized polygon

Let us consider the points resulting from the intersection of PG's contour with the two lines δ_A and δ_B swept by the edge's endpoints along γ. The contour of PG can be decomposed into elementary parts (i.e. sequences of curves) connecting two such points. Obviously, we only need to consider the parts that are interior to the domain $\mathcal{D}$ lying between δ_A and δ_B. Also note that these parts have to be treated differently according to their intersection with δ_A and δ_B. Some parts define a start point (eg. $c2, c6$), an end point (eg. $c5, c7$) or both endpoints (eg. $c1, c8$) of a collision subpath, while others do not produce any endpoint (eg. $c3, c4$).

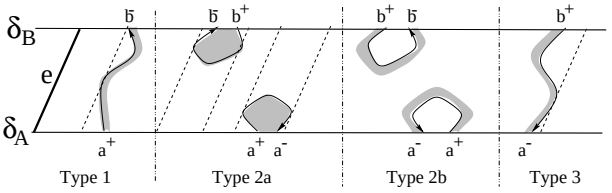

Figure 8: The three elementary cases

Figure 8 shows how a part resulting from PG's decomposition can be classified according to the labels of its start/end points. Each intersection point is labeled as follows: when PG's contour (oriented clockwise) enters into the domain $\mathcal{D}$, the point is labeled a^+ or b^+ according to its location onto δ_A or δ_B. Labels a^- or b^- are similarly assigned to the points at which the contour exits domain $\mathcal{D}$. Consider now for example a part starting at a point labeled a^+. This part either ends at a point b^- (type 1) or at a point a^- (type 2a and 2b). In the first case, the part corresponds to the beginning of a collision subpath. The end of the collision subpath will be given by the next $b^+ \rightarrow a^-$ part (type 3) encountered while following the contour. In the second case, both endpoints belong to δ_A and the part possibly generates a complete collision subpath when the start a^+ is located to the left of the exit endpoint a^- (type 2a). In the other case (type 2b), the part does not need to be considered since the corresponding collision subpath is necessarily included into a larger one obtained from other parts of the contour.

Since each part corresponds to a connected sequence of segments and circular arcs, one can easily check that the collision subpaths endpoints only occur at some points of the sequence (a vertex x_v or a tangency point x_t between a circular arc and the edge e). Let $s(x)$ be the s-value along γ at which such point x belongs to $e(s)$. A start (resp. end) point is obtained by considering the minimal (resp. maximal) value of all the $s(x)$ computed along the part.

The algorithm is first initialized by following PG's contour (from any starting point), until a first intersection x_1 with δ_A or δ_B is found. Then the algorithm continues to loop over the curves of PG until the next intersection x_2. Between x_1 and x_2, it iteratively records the extremal values of the $s(x)$ computed at the encountered vertices or tangent points. When x_2 is found, the collision subpath of the part is obtained from extremal values, according to the labels of x_1 and x_2. The algorithm next considers the part starting at point x_2, and continues until PG's contour has been completely scanned. At the end, some of the produced collision intervals may intersect. Therefore an additional step is required to compute their union. The algorithm returns the ordered set of non overlapping intervals included into the interval $[0, 1]$ (i.e. the collision subpaths of γ).

3.2 Case of arc subpaths

The principle of the algorithm remains similar to the one described above for the case of segment subpaths.

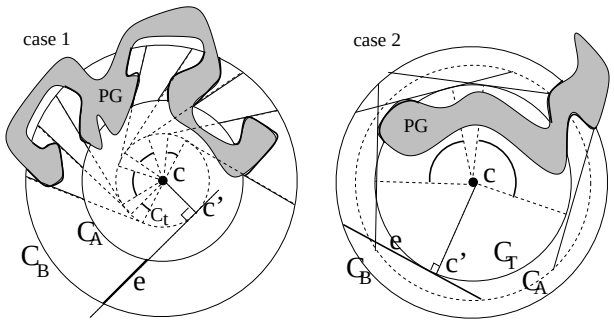

Figure 9: The two cases of arc subpaths

Let us consider the trace swept by the edge e along an arc of a circle γ with radius r and centered at c. Two situations occur depending on the relative position between e and c (Fig. 9). When the orthogonal projection c' of c onto the line supporting e does not belong to the edge (case 1), the domain $\mathcal{D}$ is limited

by an inner circle $\mathcal{C}_A$ and an outer circle $\mathcal{C}_B$, both centered at c and going through one of both edge's endpoints. When c' belongs to edge e (case 2), the inner circle corresponds to the circle $\mathcal{C}_t$ that is tangent to the edge, and the outer circle remains $\mathcal{C}_B$. The figure also shows for each case, the relevant parts of PG's contour. These parts are limited by the intersections with domain $\mathcal{D}$ and are labeled as explained in section 3.1.

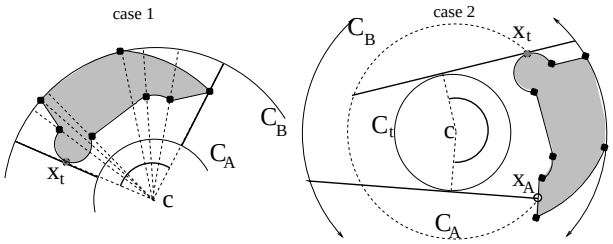

Figure 10: Relevant points considered by the algorithm

For a given part, Figure 10 shows that different sets of points have to be considered: the vertices x_v (black points onto the figure) and the tangency points x_t. Moreover, case 2 requires to consider additional points x_A resulting from the intersection between $\mathcal{C}_A$ and the part.

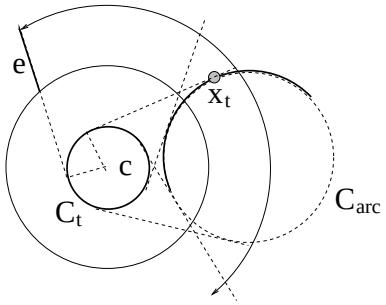

Figure 11: Tangent points between the moving edge e and a circular arc

Remark: The tangency points x_t between the moving edge e and a circular arc, are obtained by computing the common tangents between circle C_t and the support circle of the arc (see Figure 11). Points x_t are the tangent points of C_{arc} which also belong to the arc and to the domain $\mathcal{D}$ swept by the edge.

3.3 Complexity of the algorithm

The algorithm takes $O(n)$ to loop over the n curves of PG's contour and to compute its decomposition into k parts connecting the $2k$ intersections with domain $\mathcal{D}$. At most one collision interval is computed for each part. Therefore, $O(k)$ (possibly) overlapping intervals are produced at the end of the loop. The algorithm then computes the sorted union of these $O(k)$ intervals; its overall complexity is therefore $O(n + klogk)$.

4 Application to car-like robot coordination

The algorithms introduced in both previous sections are now applied in the framework of the motion coordination problem.

Let us consider two paths γ_1 and γ_2 independently planned by two car-like robots. Our objective is to compute the coordination configurations for the first robot with respect to the trace of the second one (i.e. the configurations where the first robot enters or exits the trace of the second one).

The trace of the second robot along γ_2 is first computed (by using the algorithm of Section 2). This trace is a generalized polygon.

The path γ_1 is a sequence of straight line segments and arcs of a circle. For each element of the sequence we have to compute the enter/exit configurations with respect to the trace of the second robot. The robots being rectangles, we apply the algorithm of Section 3 to the four edges of the first robot[1]. Each application of the algorithm gives rise to collision subpaths along γ_1. The union of the all collision subpaths provides the final solution, i.e., the set of collision-free configurations along γ_1 with the trace of the second robot.

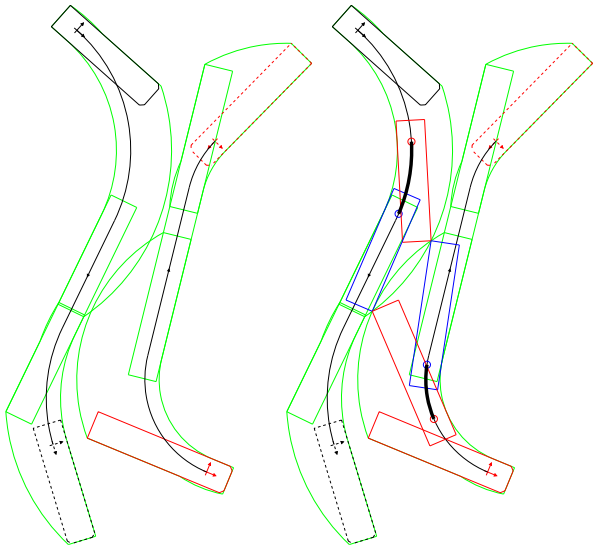

Figure 12: A first example: **a/** the paths γ_1 and γ_2 and **b/** the computed collisions subpaths

[1] This assumes that the trace of the second robot is not small enough to be included into the first robot at any point of γ_1!...

Figures 12-13-14 show some examples computed by the algorithm. For the first example, the left figure shows two paths γ_1 and γ_2 and the traces swept by each robot along these paths. The collision subpaths are represented in bold onto the right figure which also shows the robots placed at the extremities (the coordination configurations) of these subpaths. The other figures show two examples involving three robots.

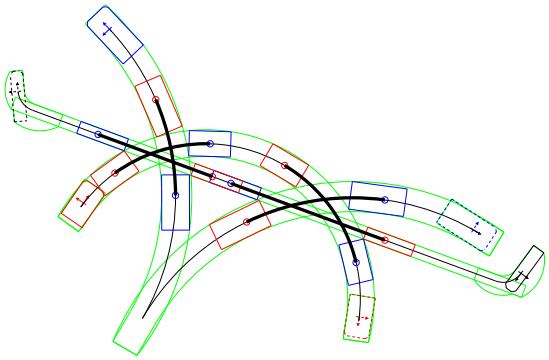

Figure 13: Coordination of three robots of different size

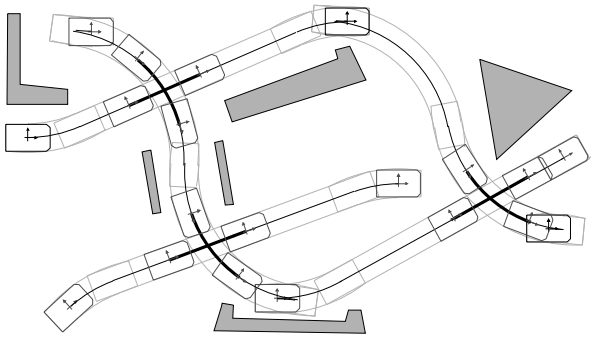

Figure 14: Coordination of three robots along their planned collision-free path

5 Conclusion

The collision checker algorithm presented in this paper may be used in several frameworks. When all the start and goal configurations of each robot are outside the traces of all the other ones, there exists a coordinated motion allowing each robot to execute its own motion in a coordinated way. The continuous nature of the motion coordination problem is transformed into a graph search working from the coordination configurations computed by our algorithm (see [1] for an example of application).

Consider now the cases where a start or goal configuration of a robot belongs to the trace of another robot; these cases are easily detected by our algorithm. We can imagine a (random) method computing a configuration (replacing the start or goal configuration) outside all the traces. This new configuration appears as an intermediate goal to reach before (or after) executing the motions of all the other robots. Such an operation may be repeated on all the robots generating deadlock situations.

References

[1] R. Alami, "Multi-robot cooperation based on a distributed and incremental plan merging paradigm," in *Algorithms for Robotic Motion and Manipulation, WAFR'96*, J.P. Laumond and M. Overmars Eds, A.K. Peters,1997.

[2] A. Foisy and V. Hayward, "A safe swept volume method for collision detection," *6th International Symposium of Robotics Research*, Pittsburg, USA, Oct 1993.

[3] Hopcroft and Wilfong, "Reducing multiple object motion planning to a graph searching", in *SIAM Journal of Computing* , 15 (3), 1986.

[4] P. Jiménez, F. Thomas and C. Torras, "Collision Detection Algorithms for Motion Planning", in *Robot motion Planning and Control*, J.P. Laumond Ed., Lecture Notes in Control and Information Science, 1998.

[5] J.C. Latombe, "A Fast Path Planner for a Car-Like Indoor Mobile Robot," in *Ninth National Conference on Artificial Intelligence, AAAI*, pp. 659–665, Anaheim, CA, July 1991.

[6] J.P. Laumond, P. Jacobs, M. Taïx, and R. Murray, "A motion planner for nonholonomic mobile robot", *IEEE Trans. on Robotics and Automation*, 10 (5), 1994.

[7] B. Mirtich, and J. Canny, "Using skeletons for nonholonomic motion planning among obstacles," in *IEEE Conf. on Robotics and Automation*, Nice, France, 1992.

[8] M. Overmars and P. Svestka, "A probabilistic learning approach to motion planning", in *Algorithmic Foundations of Robotics, WAFR'94*, K. Goldberg *et al* Eds, A.K. Peters, 1995.

[9] J. A. Reeds and R. A. Shepp, "Optimal paths for a car that goes both forward and backwards," *Pacific Journal of Mathematics*, 145 (2), 1990.

[10] P. Svestka and M. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps", in *IEEE Conf. on Robotics and Automation*, Nagoya (Japan), May 1995.