Path coordination for multiple mobile robots: a resolution-complete algorithm

Thierry Simeon, Stéphane Leroy, Jean-Paul Laumond

Path Coordination for Multiple Mobile Robots: A Resolution-Complete Algorithm

Thierry Siméon, Stéphane Leroy, and Jean-Paul Laumond, *Member, IEEE*

***Abstract*—This paper[1] presents a geometry-based approach for multiple mobile robot motion coordination. The problem is to coordinate the motions of several robots moving along fixed independent paths to avoid mutual collisions. The proposed algorithm is based on a bounding box representation of the obstacles in the so-called coordination diagram. The algorithm is resolution-complete but it is shown to be complete for a large class of inputs. Despite the exponential dependency of the coordination problem, the algorithm efficiently solves problems involving up to ten robots in worst-case situations and more than 100 robots in practical ones.**

***Index Terms*—Coordination diagram, mobile robots, multiple robots, path coordination.**

I. Introduction

THIS paper addresses the following problem: consider n mobile robots sharing the same workspace and planning their paths independently; given n such paths, we want to devise an algorithm deciding whether coordinated motions exist for the mobile robots along their own paths, so that each robot can reach its own goal without colliding with the other ones. The problem is known as the multiple robot path coordination problem [12].

A. Path Coordination Versus Path Planning

Multiple robot path coordination and path planning are two related issues in robot motion planning. In multiple robot path planning, the robot paths are not computed *a priori*. A solution to the multiple robot path planning problem is a collision-free path in the Cartesian product of the configuration spaces of all the robots. A solution to the problem exists iff the start and goal configurations belong to the same connected component of the global collision-free configuration space. Searching such a space is a combinatorially difficult problem [8]. Complete and exact centralized algorithms are therefore limited to simple problem settings involving two or three simple robots (e.g., [21]). Potential field techniques (e.g., [22]) have been proposed for more complex problems. In [2], a randomized search is combined with potential fields to centrally plan the motions of several translating robots.

To face this complexity, several authors have investigated decoupled schemes. The decoupled approach was introduced in [9] to solve problems with multiple moving objects: the method first plans a path among the stationary obstacles and then tunes the velocity along the path to avoid collisions with the moving obstacles. Such a decoupled approach has been further revisited: the prioritized planning scheme proposed in [7] assigns priorities to each robot and sequentially computes paths in a time-varying configuration space, given the paths computed for the higher priority robots. In [23], prioritization is combined with potential fields to resolve possible conflicts. Issues for selecting priorities are discussed in [4]. Some prioritized scheme is also used in the decentralized approach proposed in [1] for controlling the execution of a large fleet of autonomous mobile robots.

The path coordination problem as such was addressed in [17] where the notion of coordination diagram was first introduced for two robots. This diagram represents placements along each robot path at which mutual collisions might occur. The coordination space for two manipulators is analyzed in [3] and [6]. An algorithm based on dynamic programming was proposed more recently in [10] to find optimal strategies for three robots. This paper also introduced the idea of roadmap coordination that imposes weaker constraints than path coordination on the robot motions. A probabilistically complete planner based onto this roadmap coordination scheme [20] was applied to problems involving up to five robots.

From another point of view, cooperation-oriented approaches are based on local information (potential methods) (see, for instance, [18] and [5] for a recent overview). Path coordination is out of the scope of these methods.

B. Objective, Approach, and Contribution

Our objective is to solve practical problems involving a large fleet of mobile robots. Fig. 1 shows a coordination problem with 150 robots solved by the algorithms described in the paper. The proposed technique consists of searching an n-dimensional coordination diagram. The main contribution is an efficient algorithm that we propose to explore the coordination diagram without computing the exact shape of the obstacles. With respect to the previous works above, we do not use any regular grid representation. We propose instead an implicit model of the diagram obstacles by an adequate approximation of two-dimensional (2-D) diagrams represented by a set of bounding boxes. The algorithm is resolution–complete but it is shown to be complete for a large class of inputs. Despite the exponential dependency of the coordination problem on the number of robots, the model we propose allows us to efficiently solve problems involving up to ten robots in complex situations where most of the robots interfere. Such efficacy combined with partitioning

T. Siméon and J.-P. Laumond are with LAAS-CNRS, 31077 Toulouse, Cedex 4, France. (e-mail: nic@laas.fr; jpl@laas.fr).

S. Leroy is with the Rational Software Corporation, BP 10-31312 Labége Cedex, France (e-mail: sleroy@rational.com.)

[1]This paper is built upon work published separately in [19] and [15].

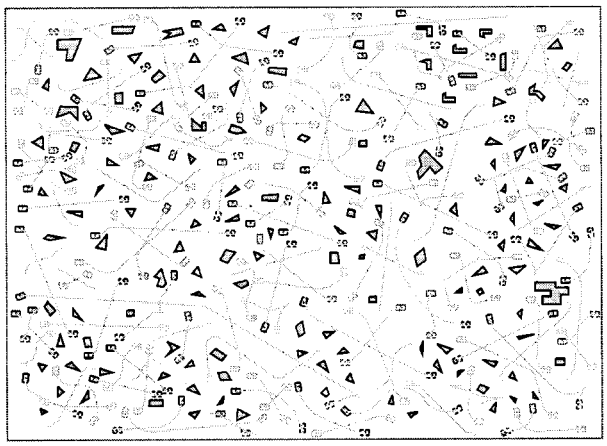

Fig. 1. A case with 150 robots.

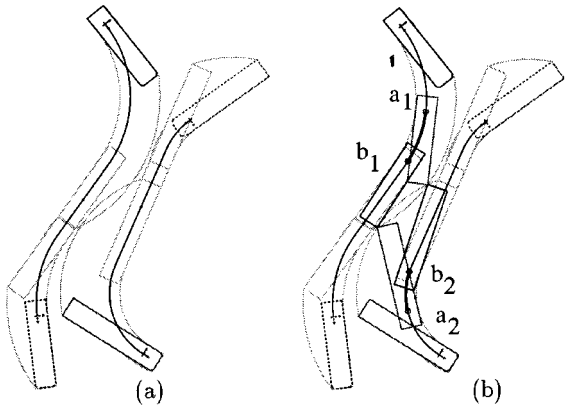

Fig. 2. Two intersecting robot traces. (a) Two intersecting robot traces. (b) The coordination configurations.

techniques allows us to solve practical problems involving a much higher number of robots in practical situations where each robot interferes locally with a subgroup of other robots. The algorithm inherits from the efficiency of simple geometric operations giving rise to an exact collision-checker dedicated to mobile robot coordination and described in Section II. After having introduced a cell decomposition of the coordination diagram for the case of two robots (Section III), we extend the algorithm to the general case (Section IV).

II. Paths $\mathcal{SA}$ and Geometric Tools

A. Paths $\mathcal{SA}$

The geometric tools presented in this section originate in [19]. They are based on the following assumption: we consider *convex* polygonal robots moving along paths defined by sequences of straight line segments (S) and arcs of a circle (A). Such sequences are denoted by $\mathcal{SA}$. This assumption is supported by both theoretical and practical considerations. First of all, it has been proved that a collision-free admissible path exists iff there exists a collision-free admissible path of type $\mathcal{SA}$[13]. Moreover, most of the existing complete motion planners for mobile robots provide solution paths of the type $\mathcal{SA}$ (e.g.,[14], [11], [20], [16]). Finally geometric algorithms like Boolean operations or swept volume computations are simple and computationally efficient when dealing with arcs of circles and straight line segments.

B. Traces

A mobile robot path being given, a *trace* is the volume swept by the robot when moving along the path. Assuming that the robot is a polygon, the trace of a path of type $\mathcal{SA}$ is a generalized polygon whose boundary is a sequence of straight line segments and arcs of a circle. In [19], we show how to compute such traces efficiently [Fig. 2(a)].

C. Coordination Configurations

To coordinate the motions of two robots along their path, it is necessary to compute the intersection of their traces. The bold subpath $[a_1, b_1]$ (respectively $[a_2, b_2]$) of Fig. 2(b) gathers the configurations at which the first (respectively second) robot intersects the trace of the other one. The endpoints of such collision subpaths are called *coordination configurations*. We describe below a geometric algorithm for the exact computation of the coordination configurations when the convex robots move along $\mathcal{SA}$ paths.

D. Computation of the Coordination Configurations

Given two paths γ_1 and γ_2, we want to compute the coordination configurations for the first robot moving along γ_1, with respect to the trace swept by the second robot moving along γ_2 (i.e., the configurations where the first robot enters or exits the trace of the second one).

Path γ_1 is a sequence of straight line segments and arcs of a circle. For each element of the sequence, elementary collision subpaths are computed by considering each edge of the polygonal robot[2] against the trace of the second robot. Each application of the algorithm produces collision subpaths along γ_1. The union of all collision subpaths provides the coordination configurations along γ_1.

The inputs of the basic geometric algorithm are therefore: 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 let $e(s)$ 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^{th} collision either begins or stops between $e(s)$ and *PG*.

1) Case of a Segment Subpath (S): Let us illustrate the algorithm on the canonical example of Fig. 3. The figure shows the collision subpaths that should be produced by the algorithm. The bold curves of *PG*'s contour represent the only curves that need to be considered for this computation.

Decomposition of PG's contour: 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. Let $\mathcal{D}$ be the domain lying between the two lines δ_A and

[2]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! . . .

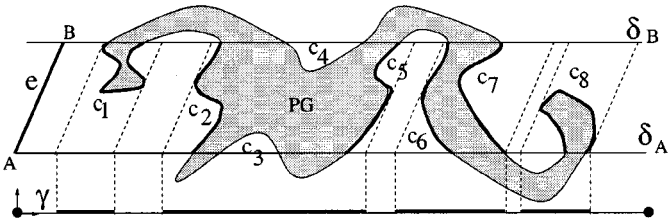

Fig. 3. Collision subpaths with a generalized polygon.

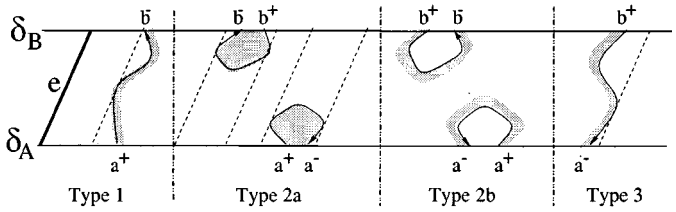

Fig. 4. The three elementary cases.

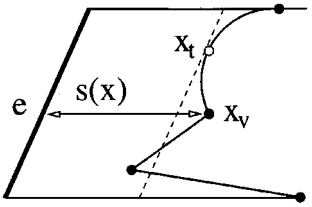

Fig. 5. Two type of events: vertices of *PG* and tangency points.

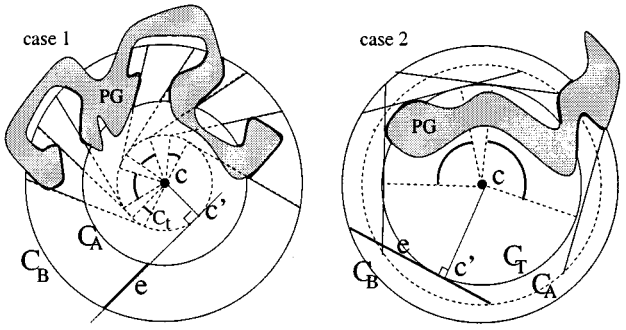

Fig. 6. The two cases of arc subpaths.

δ_B. 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 $\mathcal{D}$. 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 (e.g., *c*2, *c*6), an end point (e.g., *c*5, *c*7) or both endpoints (e.g., *c*1, *c*8) of a collision subpath, while others do not produce any endpoint (e.g., *c*3, *c*4).

Labeling the events: Fig. 4 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.

Computing the events: since each part corresponds to a connected sequence of segments and circular arcs, the endpoints of the collision subpaths 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 (see Fig. 5). Let $s(x)$ be the s-value along γ at which such a point x belongs to $e(s)$. A start (end) point is obtained by considering the minimal (maximal) value of all the $s(x)$ computed along the part.

Algorithm: 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. Then, an additional step is required to compute their union. The algorithm returns the ordered set of nonoverlapping intervals included into the interval [0, 1] (i.e., the collision subpaths of γ).

2) Case of an Arc Subpath (A): The principle of the algorithm remains similar to the one described above for the case of segment subpaths. However, two situations possibly occur when considering the trace swept by the edge e along an arc of a circle γ with radius r and centered at c. As illustrated by Fig. 6, the type of situation depends on the relative position between e and c.

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 above for the case of segment subpaths.

For a given part, Fig. 7 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 us to consider additional points x_A resulting from the intersection between $\mathcal{C}_A$ and the part.

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 Fig. 8). 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.

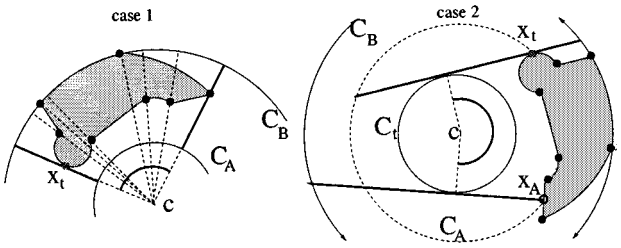

Fig. 7. Relevant points considered by the algorithm.

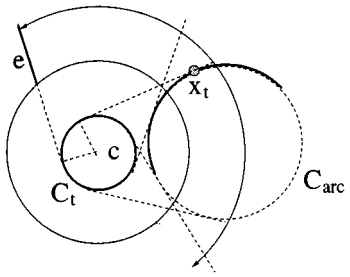

Fig. 8. Tangent points between the moving edge e and a circular arc.

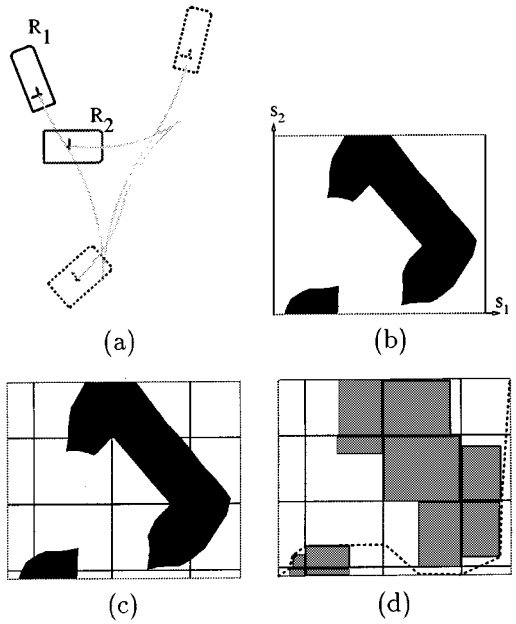

Fig. 9. (a) Two $\mathcal{SA}$ paths. (b) The coordination diagram. (c) The partition of the diagram induced by the path decomposition. (d) The bounding box representation of the obstacles and a solution path.

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. Then, $O(k)$ (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 $O(n + k \log k)$.

III. Coordination for Two Robots

A. Coordination Diagram

Coordinating the motion of two robots along two given paths is a classical problem. Its solution consists of exploring the so-called *coordination diagram* [17]. Let us consider the two paths $\gamma_1(s_1)$ and $\gamma_2(s_2)$ in Fig. 9(a). Both coordinates s_1 and s_2 are assumed to vary from 0 to 1. Fig. 9(b) shows the corresponding coordination diagram (s_1, s_2): the black domains represent the set of configuration pairs (s_1, s_2) such that the robots collide when they are at configurations $\gamma_1(s_1)$ and $\gamma_2(s_2)$, respectively. Black domains are obstacles to avoid. A coordinated motion exists iff there is a collision-free path in the diagram linking the point (0,0) (the robots are both at the beginning of their own path) to the point (1,1) (the robots are both at the end of their path).

B. A Bounding Box Representation

Our contribution is to propose an algorithm to explore the diagram without computing the exact shape of the obstacles.[3] We use a bounding box representation based on the following property: *the (minimal) box bounding an obstacle in a coordination diagram is a rectangle whose endpoint coordinates are the coordination configurations.*[4] Let us consider the case in Fig. 2. The coordinates of four points defining the rectangle in the coordination diagram are respectively (a_1, a_2), (a_1, b_2), (b_1, b_2), and (b_1, a_2). The computation of the boxes is then done by computing the coordination configurations (see above).

C. Path Decomposition

Let us now consider two $\mathcal{SA}$ paths γ_1 and γ_2. Instead of applying the bounding box representation directly in the coordination diagram of γ_1 and γ_2, we first apply a path decomposition. Each path is decomposed into its elementary pieces consisting of either straight line segments or arcs of a circle. Let $(\gamma_{1,i})$ and $(\gamma_{2,j})$ denote the pieces sequences of paths γ_1 and γ_2. The coordination diagram for γ_1 and γ_2 then appear as the union of the coordination diagrams of the various pairs $(\gamma_{1,i}, \gamma_{2,j})$. For instance, both paths in Fig. 9(a) consist of four arcs of a circle. Then, the coordination diagram appears as the union of 16 elementary coordination diagrams [Fig. 9(c)]. Then, for each elementary coordination diagram, we compute a bounding box representation of the obstacles. Fig. 9(d) shows the bounding box representation of the diagram in Fig. 9(b).

D. Search

Such a representation induces a cell decomposition of the coordination diagram into rectangles. Any classical search algorithm may be used to compute a collision-free path from the origin (0,0) to the goal (1,1). Fig. 9(d) shows a solution path. For this example, note that the widest robot R_2 (corresponding to the vertical coordinate in the diagram) should *necessarily* move forward, backward, and then forward along the first two pieces of its path.

[3]The obstacles in Fig. 9(b) have been computed with a brute force discrete approach used only for display purpose.

[4]In our context, the coordinate of a configuration on a path γ is its curvilinear abscissa s on γ.

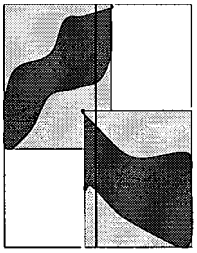

Fig. 10. This case cannot appear when at least one robot moves along a straight line segment.

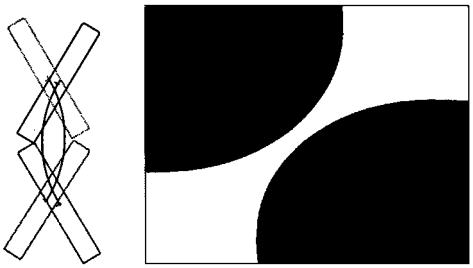

Fig. 11. A||A special case: bounding boxes would fill the space.

E. Completeness

The algorithm is complete iff it is complete when applied to the elementary diagrams corresponding respectively to three cases: S||S, S||A, and A||A.

For the first two cases, the algorithm is complete. The only way for the bounding box approach to lose a solution is if there exist one vertical *and* one horizontal line intersecting two obstacles (Fig. 10). This is, however, not possible since at least one robot moves along a straight line segment: indeed, the robot moving along the straight line cannot intersect *twice* the other (convex) robot remaining at a *fixed* position. Thus, the bounding box approximation does not affect the completeness of the algorithm for these first two cases.

Completeness is not necessarily guaranteed in the third case A||A: we may find counterexamples where the bounding box approximation of the obstacles may split the free space into two connected components. Fig. 11 shows an example where the bounding box transforms the full space into an obstacle. However, such cases can be solved by the following resolution-complete procedure: both arcs of a circle are recursively split into smaller arcs and each pair of the new elementary pieces is processed with the bounding box approach. Moreover, such cases are easily identified in the path decomposition step above. This means that, according to the inputs, the algorithm may activate the recursive subdivision. The activation condition is a function dedicated to the case A||A and is checking the existence of a collision-free vertical or horizontal line in the diagram. The activation cases are seldom seen. For instance, they never appear in the examples displayed in Figs. 13, 15, and 1.

Finally, one may note that, for concave robots, a similar recursive subdivision applied to the two first cases allows to maintain the resolution completeness of the algorithm. Therefore, the method also works for concave robots; the only critical point concerns the extension of the algorithm used for the exact trace computation (Section II-B), currently limited to convex polygons.

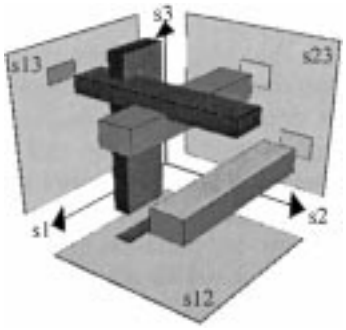

Fig. 12. Cylindrical structure of the coordination diagram.

IV. Coordination for n Robots

A. Generalized Coordination Diagram (GCD)

Let us now consider n robot paths $\gamma_i(s_i)$. The *generalized coordination diagram* is an n-dimensional cube defined by the Cartesian product of the n sets of parameter values s_i that place each robot along its path γ_i. The n-cube can be subdivided into free and obstacle regions that reflect the possible interferences between robots: a point $(s_1, \ldots, s_n)$ belongs to an obstacle iff at least two robots collide; otherwise it is free point. A solution to the coordination problem is a collision-free path between $(0, \ldots 0)$ to $(1, \ldots 1)$. This notion of obstacle induces a cylindrical structure of the coordination diagram.[5] Each obstacle is a cylinder defined by the Cartesian product of the forbidden region on some s_i, s_j face of the n-cube with the remaining axis (see Fig. 12). As a consequence, the topology of the generalized coordination diagram is fully characterized by the topology of the $n(n-1)/2$ elementary (γ_i, γ_j) coordination diagrams. The algorithm presented below for coordinating the motions of n robots exploits this property to avoid the explicit computation of the n-dimensional obstacles. The cell decomposition of the n-cube is implicitly derived from the 2-D diagrams computed for each pair of robots. Fig. 13(b) shows the 10 elementary diagrams computed for the path coordination problem of Fig. 13(a).

B. GCD Modeling and Searching

We have seen that the bounding box representation of the coordination diagram for two robots induces a decomposition of the diagram into rectangles. Let us consider three paths γ_1, γ_2, and γ_3. The cell decomposition of (γ_1, γ_2) coordination diagram induces a partition of the axis s_2. Then the cell decomposition of the (γ_2, γ_3) diagram is refined according to this partition. More generally, the cell decomposition of a (γ_i, γ_j) diagram induces a refinement of the cell decompositions of the $2(n-2)$ diagrams (γ_i, γ_k) and (γ_k, γ_j) (see Fig. 14). We denote by (i, j)-cell a cell of the (γ_i, γ_j) diagram after refinement. The cells of the n-cube are denoted by n-cells. The 2-D (i, j)-cells of all the (γ_i, γ_j) diagrams induce the cell decomposition of the n-cube into n-cells. The main advantage of this modeling is that it does not require the explicit representation of the n-cube.

[5]This property has been already noticed in [10]

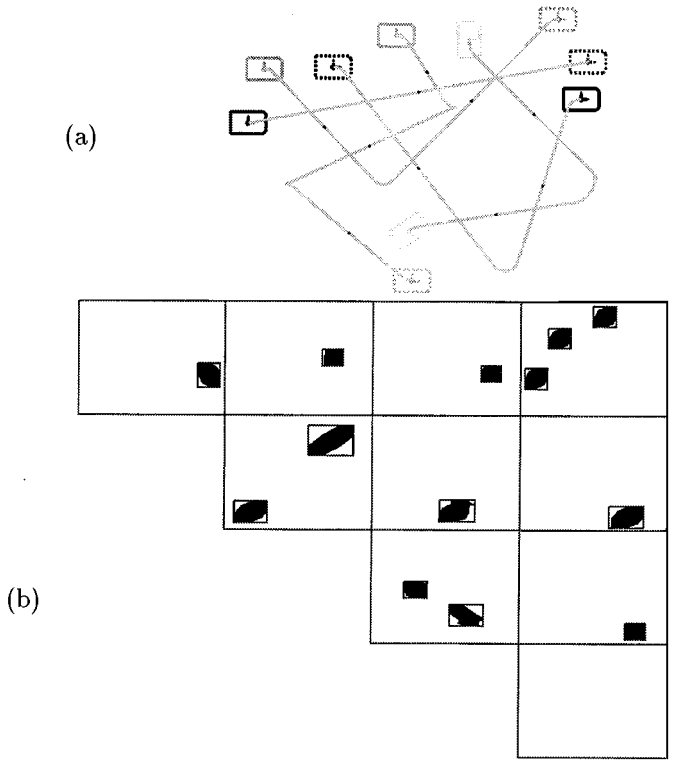

Fig. 13. The 10 elementary diagrams (b) of the generalized coordination diagram of 5 paths (a).

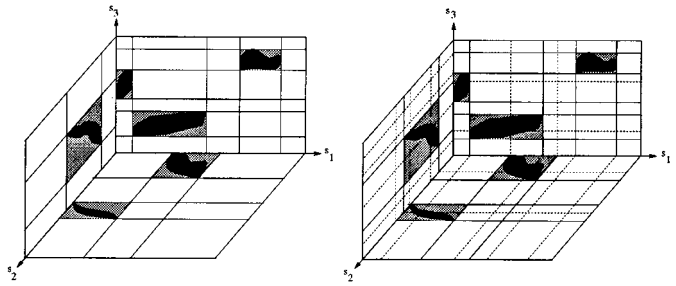

Fig. 14. The cell decomposition of a diagram refines the cell decomposition of other diagrams.

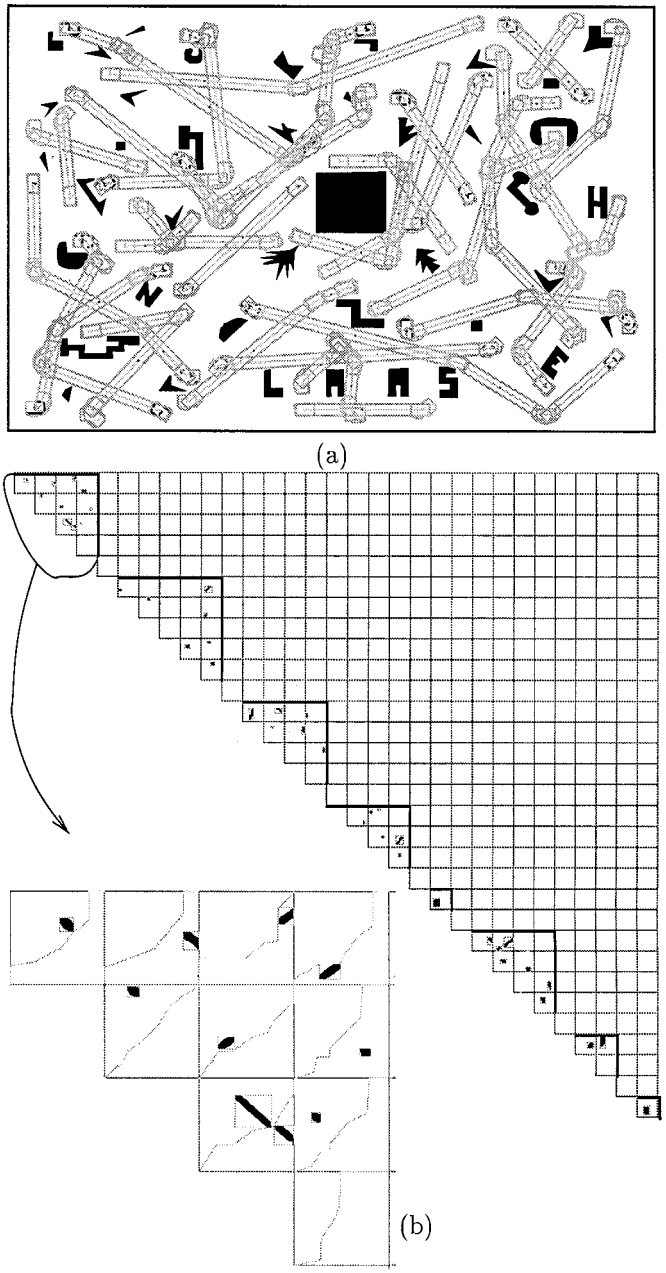

Fig. 15. A case with 32 robots: (a) the robots traces and (b) the 496 elementary diagrams. The partition into the eight robot subgroups is illustrated by the eight bold triangles.

Hence, while the size of the n-cube grows exponentially with the number n of robots, the complexity of our implicit modeling remains much lower. Let m denote the average number of intersections for a given pair of paths (i.e., number of obstacle boxes of a (γ_i, γ_j) diagram). Each of the $O(n^2)$ diagrams initially contains $O(m)$ cells. After the refinement step, each axis of the n-cube is partitioned into $O(mn)$ intervals, which means that refinement of each elementary diagram (γ_i, γ_j) produces $O(m^2n^2)$ (i, j)-cells. Therefore, the n-cube is implicitly modeled by $O(m^2n^4)$ 2-D cells.

The search is performed by an A^* algorithm with a heuristic function based onto the shortest Euclidean path to the goal point $(1, \ldots 1)$ of the n-cube. At each iteration during the search, the algorithm has to generate the k-neighbors[6] (for some $k \in [1, n]$) of the current cell. In our implementation, we use $k = 1$ in order to limit the size of the neighborhood. Thus, each step of the search generates only $2n$ cells adjacent to the current n-cell through an $(n-1)$-dimensional hyper-plane. This means that the coordination solution returned after the search corresponds to Manhattan paths where only one robot moves at once.

Let us consider an n-cell cell, adjacent to the current collision-free n-cell and corresponding to an elementary motion of robot i. Due to the cylindrical shape of the obstacles, testing if the cell is collision-free is easily performed: each of the $(n-1)$ projections of cell onto the elementary $(\gamma_i, \gamma_.)$ diagrams has to be a collision-free $(i, .)$-cell.

Despite the important size of the n-cube, a very small subset of n-cells has to be explored in most cases before a coordination solution is found. However, a problem admitting no solution

[6]In an n-dimensional space, the number of 1-neighbors, 2-neighbors, $\ldots$, n-neighbors are, respectively, $2n$, $2n^2$, $\ldots$, and $3^n - 1$.

TABLE I

	32 rob.	150 rob.
Interaction graph computation and bounding box representation of the diagrams	$30s$	$240s$
Diagram refinement	$6s$	$13s$
Search	$3.7s$	$1.5s$

may require most of the free cells to be developed before the algorithm terminates.

As noted above, the algorithm generates Manhattan paths (only one robot moves at once). Allowing simultaneous robot motion generally reduces the total path length. This can be achieved with a postprocessing step using classical smoothing techniques as in [20] to combine sequences of simple robot motions into simultaneous ones. This step produces diagonal paths, as illustrated in the inset of Fig. 15(b).

C. Completeness

Due to the cylindrical shape of the obstacles in the generalized coordination diagram, the algorithm above inherits from the completeness property of the coordination procedure for two robots presented in Section III.

D. Interaction Graph

The final extension we propose is supported by a practical assumption. When a high number of robots plan their paths independently, the path coordination problems are in general localized in different domains of the environment and only concern robot *subsets*. To reduce the combinatorial complexity of the global problem in practice, we first identify which robot traces intersect another trace. We then build an *interaction graph* whose nodes are the robots; two robot-nodes are adjacent iff both corresponding traces intersect. A simple decomposition of the graph into connected components identifies automatically the various subgroups of robots requiring motion coordination. We then apply the coordination algorithm to each subgroup.

E. Results

Fig. 15(a) shows an example of 32 mobile robots paths (including the traces). The eight connected components of the interaction graph have been computed automatically. The global coordination diagram of Fig. 15(b) clearly shows the structure induced by the eight connected components. It also shows a detailed view of the coordination diagram involving a subgroup of five robots with a display of the computed solution path for this group.

Table I presents the computation times (Sparc Ultra-1, 170 MHz) of the main steps of the algorithm for the examples[7] of Figs. 15 and 1. For the second example that involves 150 robots, the interaction graph contains 37 connected components containing up to 8 robots.

We should note that the performance depends on the decomposition of the interaction graph into connected components. The worst case appears when the interaction graph has only one component (e.g., when the trace of some robot intersects *all* the other traces). In fact, the complexity of the approach is dominated by the highest dimension of the considered n-cubes. In practice, the algorithm may efficiently explore n-cubes of dimension up to ten (i.e., involving ten robots). We just argue that this limitation is not critical in practice. Moreover, we do not know any alternative approach allowing to solve the case of Fig. 1.

[7]The motion planner computing an admissible collision-free path for each robot is based on the algorithm presented in [14].

V. Conclusion

We have presented an efficient approach to multiple robot coordination. The power of the approach comes from an implicit model of the n-dimensional coordination space exploiting the cylindrical structure of the diagram obstacles. The proposed model is derived from a bounding-box representation of the obstacles in the elementary 2-D diagrams. The method is resolution complete and the simulation results show its efficacy to coordinate large fleets of robots.

The algorithm was originally designed for mobile robotics applications. For such systems, the paper also describes efficient geometric tools for the exact characterization of the coordination configurations. However, it is important to note that the coordination approach is general and might be applied to other systems, provided dedicated geometric tools to compute the configurations at which two given paths interfere. Also, there remains other possible extensions. For example, when the coordination problem admits no solution, the analysis of the elementary diagrams might help to determine groups of few robots that cause the deadlock; a centralized planner could then be used to resolve the local conflicts. Some applications might also require the incremental update of the coordination problem each time a new robot is inserted or removed. Updating the coordination diagram could be efficiently performed due to the implicit model that mostly requires one to build (or delete) the n elementary diagrams associated with this robot.

References

[1] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki, "Multi-robot cooperation through incremental plan-merging," in *IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995, pp. 2573–2578.

[2] J. Barraquand and J. C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–649, 1991.

[3] Z. Bien and J. Lee, "A Minimum-time trajectory planning method for two robots," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 414–418, June 1992.

[4] S. J. Buckley, "Fast motion planning for multiple moving robots," in *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 322–326.

[5] Y. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 7–27, 1997.

[6] C. Chang, M. J. Chung, and B. H. Lee, "Collision avoidance of two robot manipulators by minimum delay time," *IEEE Trans. Syst., Man Cybern.*, vol. 24, no. 3, pp. 517–522, 1994.

[7] M. Erdmann and T. Lozano-Pérez, "On multiple moving objects," in *Proc. IEEE Int. Conf. on Robotics and Automation*, San-Francisco, CA, 1986, pp. 1419–1424.

[8] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independant objects: PSPACE-hardness of the Warehouseman's Problem," *Int. J. Robot. Res.*, vol. 3, no. 4, pp. 76–88, 1984.

[9] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.
[10] S. La Valle and S. Hutchinson, "Optimal motion planning for multiple robots having independants goals," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Minneapolis, MN, 1996, pp. 1847–1852.
[11] J. C. Latombe, "A fast path planner for a car-like indoor mobile robot," in *9th Nat. Conf. on Artificial Intelligence, AAAI*, Anaheim, CA, 1991, pp. 659–665.
[12] ——, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.
[13] J. P. Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," in *Int. Conf. on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, 1986.
[14] J. P. Laumond, P. Jacobs, M. Taïx, and R. Murray, "A motion planner for non holonomic mobile robots," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 577–593, Aug. 1994.
[15] S. Leroy, J. P. Laumond, and T. Siméon, "Multiple path coordination for mobile robots: A geometric algorithm," in *16th Int. Joint Conf. Artificial Intelligence*, Stockholm, Sweden, 1999, pp. 1118–1123.
[16] B. Mirtich and J. Canny, "Using skeletons for nonholonomic motion planning among obstacles," in *IEEE Int. Conf. on Robotics and Automation*, Nice, France, 1992, pp. 2533–2540.
[17] P. O'Donnell and T. Lozano-Pérez, "Deadlock-free and collision-free coordination of two robot manipulators," in *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 484–489.
[18] J. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," in *In Algorithmic Foundations of Robotics*, K. Goldberg, Ed: A. K. Peters, 1995, pp. 331–345.
[19] T. Siméon, S. Leroy, and J. P. Laumond, "A collision checker for car-like robots coordination," in *IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, 1998, pp. 46–51.
[20] P. Svestka and M. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps," in *IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995, pp. 1631–1636.
[21] J. T. Schwartz and M. Sharir, "On the piano movers problem: III. Coordinating the motions of several independant bodies," *Int. J. Robot. Res.*, vol. 2, no. 3, pp. 46–75, 1983.
[22] P. Tournassoud, "A strategy for obstacle avoidance and its application to multi-robot systems," in *IEEE Int. Conf. on Robotics and Automation*, San-Francisco, CA, 1986, pp. 1224–1229.
[23] C. W. Warren, "Multiple robot path coordination using artificial potential fields," in *IEEE Int. Conf. on Robotics and Automation*, Cincinnati, OH, 1990, pp. 500–505.

Thierry Siméon graduated from the Institut National des Sciences Appliquées in 1985. He received the Ph.D. degree in robotics and the Habilitation degree from the University Paul Sabatier, Toulouse, France, in 1989 and 1999, respectively.

After a year of post-doctoral work at the University of Pennsylvania, Philadelphia, he joined LAAS-CNRS, Toulouse, in 1990 as Chargé de Recherche. His research interests include robotics, motion planning, geometric algorithms and mobile robot navigation. He has been involved in several European projects, in particular the Esprit 3 BRA project PROMotion (Planning RObot Motion), the Eureka project I-ARES (Rover for Planetary Exploration). He is currently coordinator of the Esprit 4 LTR project Molog (Motion for Logistics, 1999–2002).

Stéphane Leroy graduated with a degree in computer science from the University of Rennes, France, in 1994. He received the DEA degree in computer science from the Ecole Polytechnique at Palaiseau, France, in 1995 and the Ph.D. degree in robotics from the University Paul Sabatier,Toulouse, France, in 1998.

His research interests include nonholonomic mobile robots and multiple robots coordination. He is currently with Rational Software Corporation, Toulouse, France.

Jean-Paul Laumond (M'95) received the M.S. degree in mathematics, the Ph.D. degree in robotics, and the Habilitation degree from the University Paul Sabatier, Toulouse, France in 1976, 1984, and 1989, respectively.

He is Directeur de Recherche at LAAS-CNRS, Toulouse. His research interests include mainly robotics and algorithmic motion planning. He was a member of the Comité National de la Recherche Scientifique from 1991 to 1995. In 2001, he created Kineo, Computer Aided Motion, a spin-off of CNRS diffusing a software development kit dedicated to path planning toward CAD-CAM, Graphics, games and 3-D web markets.