2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Coordination of Multiple AGVs in an Industrial Application

Roberto Olmi, Cristian Secchi and Cesare Fantuzzi

Abstract— **In this paper we propose a methodology for coordinating a group of mobile robots following predefined paths in a dynamic industrial environment. Coordination diagrams are used for representing the possible collisions among the robots. Exploiting the structure of the industrial application we are dealing with, we propose an algorithm for efficiently composing the coordination diagram. Furthermore, we classify the possible collisions that can take place and the induced geometry of the resulting coordination diagram. Finally, we exploit this information for developing a planning algorithm that allows to coordinate the robots and to take into account unexpected events that can occur in an industrial environment.**

I. Introduction

Automated guided vehicles (AGVs) are used more and more in industrial plants and warehouses. In these applications, a central issue is how to plan the motion of the AGVs in order to minimize the delivery time while avoiding collisions and deadlocks.

A fleet of robots can be coordinated using either a centralized or a distributed control architecture. In the first case, a central unit computes all the control actions to deliver to the robots. In the second case, each robot computes its action. The interested reader is addressed to [1], [2], [3], [4], [5] for further information.

Furthermore, it is possible to classify the coordination strategies into two main classes: centralized approaches and decoupled approaches. *Centralized approaches* search the solution of the motion planning problem in a composite coordination space, which is formed by the Cartesian product of the configuration spaces of the individual robots. A method that uses the notion of composite robot, but that does not require the computation in composite configuration space, is presented in [6]. In [7] a technique, based on a Petri nets, to avoid deadlock through re-routing is presented. *Decoupled approaches* face the complexity of the coordination by breaking the problem into two distinct phases: path planning and motion coordination. During the first phase a path for each vehicle is planned without considering the presence of other vehicles. In the second phase the velocity profile with which each robot has to track its assigned path is computed. In [8] the problem of defining a velocity curve is transformed in a Mixed Integer Non-Linear Programming problem for finding a global optimum for the coordination problem. In [9] an algorithm for computing Pareto optimal coordination path within the coordination space is presented. In [10], [11] time optimal coordination algorithms which can be considered somewhere between centralized planning and decoupled planning are presented. In [12] a technique to control the motions of several robots moving along fixed independent paths is presented. This technique exploits an algorithm for computing a bounding box representation of the so called coordination diagram, given the path of each robot. This representation induces a cell decomposition of the diagram that allows any classical search algorithm to be used for computing the collision-free path within the coordination diagram.

The work presented in this paper is made in cooperation with a company producing AGVs for transporting goods in warehouses. The robots are controlled through a centralized architecture. Given a warehouse, a roadmap along which the AGVs can move is designed. The central system assigns to each AGV a mission, namely a path on the roadmap from a starting point to a goal, that the robot has to track. Our goal is to provide a coordination algorithm which is efficient and that avoids collisions between robots. We develop a coordination algorithm based on coordination diagrams. The construction of the coordination diagram is, in general, a computationally demanding task [12]. As a first contribution of this paper, we exploit the structure of the roadmap to develop a technique to build the coordination diagram online with a very limited computational effort. Loosely speaking, when the roadmap is defined we generate a set of offline sub-coordination diagrams. When a group of robots starts moving over the roadmap, the coordination diagram is composed by putting together the proper sub-coordination diagrams which are identified by the paths tracked by the robots. Furthermore, we classify the possible collisions that can take place and the induced geometry of the resulting coordination diagram. Finally, as a second contribution, we exploit this information for developing an incremental coordination algorithm. This latter feature allows to take into account unexpected events that can occur in an industrial environment (e.g. temporary malfunctioning of the AGVs, emergency stops).

The paper is organized as follows: in Sec. II a formal definition of the problem is reported and in Sec. III the technique for composing the coordination diagram is presented. In Sec. IV we illustrate the possible kinds of cordination diagram.In Sec. V we present the planner for the motion coordination of the vehicles and in Sec. VI we propose some simulations Finally, in Sec. VII some conclusions are drawn and some future work is addressed.

Authors are with the Department of Sciences and Methods of Engineering, University of Modena and Reggio Emilia, via G. Amendola 2, Morselli Building, Reggio Emilia I-42100 Italy {roberto.olmi,cristian.secchi, cesare.fantuzzi}@unimore.it

The authors wish to greatly acknowledge the company Elettric80 s.p.a. (www.elettric80.com) which supported this research.

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

Authorized licensed use limited to: ULAKBIM UASL - Bilkent University. Downloaded on July 13,2023 at 09:32:58 UTC from IEEE Xplore. Restrictions apply.

II. Overview of the problem

A. Roadmap and missions

We consider N AGVs that are moving in the same environment and that share the same configuration space $\mathcal{C}$ (e.g. $SE(2)$). In the application that we are considering, for a given plant to be served, a network of paths which the AGVs can follow is defined. We can model this network as a *roadmap* $\mathcal{R}$, that is a one-dimensional connected subset of $\mathcal{C}$. The roadmap is formed by a collection $\mathcal{T}$ of regular curves. Each element of $\tau_i \in \mathcal{T}$ is a mapping $\tau_i : [0,1] \to \mathcal{R}$ and we call it *segment*. A *route* π_i is defined as a sequence of adjacent segments and it can be considered as a mapping $\pi_i : s_i \mapsto \mathcal{R}$, $s_i \in [0, l(\pi_i)]$ where $l(\pi_i)$ is the total length of the path.

We indicate with $\mathcal{A}(x_i)$ the portion of space occupied by an AGV at the configuration $x_i \in \mathcal{R}$. Two segments $\tau_i, \tau_j \in \mathcal{T}$ represents a pair of *colliding segments* if there exists a pair of scalars $(\alpha, \beta) \in [0,1]^2$ such that

$$\mathcal{A}(\tau_i(\alpha)) \cap \mathcal{A}(\tau_j(\beta)) \neq \emptyset \tag{1}$$

This means that when two AGVs are moving through τ_i and τ_j it can happen that a collision takes place.

For a given plant, a centralized planning system plans a set of missions to be executed by the AGVs. Each AGV has to execute a mission, namely to reach a goal configuration $x_i^{goal} \in \mathcal{R}$ starting from its initial configuration $x_i^{init} \in \mathcal{R}$. In case no more missions are scheduled, a homing mission is assigned to the AGV which is taken to a garage position. It is possible that the planning system can decide to change a mission previously assigned to an AGV. Furthermore, vehicles can be blocked by unexpected events for an unpredictable amount of time.

When a mission is assigned to an AGV, the shortest path $\pi_i : [0, l(\pi_i)] \to \mathcal{R}$, where $\pi_i(0) = x_i^{init}$ and $\pi_i(l(\pi_i)) = x_i^{goal}$, that it has to track is computed. Our goal is to determine the velocity profile by which each AGV has to move through the assigned path in order to avoid collisions among the AGVs and to minimize the total time required by the fleet for reaching the goal configurations.

For the moment, we do not take into account robot dynamics. In order to guarantee that the coordination problem admits a solution, we assume that, for each AGV, the initial and the goal configurations do not belong to an already planned path.

B. Coordination diagram

The coordination strategy that we are going to develop in the paper is based on the concept of *coordination diagram* [13]. Given N paths $\pi_1, \dots \pi_N$ parameterized by $s_1 \in [0, l(\pi_1)], \dots, s_N \in [0, l(\pi_N)]$, the coordination diagram represents all the configuration set of the robots along their paths and, therefore, it is given by $\mathcal{S} = [0, l(\pi_1)] \times \cdots \times [0, l(\pi_N)]$. A point $s = (s_1, \dots, s_N)$ in the coordination diagram represents a possible configuration of the robots along their paths. For each pair of paths, a *collision region* is defined as:

$$X_{ij}^{coll} = \{(s_1, \dots, s_N) \in \mathcal{S} \,|\, \mathcal{A}(\pi_i(s_i)) \cap \mathcal{A}(\pi_j(s_j)) \neq \emptyset\} \tag{2}$$

This region defines all the possible configurations of the fleet such that two vehicles collide moving along paths π_i and π_j. Since it depends only on the configuration of two robots, this region can be completely characterized by its 2D projection onto the (s_i, s_j) plane of the coordination diagram (that we will denote for shortly with CD_{ij}).

A *coordination path* is a map $\gamma : [0, T] \to \mathcal{S}$ that defines a coordinated motion of the robots along their predefined paths. A possible solution to the collision free coordination of the N AGVs is to find a coordination path that joins the point $s_0 = (0, \dots, 0) \in \mathcal{S}$, (starting configurations of the paths), to the point $s_f = (l(\pi_1), \dots, l_(\pi_N))$, (goal configurations), which avoids the collision regions in the coordination diagram.

III. Construction of the Coordination Diagrams

In this section we propose a strategy for building the coordination diagram corresponding to N AGVs moving along paths over a predefined roadmap $\mathcal{R}$.

When an AGV completes its last mission, a new one is assigned to it. This implies that the coordination diagram should be modified. In order to avoid to stop all the AGVs for computing the coordination diagram the algorithm must require a small computational effort. In [12] the path of each robot are considered to be not known a priori and the coordination diagram is computed online subdividing the path into straight line segment and arc of a circle. Our approach exploits the knowledge of the roadmap (recall Sec. II-A) in order to split this computation in an offline phase and an online phase. This will reduce the time needed for the computation of the coordination diagram.

The offline phase is run once after the definition of the roadmap. For each $\tau_h \in \mathcal{T}$ we define the *collision segments set* CSS_h (function Determine CSS_h in Alg. 1) as

$$CSS_h = \{\tau \in \mathcal{T} \mid \exists (\alpha, \beta) \in [0,1]^2 \mathcal{A}(\tau_h(\alpha)) \cap \mathcal{A}(\tau(\beta)) \neq \emptyset\} \tag{3}$$

namely the set of segments that are colliding with τ_h.

For each pair of colliding segments (τ_h, τ_k) we compute and store their relative two dimensional sub-coordination diagram sCD_{hk} using standard collision checking algorithms [1] (function Compute sCD_{hk} in Alg. 1). Furthermore, for each segment $\tau_h \in \mathcal{T}$, an empty *Booking Table* BT_h is created. This object contains the list of paths including that segment. The procedure for creating the sub-coordination diagrams is illustrated in Alg. 1. In summary, the main output of the offline phase, is a set of sub-coordination diagrams stored in a database as pieces of a puzzle that will be used in the online phase for building the global coordination diagram.

When a new path π_i is assigned to an AGV, all the sCD_{ij} describing the collision regions between the AGV along π_i and the other vehicles must be computed. For each segment τ_k of the path, we make a reservation on its corresponding

Authorized licensed use limited to: ULAKBIM UASL - Bilkent University. Downloaded on July 13,2023 at 09:32:58 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Sub-coordination Diagrams Computation

Require: Roadmap as a collection $\mathcal{T}$ of segments τ_i
  for all $\tau_h \in \mathcal{T}$ **do**
    DETERMINE CSS_h
    for all $\tau_k \in CSS_h$ **do**
      COMPUTE sCD_{hk}
    end for
    CREATE BT_h
  end for

booking table; in this way we specify that the segment τ_k is contained in the path π_i. We then check the set of colliding segments CSS_k. For each colliding segment $\tau_h \in CSS_k$, we check its booking table and in case it has been booked by another path π_j, we fetch the corresponding sub-coordination diagram sCD_{hk} from the database that we have built before. The path π_j can be on its turn split into a sequence of segments and τ_h will be part of this sequence. The decomposition of π_i and π_j into sequences of segments, induces a partition on the coordination plane. Each block of the partition is identified by a segment in the sequence of π_i and a segment of the sequence of π_j. Thus, the sub-coordination diagram sCD_{hk} is inserted (function INSERT(sCD_{hk}) in Alg. 2) in correspondence of the partition block of the (s_i, s_j) plane of the overall coordination diagram that is identified by τ_h and by τ_k.

Remark 1: The sub-coordination diagram computed in Alg. 1 are computed assuming a certain default travel direction over the two segments. During the composition phase, the real direction along which the colliding segments are crossed by the AGV is considered and the sub-coordination diagram is properly reversed before being inserted in the coordination diagram.

In summary, the composition of the diagram is the result of picking the right piece from a database that has been defined once the roadmap has been defined. The algorithm for composing the coordination diagram is reported in Alg. 2:

Algorithm 2 Coordination diagram composition

Require: Paths currently covered by moving vehicles
Require: New path π_i
  for all $\tau_h \in \pi_i$ **do**
    $BT_h \leftarrow \pi_i$
    for all $\tau_k \in CSS_h$ **do**
      if $BT_k \neq \emptyset$ **then**
        for all $\pi_j \in BT_k$ **do**
          $CD_{ij} \leftarrow$INSERT(sCD_{hk})
        end for
      end if
    end for
  end for

IV. TAXONOMY OF THE COORDINATION DIAGRAMS

In this section we provide a classification of the possible collisions that can take place between two vehicles and the induced geometry of the corresponding collision diagram. For the sake of simplicity, we consider circular robots of the same size. We consider roadmaps composed only by straight segments, spaced enough so that vehicles on parallel segments don't collide. Under these assumptions, the condition for which there is a collision is given by:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq d_{min} \tag{4}$$

where x_i and y_i, $i = 1, 2$, represent the coordinates of the center of the circle, and d_{min} is the diameter of the circle.

There are only three possible kinds of collisions between two AGVs:

- *Intersection collision:* it takes place when the paths intersect (see Fig. 1). By the paths parameterizations (using notations in Fig. 1, where c_{xi}, c_{yi} are the initial coordinates of vehicle i) the condition (4) becomes:

$$((c_{y2} - c_{y1}) - s_1)^2 + ((c_{x1} - c_{x2}) - s_2)^2 \leq d_{min}^2 \tag{5}$$

- *Front collision:* it takes place when the paths have some common segments and the AGVs move on them in opposite senses. Using the path's parameterizations (with notations in Fig. 1, where we have posed $\beta = c_{y2} - c_{y1}$, and where $c_{x1} \equiv c_{x2}$), the condition in (4) becomes:

$$\beta - d_{min} \leq s_1 + s_2 \leq \beta + d_{min} \tag{6}$$

  Thus, the collision zone is defined by all points between the lines of equation $s_2 = (\beta - d_{min}) - s_1$ and $s_2 = (\beta + d_{min}) - s_1$.
- *Back collision:* it takes place when the paths have some common segments and the AGVs move in the same sense (see Fig. 1). By computations analogous to the ones for front collision, we obtain that the collision zone is defined by all points between the lines of equations $s_2 = (\beta + d_{min}) + s_1$ and $s_2 = (\beta + d_{min}) + s_1$.

To simplify the shape of the collision regions, we approximate (with a little loss in completeness) them with an

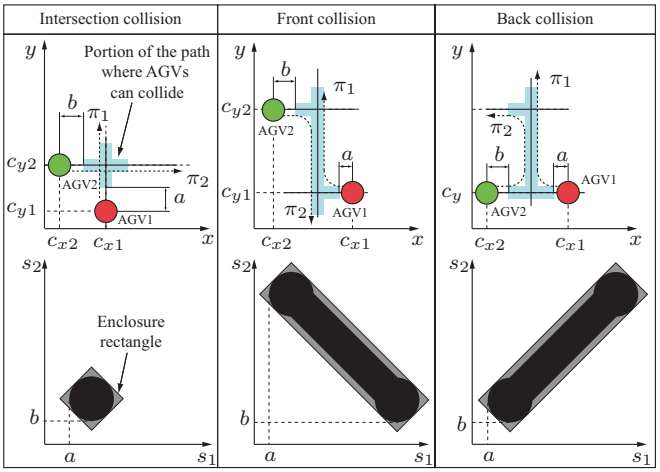

Fig. 1. Three kinds of collision and the corresponding coordination diagrams

Authorized licensed use limited to: ULAKBIM UASL - Bilkent University. Downloaded on July 13,2023 at 09:32:58 UTC from IEEE Xplore. Restrictions apply.

enclosing rectangle having two sides parallel to the bisector of the diagram. In Fig. 1 we have represented the enclosure rectangle gray filled.

V. Motion planner

As reported in Sec. II, in order to find a solution to the coordination problem it is sufficient to find a coordination path $\gamma : [0, T] \mapsto \mathcal{S}$ in the coordination diagram $\mathcal{S}$ such that $\gamma(0) = (0, \ldots, 0)$ and $\gamma(T) = (l(\pi_1), \ldots, l(\pi_N))$.

We want to determine an incremental algorithm which determines the coordination action step by step (like in [2]) rather than an algorithm that defines the overall coordination strategy in one shot (like in [12]). This choice is due to the fact that in factory applications a lot of unexpected events could prevent some AGVs from performing the pre-planned action. The problem of finding an optimal coordination path has an exponential complexity [12], [1] and, therefore, in case the number of AGVs is big, it could be necessary to stop the AGVs for a significant amount of time waiting for each new re-planning. An incremental algorithm decides which action to implement when the robots are at a given configuration by looking at all the possible collisions (due both to the presence of collision regions in the coordination diagram and to unexpected events) and decides the motion that the AGVs should make. Thus, the algorithm that we are proposing, allows to take into account also unexpected events without the need of re-planning each time the coordination of the robots.

In order to build the incremental coordination algorithm, we first need to impose a grid structure over the coordination diagram S. Thus, for each axis, we split the interval $[0, l(\pi_i)]$ of the coordination diagram into p_i segments. This induces a grid structure over each CD_{ij} plane. The granularity of the partition (i.e. p_i) depends on the particular application.

Remark 2: Notice that the partition of the axis for the planning algorithm is different from that considered in Sec. III since the granularity required is, in general, different from that induced by the segments in $\mathcal{T}$.

When moving over a path π_i, an AGV, executes an action u_i. It can either move forward to the next segment ($u_i = 1$) or move backward to the previous segment ($u_i = -1$) or remain motionless ($u_i = 0$). The role of the coordination algorithm is to tell to all the vehicles which action has to be executed. We define the *action set* as $U = \{(u_1, \ldots, u_N) \mid u_i \in \{-1, 0, 1\}\}$. The problem of choosing the right coordination action has, in general, exponential complexity. In order to decrease the computational effort, we will not consider generic collision regions, but we will limit ourselves to the three kinds of collision regions outlined in Sec. IV.

It is clear that, locally, the best coordination action is the one which leads to the major advancement of the fleet. Thus, the first criterion by which the actions are chosen is the maximization of overall fleet advancement. Since all robots can take only three actions, we can identify $2N + 1$ subsets $U_r \in U$ that contain actions with lead to the same *advancing factor* r:

$$U_r = \{(u_1, \ldots, u_N) \mid \sum_{i=1}^{N} u_i = r\} \tag{7}$$

Loosely speaking, the advancing factor, provides a measure of the advancement of the overall fleet. For example, the action subset U_{N-1} contains all the actions that make advancing all vehicles but one that remains motionless.

The algorithm evaluates, in decreasing order, the actions belonging to each subset U_r, starting from $r = N$, until it is found a subset that contains a valid action. In Alg. 3, we have denoted with Action Set(r) the function that gives the subset to U_r.

Thanks the cylindrical structure of the collision regions the algorithm can realize a coordination path considering just all the CD_{ij}. As we have seen in Sec. IV, collision regions on the two dimensional planes have a well defined shape. Thanks to this particular structure, we can define over each plane some regions that we call *shadow zones*. These regions are defined between the collision region, the axes of the plane and the two rays r_1 and r_2 tangent to obstacle borders and parallel to the bisector of the plane (see Fig. 2). The goal is to partition each plane in a set of zones in which some actions are forbidden since they would lead to collision or they would delay the completion time of the missions.

The algorithm has to choose an action that is allowed by all planes (a valid action). We denote by s_{ij} the point in CD_{ij} that denotes the configuration of two AGVs considered along their paths π_i and π_j. For all possible collision region (Sec. IV) we can define (see as an example Fig. 2):

- *Light region:* When s_{ij} falls in this region there are no actions that have to be discarded.
- *Antumbra region:* This region means that the robots are approaching to collision. As long as s_{ij} remains in this region, there are no action that penalize the possibility of escaping from the shadow zone. Thus, all the actions remain valid.
- *Penumbra region:* This corresponds to the situation in which one vehicle has reached a segment that belongs to the path of another vehicle. To escape from this region the following constraint between the actions has to be satisfied:
$$u_i + u_j \leq 1 \tag{8}$$
- *Umbra region:* This corresponds to the situation in which a couple of AGV has reached a common portion of path. The constraints imposed by this region are:
$$u_i + u_j \leq 0 \tag{9}$$
- *Obstacle border:* In this region, all the directions that enter into the collision region are forbidden. Referring to Fig. 2, the following constraints on the control actions have to be satisfied:
$$\begin{array}{ll} u_i - u_j \geq 0 & s_{ij} \in \overline{AB} \\ u_i + u_j \geq 0 & s_{ij} \in \overline{BC} \\ u_j - u_i \geq 0 & s_{ij} \in \overline{CD} \\ u_i + u_j \leq 0 & s_{ij} \in \overline{DA} \end{array} \tag{10}$$

Authorized licensed use limited to: ULAKBIM UASL - Bilkent University. Downloaded on July 13,2023 at 09:32:58 UTC from IEEE Xplore. Restrictions apply.

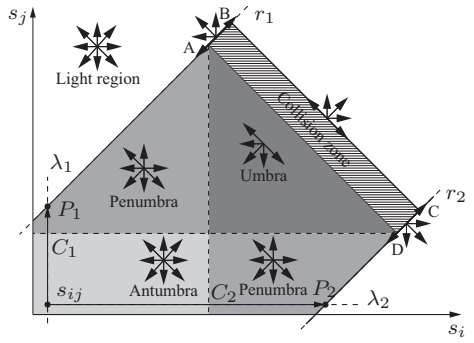

Fig. 2. Shadow sub-zones and corresponding actions allowed

For each action set U_r to be evaluated, the algorithm reads in which zone the coordination point projection falls (READ(CD_{ij}, s) in Alg. 3) and discards all the actions of U_r that do not respect the constraints imposed by the region. In Alg. 3 we refer to this operation with PRUNE$(Region, U_r)$. Further actions are forbidden in case they imply a movement of an AGV that has to implement an emergency stop. In this way, the emergency handling is embedded online in the coordination controller. If, after pruning, all the actions contained in U_r are discarded, the next subset U_{r-1} is considered. The evaluation of the action subsets terminates when, after pruning, the action set contains at least one action that satisfies the constraints induced by the shadow zones of each coordinate plane. We refer to this subset as the *valid action set*.

In general, when the evaluation stops, the valid action set contains more than one action. The final choice of the action to execute is made evaluating a cost function $C(u)$. This cost is an indicator of the total time spent by the vehicles to avoid the collision with the others and it depends on the way chosen to bypass each collision region on the coordination diagram. Consider a given configuration of the robots and a valid action set. For each s_{ij} falling in a shadow zone, two points in the plane are individuated. These points, indicated as P_1 and P_2 in Fig. 2, are the intersection between the two half-lines λ_1 and λ_2 (outgoing from the actual position in the coordinate plane, parallel to and directed in the positive direction of the coordinate axis) and the two rays r_1 and r_2 as depicted in Fig. 2. The distance between s_{ij} and the points P_1 and P_2 is proportional to the time required by two AGVs for bypassing the collision region traveling in a given direction. Thus, to each action moving toward P_1 (P_2) is associated a cost equal to the distance between s_{ij} and P_1 (P_2). On the other hand, to each action moving away from P_1 and P_2 (or standing in the same position) is associated a cost equal to the sum of the distances between s_{ij} and the points P_1 and P_2. This means that actions that don't tend to resolve the collision condition are penalized with respect to those that tend to escape from the shadow zones. Finally, a cost equal to the minimum distances between s_{ij} and P_1 and between s_{ij} and P_2 is associated to actions which yield an advancement of both vehicles. This means that the choice of bypassing the obstacle is postponed but, since both AGVs are moving toward their goals, these action are not penalized. In case the coordination point in the plane doesn't fall into a shadow zone, a zero cost is associated to all actions.

These calculations in Alg. 3 are indicated by the function COST(u, CD_{ij}). The total cost of the action u is defined as the sum of the costs associated to u on each coordinate plane. Once that each valid action has been assigned to a cost, the algorithm picks a minimum cost action. Thus, the chosen action u^* is given by:

$$u^* = \underset{u \in U_r}{\operatorname{argmin}} C(u) \tag{11}$$

Algorithm 3 Local planner

```
loop
    Read the current positions of the robots s_i
    r ← N                                   ▷ N active robots
    U_r ← ∅
    while U_r = ∅ do
        U_r ← ACTION SET(r)
        for all CD_ij do
            Region ← READ(CD_ij, s)
            U_r ← PRUNE(Region, U_r)
        end for
        r ← r − 1
    end while
    for all CD_ij do
        if Region = Shadow then
            for all u ∈ U_r do
                C(u) ← C(u)+ COST(u, CD_ij)
            end for
        end if
    end for
    u* ← ARGMIN(C(u))
    return u*
end loop
```

VI. SIMULATIONS

We have tested our approach running a simulation with 5 robots in MATLAB on a Intel Pentium D 3,20 GHz. In Fig. 3 some snapshots illustrating the coordination problem are reported. In the first one we have marked the goal positions $G1, \ldots, G5$. See also the attached video.

The vehicles are starting in random positions along given paths in order to consider the fact that the mission can be released asynchronously. We have also considered unexpected events by randomly blocking one vehicle for a random time interval. In Fig. 4 we show some planes of the GCD in which are displayed the shadows and the projection of the coordination path computed. The numbers p_i of partitions in which each path is split are 375,400,400,575 and 550 respectively. We have replicated 10 times the same test (i.e. the same paths, but with different starting positions and unexpected events). For the online composition of the diagrams and the definition of the shadow zones the algorithm on average takes 260 ms. The local planner algorithm, for the

Authorized licensed use limited to: ULAKBIM UASL - Bilkent University. Downloaded on July 13,2023 at 09:32:58 UTC from IEEE Xplore. Restrictions apply.

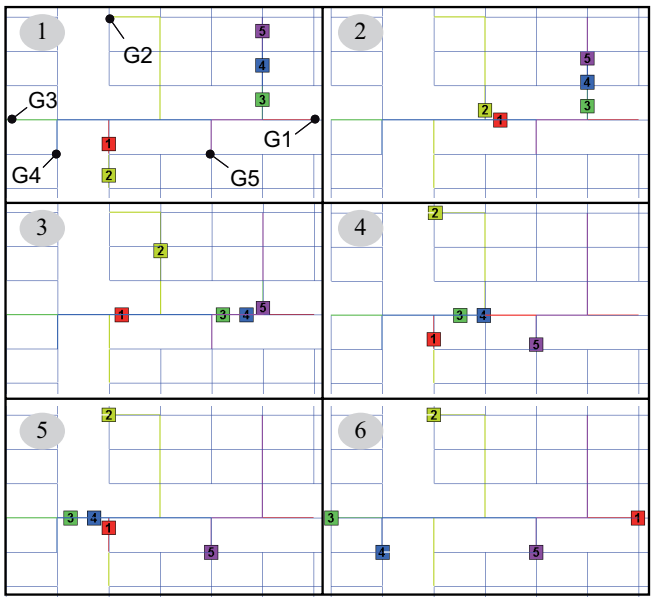

Fig. 3. Snapshots of the simulation

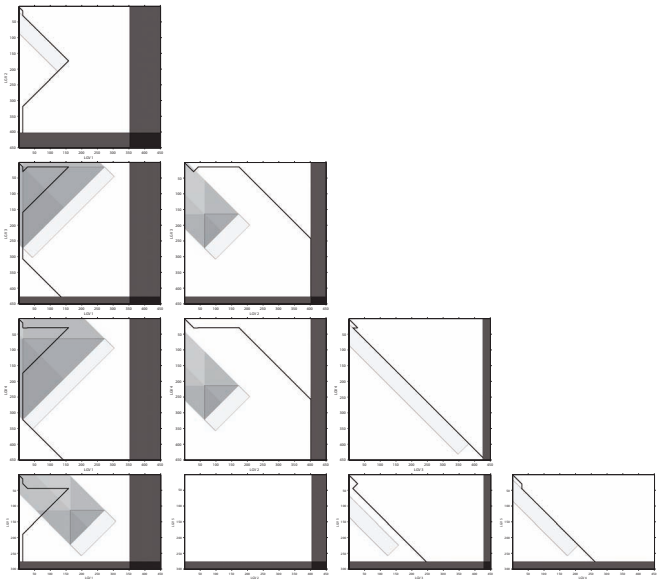

Fig. 4. The solid black line describes the projections of the coordination path within the coordination diagrams.

computation of a single action takes at maximum 8 ms (while the average is 2 ms).

VII. Conclusions and Future Work

In this paper we have proposed an algorithm for coordinating multiple AGVs moving on a predefined roadmap for an industrial application. Our algorithm is based on the use of coordination diagrams. Using the particular structure of the considered application we have been able to considerably reduce the time required for computing the coordination diagram. Furthermore, since only some kinds of collisions can take place, we have been able to develop a motion planner with a limited complexity and which can handle both collisions avoidance and emergency stops that can take place in industrial plants.

Future work aims at studying the complexity of the proposed algorithm from a rigorous point of view. On the experimental sides, we want to implement the proposed control strategy on a real industrial setup. In order to do this, we will need to relax some assumptions made since now. To handle the vehicle dynamics we will make the algorithm planning always some steps ahead with respect the current configuration. This is necessary also to let the vehicles stop if the communication with the central unit is lost.

References

[1] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[2] R. Alami, S. Fleury, M. Herrb, and F. I. F. Robert, "Multi-robot cooperation in the martha project," *IEEE Robot. Automat. Mag.*, vol. 5, no. 1, pp. 36–47, Mar 1998.
[3] L. Chun, Z. Zheng, and W. Chang, "A decentralized approach to the conflict-free motion planning for multiple mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, Detroit, Michigan, May 1999, pp. 1544–1549.
[4] S. Morinaka, T. Nishi, M. Konishi, and J. Imai, "A distributed routing method for multiple agvs for motion delay disturbances," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1986–1991, Aug 2005.
[5] Y. Guo and L. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, Washington, DC, May 2002, pp. 2612–2619.
[6] P. Svestka and M. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, Nagoya, May 1995, pp. 1631–1636.
[7] N. Wu and M. Zhou, "Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking," *IEEE/ASME Trans. on Mechatronics*, vol. 12, no. 1, pp. 63–72, Feb 2007.
[8] J. Peng and S. Akella, "Coordinating multiple double integrator robots on a roadmap: Convexity and global optimality," in *IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, Apr 2005, pp. 2751–2758.
[9] R. Ghrist, J.M.O'Kane, and S. LaValle, "Computing pareto optimal coordination on roadmaps," *Int. J. Robot. Res.*, vol. 24, no. 11, pp. 997–1010, 2005.
[10] S. LaValle and S. Hutchinson, "Path selection and coordination for multiple robots via nash equilibria," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, May 1994, pp. 1847–1852.
[11] ——, "Optimal motion planning for multiple robots having independent goals," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, Dec 1998.
[12] T. Simon, S. Leroy, and J. Laumond, "Path coordination for multiple mobile robots: a resolution-complete algorithm," *IEEE Trans. Robot. Automat.*, vol. 18, no. 1, pp. 42–49, Feb 2002.
[13] P. O'Donnell and T. Lozano-Perez, "Deadlock-free and collision-free coordination of two robot manipulators," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, May 1989, pp. 484–489.

Authorized licensed use limited to: ULAKBIM UASL - Bilkent University. Downloaded on July 13,2023 at 09:32:58 UTC from IEEE Xplore. Restrictions apply.