Decentralized Coordination of Automated Guided Vehicles (Short Paper)

D. Herrero-Perez
Department of Information and Communication Engineering, University of Murcia
30100 Murcia, Spain
dherrero@dif.um.es

H. Martinez-Barbera
Department of Information and Communication Engineering, University of Murcia
30100 Murcia, Spain
humberto@um.es

ABSTRACT

This paper approaches the issue of coordination of highly autonomous Automated Guided Vehicles (AGVs) working on an automated factory. These vehicles are used for goods delivery tasks between different points of the production system. The coordination is based on a decentralized architecture where each vehicle broadcasts the information about its state in the working environment, and by combining all these states in a local way, each AGV decides which action to take. The heuristic that allows the decentralized traffic control is based on a priority system, based on the current task, and a set of dangerous zones which are defined to avoid possible deadlocks, where mutual exclusion should be ensured. The process is somehow similar to that used by humans when circulating in cars: a set of rules and a set of signals/places. The interaction of many vehicles working on the same area under different collision conditions has been tested in a real industrial warehouse environment.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI– Coordination, Intelligent agents, Multiagent systems.

Keywords

Industrial Robots, Multi-Robot Systems, Automated Guided Vehicles

1. INTRODUCTION

The different approaches used to solve the coordination problems can be classified according to their architecture into two groups: centralized and decentralized.

The centralized architecture aims to solve the problem considering the whole system. Therefore, it is possible to find the global optimal solution, but the complexity to solve this problem is high, increasing exponentially with the number of robots, being quite difficult to implement in real time. Normally, the space of configuration of all the robots is combined by performing a search of the route of every robot together. Some works [7, 9] formulate the problem considering fixed or specified paths, and they are focused on calculating the velocities to avoid collisions and minimize the completion time of the trajectories. Fraichard [3] uses a centralized supervisor, which is periodically executed, for calculating the route the vehicles should follow using priorities for planning the routes and considering the static objects.

The decentralized architecture aims to solve the coordination problem only when there is some conflict between the different robots, i.e., each agent performs its tasks and only if these are dependents the conflict is handled. This architecture seems preferable to approach the multi-path planning, and the navigation of multiple robots given these trajectories, in applications with real-time restrictions. Bennewitz [1] uses priorities for the multi-path planning; the routes are calculated ignoring the others using an algorithm based on A*, and then the possible conflicts are solved by using the priorities to designate the order of the re-planning.

We use a decentralized architecture for solving the problem of coordination between n AGVs in an industrial environment. Each vehicle broadcasts periodically the information about its state, like location, task and destination. Using this information, each vehicle generates a route to its goal, considering the other vehicles as static obstacles, by using the search algorithm D*. Logically, this is not enough to avoid collisions between vehicles because all of them are usually in motion. Therefore, we use an obstacle avoidance method (*Polar Kinematics Bug*), a priority system and critical zones to define special regions with critical constrains. The critical zones are mutual exclusion zones, like corridors and docking areas are, which should be carefully handled to avoid deadlocks. This system has been validated in a real industrial environment with AGVs transporting pallets simultaneously between different points of production lines in a factory.

The paper is organized as follows. Section 2 is focused on the high-level representation for planning the tasks of the AGVs. Section 3 describes the path planning and the method used for avoiding the obstacles by the AGVs while they are navigating. The traffic control to coordinate the vehicles using priorities and critical zones are described in Section 4. Section 5 shows the experimental validation using some autonomous vehicles in a real industrial environment, and finally Section 6 presents some conclusions.

2. ENVIRONMENT'S REPRESENTATION

In order to operate, we provide a rough description of the floor plant to each AGV; a 2D CAD-like representation of the world which specifies the most relevant features of the environment, such as walls, reflective strip beacons for the

Cite as: Decentralized Coordination of Automated Guided Vehicles (Short Paper), D. Herrero-Perez and H. Martinez-Barbera, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16.,2008,Estoril,Portugal,pp.1195-1198.

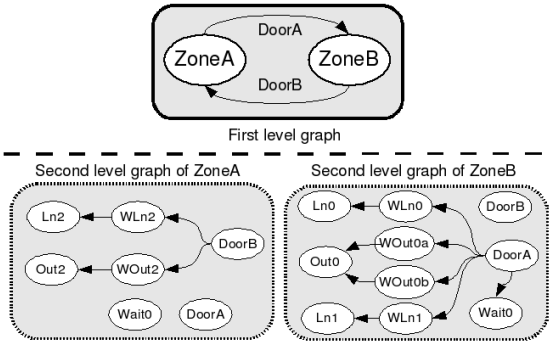

Figure 1: Example of topological representation: first and second level graphs of two zones

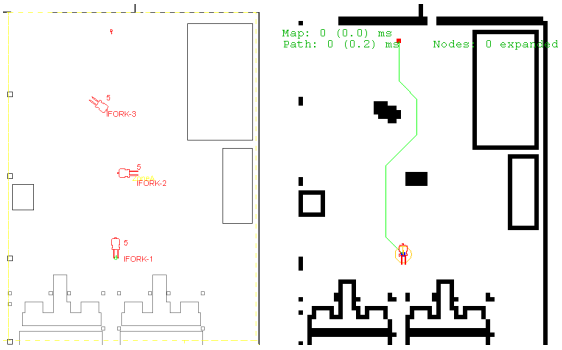

Figure 2: Path planning using D* method and considering the other AGVs as static obstacles

localization system, zones or areas, docking-points for loading or unloading and doors. This map is not intended to be an accurate representation of the environment, but the important details of it. Thus, it does not provide enough information for the navigation tasks. For this purpose, there are two fundamental paradigms for path-planning: the grid based paradigm [2] and the topological paradigm [4]. The complexity of grid based methods, both in time and space, presents a critic difficulty for task planning. On the other hand, topological maps are more compact permitting fast planning and joint use with high level planners and traffic coordination systems [5]. However, this kind of maps presents other problems, such as the correct identification of landmarks or the maintenance in large environments.

Because we have to deal with a relative large environment, divided in several zones, some of them connected through narrow passageways and with a number of relevant locations, a reasonable solution is the use of both approaches; a topological map is used for a high level description of the environment and a grid map for a local description of the environment. We apply a hierarchical vision, where [4] different maps are used depending of the level of the control architecture and where [2] several topological maps with different resolution are used in order to cover the entire area. In particular, we use a two-level topological map for storing the relevant places that the AGV could need to reach and representing how to move from different zones or rooms to others. This topological map is used for high level routing only. In the first level topological map, each node represents a relevant zone, typically a room-like area. Should a door or passageway exist between two zones, then an arc between the two nodes representing the zones is added. The arcs connecting the nodes are directed and weighted, and thus the direction can be used to force the traffic through the different zones in a given sense and the weights to express preferred directions. Each arc has a label associated, which corresponds to the passageway connecting both zones. In addition, a first level node contains a pointer to a grid map which covers the entire zone and it is initialized from the 2D world description and a pointer to a second level topological map. In this map each node represents a relevant place. In our current implementation these places are *doors*, *wait-points*, *dock-points* and *way-points*. The *doors* are used to assist the AGV in crossing doors and for implementing a door locking mechanism to avoid two AGVs crossing it simultaneously. The *wait-points* define locations to stop the robot for charging or servicing purposes. The *dock-points* are used to define the places where the loads are taking from or released in. The *way-points* are used to generate a path to maneuver the AGV. An arc connecting two places represents that the robot can navigate between those two places. In the same way as in the first level map the arcs are directed.

When a vehicle receives an order, like load or unloads a pallet located in a given place, a plan is generated to achieve the task. A plan typically consists on several subplans. First, the first level graph is searched to obtain a list of *doors* and *wait-points* needed to reach the zone where the vehicle should navigate. Then, the second level graph is searched to obtain the destination points, in particular, *way-points* and *dock-points*.

Figure 1 shows a simple example of a topological map. In order to clarify the system, we describe the plan generated for docking in *Out0* when the vehicle is located in the *ZoneA*. The process for generating the subplans is: i) it is computed the first level topological path to reach the *ZoneB*, where the goal *Out0* is located, crossing the *DoorA*, then ii) it is computed the second level topological path, and finally iii) the final plan is composed as a sequence of behaviors: <Navigate (*DoorA*)>, <Cross (*DoorA*)>, <Navigate (WOut0a/WOut0b)> and <Dock (Out0)>.

The use of these topological maps, task planning and coordination results simple but effective way to organize navigation, because the access to zones, the connection between zones, and the connection between points can be constrained. In addition, the path planning is also simplified because the overall area is divided in several zones, reducing the complexity of the search, both in time and space.

3. NAVIGATION SYSTEM

In order to navigate, the AGVs calculate the path dynamically using a search algorithm D* over a fuzzy grid map, what makes the system more flexible and facilities the implantation and the starting of the whole system. When the vehicle detects a possible collision, an avoidance obstacle behavior is activated. This behavior uses a reactive method adapted to the configuration of the vehicles, which are tricycle type. When there is a conflict between some AGVs, this is solved using re-planning and obstacle avoidance. These techniques are detailed below.

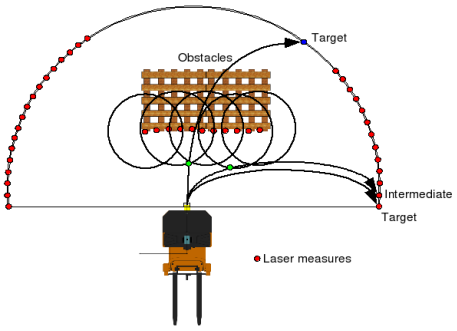

Figure 3: Polar Kinematics Bug method using tricycle kinematics

3.1 Trajectory planning

In order to navigate inside of each zone, a representation based on a fuzzy grid map [6] is used. Each cell of the map has two values associated; the degree of certainty that the cell is empty, and the degree of certainty that the cell is occupied. These values are fused to obtain the fuzzy grid used for the navigation of each AGV. This map is initialized using the values of the *a priori* map, and it is updated using the information of a laser rangefinder.

Because of one local fuzzy grid map is used for each zone, the computational time is decreased for each zone. The point-to-point trajectory, between the location of the vehicle and the point to reach in the plan, is calculated using the search algorithm D*. This method re-calculates the path-planning by a local way when there are changes in the fuzzy grid map without re-calculate the whole route again.

The location of each AGV is broadcast for the coordination between vehicles. This information is also used for mapping, and then considering the other vehicles as static obstacles. In order to include the other vehicles in the fuzzy grid map, we are modeling each AGV as a bounding box, Figure 2, considering this place to generate the path of the robot, i.e., avoiding that the path generated cross the area occupied by the other vehicles.

3.2 Obstacle avoidance

In order to avoid the obstacles, we are using a reactive method based on the principles of the method *PolarBug* [8], which uses the laser rangefinder readings to provide a representation of the areas to avoid. The method makes many assumptions to simplify computations and provide a fast solution because obstacle avoidance in industrial robot has logically strong real time restrictions. However, this method is designed for mobile robots using a differential kinematics model, and thus the solution is not valid for vehicles using a tricycle kinematics model, like our AGVs use. Therefore, we have developed a similar method considering the kinematics constraints of the vehicle, namely the *Polar Kinematics Bug* method.

When a trajectory is generated for the navigation, a point is calculated given a distance to the AGV, this point is the *Target* where the robot is directed to. The points detected closer are expanded using a circumference with a safety radius that depends of the size of the AGV. The *Polar Kinematics Bug* method, Figure 3, traces circular-segments from the robot's location to the *Target*, and when this circular-segment intersects with some circumference, a collision is detected. In this case, we should to calculate a point, *Intermediate Target*, which does not intersect any circumference; this is the new point that the robot might to reach to avoid the obstacle. Because of there are several possible points, we select the more optimal, for instance, the closer to the *Target*. In the case that we are not able to find any solution, the AGV might turn with maximum velocity to avoid the obstacle or stop in the case the obstacle is too close.

Because of the angle of view of the laser rangefinder is limited to 180 degrees and this device is installed in the frontal part of the vehicle, some lateral obstacles are not visible and the vehicle could collide. In order to avoid this problem, a local buffer is used to store some close points that are not visible from new locations, and these point are likewise handled.

4. DECENTRALIZED COORDINATION

In order to coordinate the robots, we are using a decentralized architecture in which AGVs navigate independently and broadcast the information about their state. The path planning method, which considers the other vehicles, and the obstacle avoidance are enough to avoid collisions between vehicles, but it is not enough for avoiding deadlocks: AGV traffic jams where two vehicles are waiting each other for finishing a task while they stand still. Thus a set of coordination rules are mandatory. Each AGV periodically broadcasts over a wireless link a coordination tuple, which contains the following information: location, priority, current task and the critical zone that is currently occupying.

The priority of each vehicle is calculated considering its current task, from minor to major: *stopped*, *giving way*, *navigating*, *docking* and *crossing door*. Tasks with higher priorities are the tasks that imply carrying a pallet and the change of zone, i.e., crossing doors. The state of the vehicle also influences in its priority. The possible states of the vehicle are: *stopped*, *waiting*, *giving way* and *normal* state.

Due to the nature of a decentralized architecture, it is difficult to avoid collisions between vehicles when they are all moving independently, i.e., they can suddenly change goal point or current path. Because of this, we only allow the navigation when the distance between vehicles is larger than a given safety distance. Therefore, each vehicle calculates the distance to the others using the broadcasted positions, and then decides if it should give way according to the priorities of each one. The vehicle with lower priority should give way to the others. In case of having the same priorities, i.e., their assigned tasks are of the same complexity and relevancy, the tie is broken by using their network addresses. When all the vehicles inside a dangerous zone are stopped, the one with the higher priority is able to reach its destination point by re-planning and obstacle avoidance. Despite the method seems to have all but one AGV non-navigating, it is important to note here that this is only applied when they are very close each other, and during normal operation this only occurs a few times.

When a vehicle is performing a docking or undocking operation, it needs some free space to perform the maneuver. In case those two vehicles compete for the same maneuvering area, i.e., two close-by docking places, a deadlock is possible. In order to avoid it, these maneuvering areas are manually designated as critical zones, i.e., mutual exclusion zones. The first vehicle that enters one of these critical

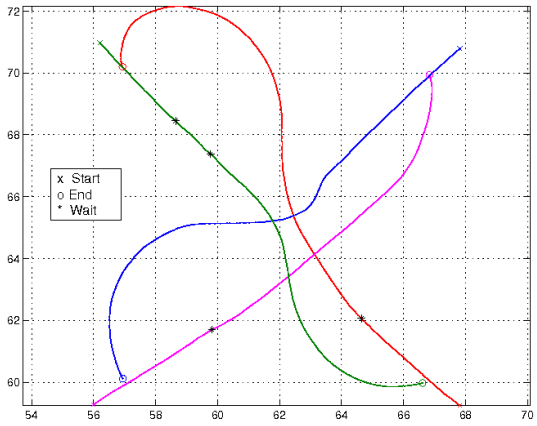

Figure 4: Trajectories for four crossing vehicles

zones broadcasts a coordination tuple to indicate that it is the only vehicle permitted in that zone. The other competing AGVs stay still outside the critical zone until the one inside it leaves the zone, which is designated broadcasting a coordination tuple. The same method is used for doors and narrow passageways. They are designated as critical zones.

5. EXPERIMENTAL VALIDATION

The coordination of the AGVs working on the same area has been tested in an industrial warehouse environment [1]. The method has been tested reproducing many collision conditions and evaluating the interaction between the vehicles involved in the possible collision. Because normal operation does not produces collision conditions often enough, we evaluate them by forcing the AGVs follow collision routes. We show an example with four vehicles in two perpendicular trajectories with different senses per trajectory. The four AGVs are distributed at the corners of a square, and each AGV has to reach the opposite corner, leading all the vehicles to meet in the center of the hypothetical square. Figure 4 shows the trajectory followed by each vehicle with their stopping points. In this case we need many stops to solve the conflicts, mainly due to the complexity of the problem. We can observe that the vehicle of the blue trajectory is the first to reach the goal point without stopping. Then, the vehicles of the red and green trajectories restart again, and when they are at the safety distance, the vehicle of the green trajectory stops again. When the vehicle of the red trajectory avoids the vehicle of the green one, this starts again and it reaches the goal. Finally, when dangerous condition is not satisfied for the vehicle of the pink trajectory, the fourth vehicle starts the navigation again and all the vehicles are able to reach the destination points.

6. CONCLUSIONS

This work has presented a decentralized architecture for the coordination of several vehicles which is specially focused to transportation tasks. Although assuring optimal paths in a fully decentralized architecture is very difficult, we aim to satisfy avoiding deadlocks.

[1]Videos of the multi-robot in warehouse operation and experiments described in the paper can be found at http://robolab.inf.um.es/ifork/

We are able to coordinate a considerable number of AGVs by using a heuristic which is based on priorities and critical zones while avoiding collisions and deadlocks in mutual exclusion zones. In all situations we only allow one of the conflicting AGVs to enter or navigate the mutual exclusion zone.

For this method to work, we need a high degree of autonomy in the AGVs. This is accomplished by using a D* path planning algorithm and a reactive obstacle avoidance method, the *Polar Kinematics Bug*, for low level navigation tasks. For high level navigation tasks, we represent the environment by using a two-level topological map, which greatly reduces the complexity of path planning in such large areas as encountered in industrial scenarios.

7. ACKNOWLEDGMENTS

This work has been supported by CICYT project DPI2004-07993-C03-02. The authors would like to thank the company Frutas El Dulze S.A. for the accessing to the facilities required for this research.

8. REFERENCES

[1] M. Bennewitz and W. Burgard. Finding solvable priority schemes for decoupled path planning techniques for teams of mobile robots. In *Proc. of the International Symposium on Intelligent Robotic Systems (SIRS)*, 2001.

[2] K. de Meyer, O. Lemon, and U. Nemhzow. Multiple resolution mapping for efficient mobile robot navigation. 1997.

[3] T. Fraichard and Y. Demazeau. Motion planning in a multi-agent world. In *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 137–154. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[4] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[5] Q. Ling and H. Wen-Jing. Scheduling and routing algorithms for agvs: a survey. 1999.

[6] G. Oriolo, G. Ulivi, and M. Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Trans. on Systems, Man, and Cybernetics*, 28(3):316–333, 1998.

[7] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. In *WAFR*, pages 221–238. Algorithmic Foundations of Robotics V, 2002.

[8] R. Schraft, B. Graf, A. Traub, and J. D. Polarbug - ein effizienter algorithmus zur reaktiven hindernisumfahrung im dauereinsatz. In *In Proceedings of Autonome Mobile Systeme*, pages 96–101, 2000.

[9] T. Simeon, S. Leroy, and J. Laumond. Path coordination for multiple mobile robots: a resolution complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49, 2002.