Path Planning and Collision Avoidance Regime for a Multi-Agent System in Industrial Robotics

MSc. Ivan Gochev, MSc. Gorjan Nadzinski, Prof. DSc Mile Stankovski,

Faculty of Electrical Engineering and Information Technology – University of Ss Cyril and Methodius, Republic of Macedonia, Skopje

ivang@feit.ukim.edu.com.mk; gorjan@feit.ukim.edu.mk; milestk@feit.ukim.edu.mk

Abstract: *Industry 4.0 which creates "smart factories" present a recent trend in development. The area represents a merge of cyber-physical systems and Internet of Things, which aims to improve manufacturing technologies. Industry 4.0 strives to boost the algorithms and technologies used in industrial processes during the production processes, process preparations, and products delivery. Our intention is to improve the robotics transport system in factory floor. There are a lot of different research approaches in this area for further improvement. Our approach is to deal with multi-agent systems control, because of the great potential it has in practical applications in industrial robotics. The strive for minimizing the work time and maximizing the efficiency can be satisfied through the usage of multiple coordinated agents to achieve the end goal. The use of Automated Guided Vehicles (AGVs), combined with concepts for task planning of multiple agents broadened during the late 20th century. In this paper, the multi-agent system consists of several mobile robots, in other words platforms, which need to transport materials in a workhouse. The goal of each mobile platform is to carry the specified object to a set position. These appointed goals are not predefined and can be changed according to the needs of the user. Working in a dynamic environment, numerous agents with different tasks to complete can be exposed to many obstacles which may be the cause of accidents. For this reason, a careful path planning is required in such environments. The suggested path planning algorithm for this system is A*. A* is a fast path finder, which can navigate quite well in a planar environment, but it is not favorable for dynamical settings. Therefore, a combination of the A* algorithm with a collision avoidance method is proposed for overcoming these difficulties. By doing this, the A* algorithm is expanded to work in dynamical situations and can assure the convergence of any agent towards their goal. This fusion of both, the path finding algorithm and the collision avoidance method, can aid the cooperation of the agents and improve the efficiency of the system as a whole.*

Keywords*:* Industry 4.0, smart factory, factory floor, multi-agent systems, mobile robotics, A* algorithm, collision avoidance.

1. Introduction

Recent technological advances indicate that we are witnessing the dawn of an era of Internet of Things [1]. Internet of things is broadening its possibilities in industry as well, providing more flexibility and maneuverability in production. Intelligent agents creating multi-agent systems provide better performance instead of single agents completing different tasks. Not only that, but multi-agent systems can even found themselves in a situation to complete tasks which other individual agents would not be able. The control of multi-agent systems is presented in [2]. The industry strives to create manufacturing systems that would be fully autonomous, in order to increase the time and capacity of production. The first step to automate the environment is to plan the trajectory that each robot has to follow to reach the end goal. A* is one of the most used algorithms for finding optimal paths [3] [4].

In this paper, a small representation of a smart factory is presented where automatic guided vehicles (AGVs) are accomplishing different tasks. The main idea is to design an autonomous warehouse, where mobile vehicles (agents) deliver packages. For that very reason in this paper path planning in the environment and preventing collisions between agents of the multi-agent system is accented the most. In section 2, the environment (warehouse) layout and the multi-agent system working in those settings are presented. The following sections, Section 3 and Section 4 explain the planning and control algorithms used to create the autonomous work of the whole system, and the avoidance rules of agents and static environment obstacles. Section 5 briefly shows the function of the different aspects of in the whole project and the results from the implemented algorithms. In Section 6 a conclusion is given, as well as the outlook for future work.

2. Environment settings and Multi-agent system

V-REP and factory settings

The multi-agent system along with the environment in which it operates is represented in the programing package V-REP (Virtual Robot Experimentation Platform), a program for prototyping robotics systems. V-REP is the first step for system design and algorithm testing, because it offers a flexible program with a lot of possibilities. The free open source educational version of the robotics toolbox is provided by Coppelia Robotics. Robotic simulators represent the connection between artificial creatures' theory and robotics. They provide an additional guarantee that cognitive framework developed can be applied to real robots, with need of minor adjustments needed [5].

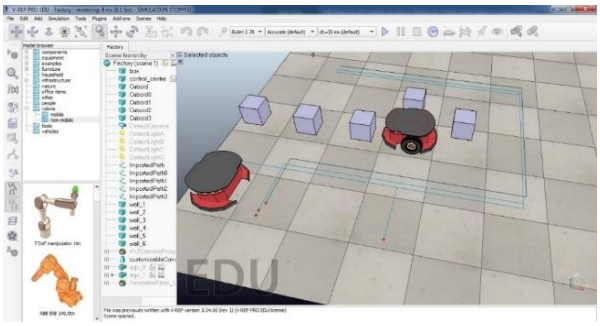

Fig. 1: V-REP Simulation interface

The simulator is running from Python code where most of the main algorithms are written. The basic idea is that the hardware aspects of the system are realized in the V-REP simulation interface, whereas the logic is implemented in Python. V-REP is the server side, and Python is the client side. Both programs are communicating through an Application Programming Interface (API). The remote API modus operandi represent a set of functions, subroutines, protocols and tools for building application software.

Multi-agent system

The multi-agent system consists of automatic guided vehicles (AGVs), which are by definition the agents of the system. They are equipped with 16 ultrasonic sensors and 10 force sensors. The end goal of each mobile robot of the multi-agent system is to hand out a specific object to the correct position in the warehouse. They need to navigate in a time differing and dynamic environment. The system is made up of *N* automatic guided vehicles which distribute certain goods in a warehouse and the best way to automate that is to implement some kind of planning logic. Having an optimal planning

algorithm would enhance the performance of the system as a whole and therefore a heuristic pathfinding algorithm is proposed. The suggested algorithm is A* (A-star) and this is only the first step in designing the whole factory floor. To ensure the convergence towards the final objective of the whole project, the pathfinding algorithm is enhanced with a collision avoidance regime, which prevents any accidents that might occur.

A multi-agent system is a system composed of multiple interacting intelligent agents in a specific environment. The agents of the system are small mobile robots generated in V-REP called the "Pioneer" model. (Fig. 2)

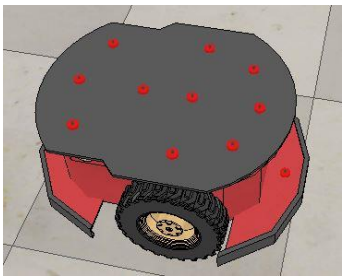

Fig. 2: Pioneer robot model

The mobile robot is equipped with ultrasonic and force sensors. Based on the information from the ultrasonic sensors the specific agent will either enter a collision avoidance regime or it will continue finishing the designated task. When the force sensors are activated, the agent will know that a specific object to transport is given to it and will start completing the task. The algorithms are tested for two agents in the environment, but there are no restrictions on how many agents can be involved if the hardware can endure.

The mobile robots work in a warehouse which is 60 meters long and 20 meters wide. The map of the environment is represented as a grid with 120x60 elements. One cell of the grid is 0.5x0.5m^2. Each vehicle has information about all other objects in the environment, including the other agents, all seen as rigid bodies. A rigid body in space is defined by 3 positions (*x, y, and z coordinates)* and 3 angles (α, β, γ) which give the orientation of the body around a specific coordinate system. When the position of the agent is known the cell coordinates can be obtained through the following equations:

$$\begin{cases} i = \left\| \left[\frac{x}{cell\_length} + environment_{length} - 1 \right] \right\| \\ j = \left\| \left[\frac{y}{cell\_length} + environment_{width} \right] \right\| \end{cases}, \quad (1)$$

where *x* and *y* are the agent coordinates, $cell\_length$ is the length of each cell in the grid, $environment_{length}$ and $environment_{width}$ are the length and width of the whole factory floor (an offset of the grid), and i and j are the obtained cell coordinates. Having the environment mapped as a 120x60 grid, and translated to a graph, the pathfinding algorithm can easily navigate in it and find the optimal trajectory [3].

3. Path planning

One of the most used algorithms for finding an optimal path in an environment is the A* (A-star) algorithm. This is a heuristic search algorithm for graph traversal. It minimizes a cost function for each neighboring node and expands to the nodes with the smallest function value. The algorithms' efficiency depends on the defined heuristic function. The heuristic function is the main decision maker in how the algorithm goes over the vertices of the graph. Because of the grid like representation of the environment, the chosen heuristic is the Manhattan heuristic, given with the equation:

$$h = \left| cell_{1_x} - cell_{2_x} \right| + \left| cell_{1_y} - cell_{2_y} \right| \quad (2)$$

The two parts which construct the A* algorithm are the heuristic and distance between the points in the graph. To complete the whole algorithm, it is needed to get the distance between the points. The distance is calculated as Euclidean with the equation:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (3)$$

To sum up everything, the A* is storing each neighbor in a priority queue and is looking for the nearest node with minimal f, where f is:

$$f(n) = d(n) + h(n), \quad (4)$$

where n is a neighboring node of the current node in the graph. Fig. 3 shows the generated path from the algorithm for specific settings.

However, there is a possibility for the A* algorithm to generate two or more trajectories for different agents that can intersect, leading to a collision between them. To avoid this the algorithm is enriched with a force model collision and obstacle avoidance regime presented in the following section.

4. Collision avoidance

To increase the certainty of the operation of the multi-agent system in the specific environmental settings, the planning algorithm is augmented with a collision avoidance method. The regime is divided into two parts: prevention of collision between the agents of the system and obstacles avoidance. In the following subsections the proposed methods are shown.

Collision avoidance between agents

Every agent of the system is equipped with ultrasonic sensors which construct a reaction zone around each agent. As soon as another agent is detected in the safe zone, both of them activate the code for collision avoidance. This upgrade enables the agents to quickly react when they are nearing each other and create a roundabout along the generated trajectory, instead of stepping back and changing directions which would be more time consuming. The collision avoidance is based on the force and momentum models of the agents seen as a rigid body [6]. This way the control rule is a whole of a fast path planning algorithm and a quick method for reaction if a potential collision is detected.

As said before, the method is based on the force and momentum models of the agents seen as a rigid body. This means that as soon as two or more agents enter each other's safe zones they become "risky neighbors", and based on the force vectors the new trajectory for the agents is updated with the following equation:

$$\overrightarrow{new_{position}} = \overrightarrow{current_{position}} + C * \widehat{new_{direction}}, \quad (5)$$

where the current position of the agent are the *x* and *y* coordinates. The new direction vector is determined from the direction of the force vector needed to be applied in order for the agent to reach the goal. The amount the new position is updated relies on the weight factor *C*, which is tuned by the user. Two force vectors are needed to complete the control law. Those are the external force vector and the force vector needed to reach the goal. The external force vector represents a sum of the force vectors of all agents in the reaction zone (risky neighbors).

To define whether a force or momentum model will be used, the formulated external force vector and the goal force vector are used to calculate the similarity between them. This similarity states how the agents are set in the environment and how they are moving. Having this information, both agents can move in a linear direction (force model) or rotate around each other (momentum model).

Obstacles avoidance

The obstacle avoidance rule is based on the Braitenberg approach for mobile robot navigation. This method can also be used instead of

the force and moment models for constructing a roundabout between agents, but both algorithms are implemented since the first method preserves the movement of the agents. The deviations from the original path with the first method do not have a large impact on the performance of the system, whereas the Braitenberg method would completely change the direction of the agents, leading to the need to regenerate a new path. The Braitenberg method is used for obstacle avoidance between an agent and other static objects in the environment, such as: walls, boxes, and conveyor belts. The outputs of the ultrasonic sensors directly affect the movement of the vehicle. When the detected obstacle is different than another agent, this part of the control code is activated. The measured distance affects the vehicle motors speeds proportionally to specific weight coefficients. The coefficients can be tuned in order to make the avoidance faster or slower. Although higher coefficients will make the avoidance faster, it will also make the movement of the vehicle more oscillatory. The motor speeds are modified as in the following equation:

$$\widehat{v_m} = v_m + B_c * \bar{s}, \quad (6)$$

where v_m is the motor speed, B_c is the braitenberg weight coefficient, $\bar{s}$ is the normalized data from a specific ultrasonic sensor. The normalization is a function of the minimum safety distance and the maximum detection radius.

5. Simulation

There are 3 aspects that have to be taken into consideration, which are: path planning, obstacle avoidance and agent avoidance. In Fig. 3 we can observe the generated path which the vehicle has to go over. It can be seen that the algorithm correctly finds the optimal path.

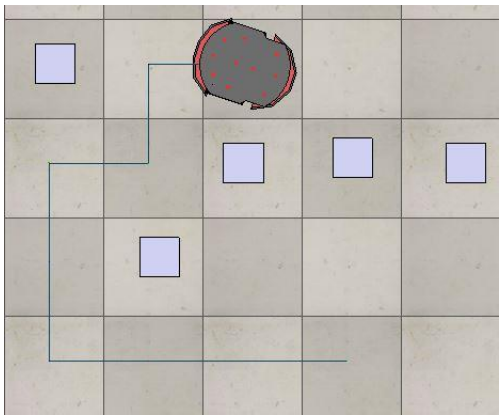

Fig. 3: Generated path from the algorithm

As it can be seen the mobile robot needs to reach the fourth cell in the bottom row. The A* algorithm generates the correct shortest path, instead of a longer one which would make the vehicle circle around the terrain.

Fig. 4 presents the Braitenberg algorithm for obstacle avoidance, where a vehicle switches the direction in order to escape the obstacle.

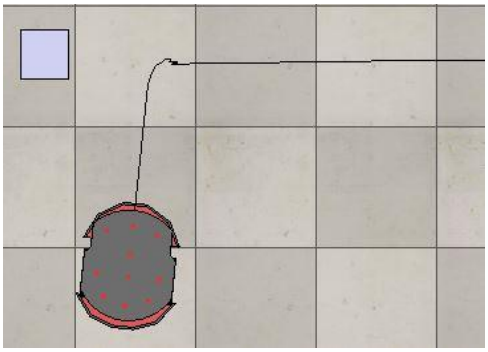

Fig. 4: The vehicle avoids the barrier in front of it

Fig. 5 shows the *x, y,* and *z* coordinates of the vehicle when it avoids the obstacle.

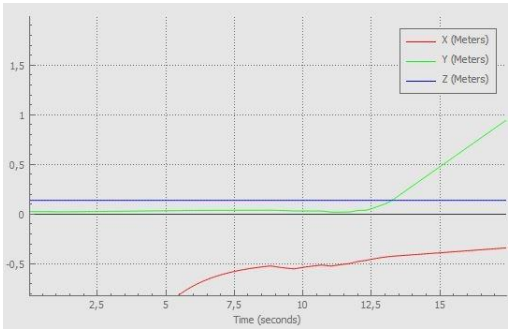

Fig. 5: x, y, and *z* coordinates of the vehicle avoiding the obstacle with the Braitenberg algorithm

The graph shows that the mobile robot successfully changes the path and avoids the static obstacle.

The avoidance between agents is shown on Fig. 6, where if two agents are moving in a direction they construct a roundabout each other and continue moving.

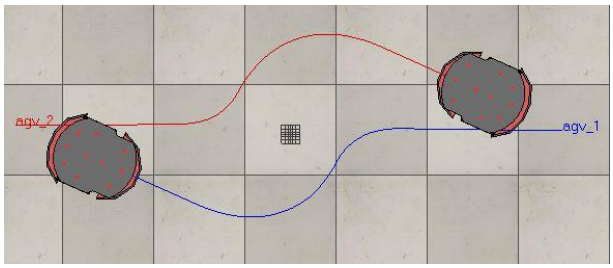

Fig. 6: Agent avoidance

The next figure (Fig. 7) shows the symmetry between the *x, y,* and *z* coordinates of the agents.

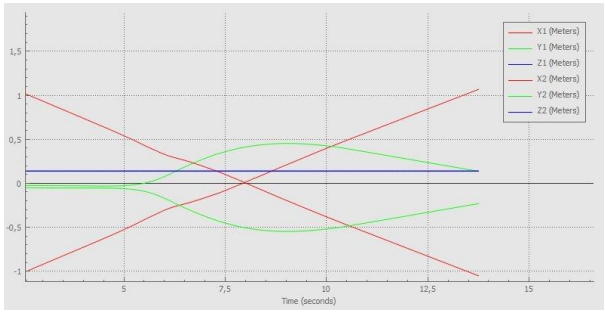

Fig. 7: Coordinates of the agents while avoidance

The symmetry of the mobile robots' coordinates follows the reconstructed path while avoiding other agents shown on Fig. 6.

Figures 3, 4, and 6, show that the algorithms are functioning correctly and giving the correct task for the agents to complete. This states that the three main aspects of the project are tackled, which leads to the project functionality.

6. Conclusion and outlook for future work

In this paper a prototype of a smart factory is presented with accent put on the path planning in the specific environment settings. The environment is mapped as a 2D grid represented as a graph where the objects are stationed and the automatically guided vehicles can operate. The proposed path planning algorithm is A*, a heuristic search algorithm which is used for graph traversal. The algorithm generates a path for each vehicle on factory floor, to the end position the vehicle needs to reach. To ensure the operation of the whole multi-agent system the A* algorithm is implemented together with a collision avoidance and obstacles avoidance rules.

Future work in this area consists of further upgrades of the proposed algorithms, or switching to dynamical pathfinding algorithms which can significantly improve the multi-agent systems' performance and even replace the collision avoidance regime. Another idea includes finding a way to modify the graph vertices which represent the environment in such way that the generated path will avoid the static obstacles [7]. The Braitenberg algorithm used for obstacle avoidance can be upgraded to be used in agent avoidance as well, which would lead to less computational power used if there are two algorithms instead of three. This can be achieved if the Braitenberg algorithm is modified so that the sensors values influence the actuators in a symmetrical way, which would lead to the same outcome as the force and moment model method. Furthermore, a wide variety of data can be obtained in the environment in which the multi-agent system operates. The information from those data sets can be used to improve the performance of the multi-agent system.

ACKNOWLEDGMENTS

The work was funded by the Faculty of Electrical Engineering and Information Technologies - University of Ss Cyril and Methodius, Republic of Macedonia, Skopje, through the ERESCOP Project.

References

[1] Feng Xia[1], Laurence T. Yang[2], Lizhe Wang[3], Alexey Vinel[4] "Internet of Things," [1]*School of Software, Dalian University of Technology, China,* [2]*Department of Computer Science, St. Francis Xavier University, Canada,* [3]*Indiana University, USA,* [4]*Tampere University of Technology, Finland*

[2] Shamma, J.S. (2007) *Cooperative Control of Distributed Multi-Agent Systems*, Wiley Online Library

[3] Peter Yap, "Grid-Based Path-Finding," Department of Computing Science, University of Alberta Edmonton, Canada

[4] Xiao Cui, Hao Shi"A*-based Pathfinding in Modern Computer Games," *School of Engineering and Science, Victoria University, Melbourne, Australia*

[5] Lucas Nogueira, "Comparative Analysis Between Gazebo and V-REP Robotic Simulators," School of Electrical and Computer Engineering Universidade de Campinas

[6] Victor Casas, Andreas Mitschele-Thiel, Mehdi Harounabadi, "On the Emergence of Virtual Roundabouts from Distributed Force/Torque-based UAV Collision Avoidance Scheme," 2017 13th IEEE International Conference on Control & Automation (ICCA) July 3-6, 2017. Ohrid, Macedonia

[7] Tomas Lozano-Perez, Michael A. Wesly, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," IBM Thomas J. Watson Research Center