IOP Conf. Series: Materials Science and Engineering **227** (2017) 012061 doi:10.1088/1757-899X/227/1/012061

Modelling cooperation of industrial robots as multi-agent systems

P Hryniewicz[1], **W Banas**[2], **K Foit**[3], **A Gwiazda**[4] **and A Sekala**[5]
[1-5]Silesian University of Technology, Faculty of Mechanical Engineering, Institute of Engineering Processes Automation and Integrated Manufacturing Systems, Konarskiego 18A, 44-100 Gliwice, Poland

E-mail: waclaw.banas@polsl.pl

Abstract. Nowadays, more and more often in a cell is more than one robot, there is also a dual arm robots, because of this cooperation of two robots in the same space becomes more and more important. Programming robotic cell consisting of two or more robots are currently performed separately for each element of the robot and the cell. It is performed only synchronization programs, but no robot movements. In such situations often placed industrial robots so they do not have common space so the robots are operated separately. When industrial robots are a common space this space can occupy only one robot the other one must be outside the common space. It is very difficult to find applications where two robots are in the same workspace. It was tested but one robot did not do of movement when moving the second and waited for permission to move from the second when it sent a permit - stop the move. Such programs are very difficult and require a lot of experience from the programmer and must be tested separately at the beginning and then very slowly under control. Ideally, the operator takes care of exactly one robot during the test and it is very important to take special care.

1. Introduction

Programming robots move in the same space at the same time requires synchronization of the movements of robots [3,4] and fully cooperate with robots. Even a small delay may cause a collision [1, 2]. It is very difficult and requires a lot of time to program the cell. The cost of such a program is very high, and the work is very dangerous. When robots will operate as agents can control not only their movements but also movements of another robot. This is a way to avoid collisions and better cooperation robots. The model of cooperating robots perform a task that is very difficult or even impossible for robots working separately [5, 16].

The most popular algorithms for trajectory determination are A * algorithm and D * algorithm. Both algorithms are designed for single robots.

The paper presents the method of adaptation of A * algorithm to the needs of two-arm robot.

2. Algorithm A* for Occupancy Grid Map

Before you apply the A * algorithm it is necessary to properly prepare a map. In the space described by the map will move the robot. This method is used when the space described is compact and the movement of arms is possible at every point of the map. In this case, creating a map is very simple and all is sectioned. Each section is a rectangle that has 8 neighbours and can be accessed by one of its neighbours.

Figure 1. Tools for creating an occupation map.

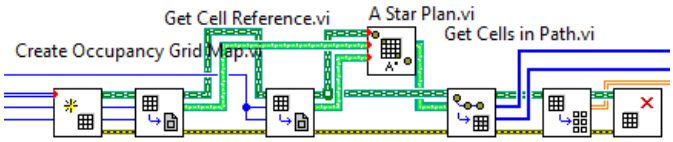

Figure 2. Grid map creation and trajectory planning program.

Map can be obtained in different ways. It would be most convenient to imprint CAD software [10, 11], in some cases, the map is built using a vision system [14]. The easiest way is to build a two-dimensional array and enter values into it, but it's uncomfortable and time-consuming. For this reason, for testing, a two-dimensional array of binary variables was built. This array is located in the left part of figure 3. Enabled bits mark an obstacle, disabled bits indicate possible routes.

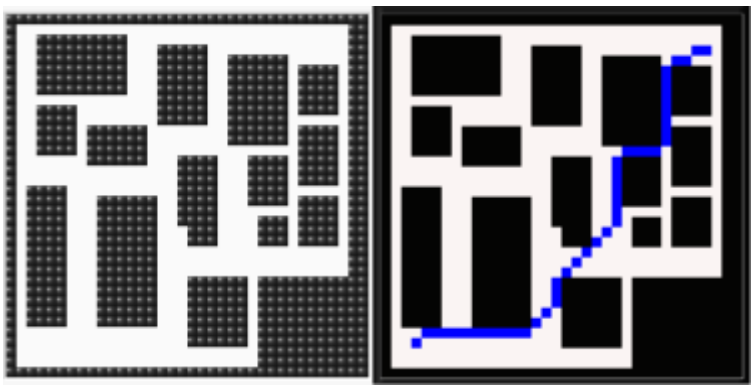

Figure 3. Grid map creation and obtained path.

The prepared map is entered as the input of the "Create Occupancy Grid Map" tool in figure 2. This tool was selected and the menu "Occupancy Grid Map" figure 1 and inserted into figure 2. On grid map, it is necessary to mark the starting point and the end point, that is done with the "Get Cell Reference" tool. Now the A * algorithm can work. The results of the algorithm can be read by the "Get Cells in Path" tool.

The result obtained as a path is shown in figure 3. It is easy to note that the path is the shortest, but the robot moves in a space where it is very tight. In such places the robot should slow down, Because the robot as a result of vibration [6, 7] can collide. It is possible by changing the cost of crossing the cell.

Figure 4. Grid map creation and obtained path.

The "Enqueue Change to Cell Cost" tool changes the cost of moving through cells. The higher cost of crossing certain cells results in a different route figure 4. This is really useful for determining path. Sometimes the shortest route is not the best because it requires a lot of programming work and can cause collisions. Choosing the right cost for each cell will make it easier to find the optimal and safe path. In the example in figure 4, the cost of not all cells was introduced, since this is not the main goal but only an indication of further possibilities for developing the method.

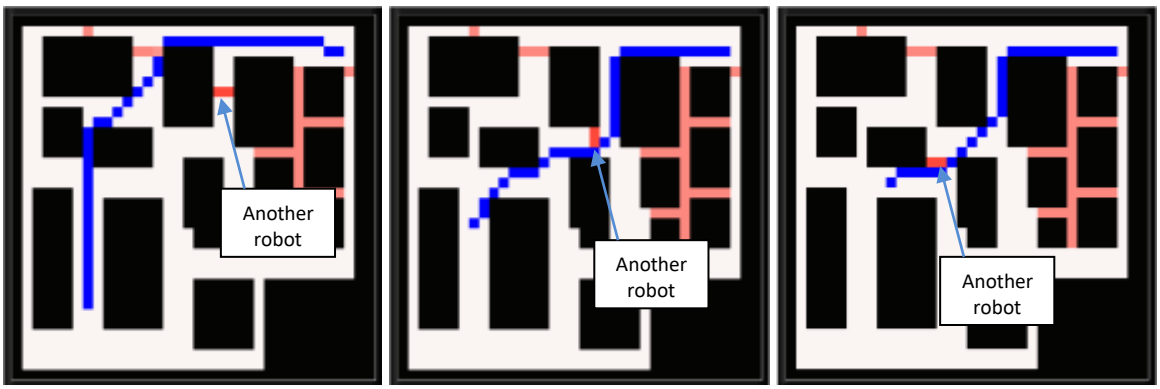

Figure 5. Grid map creation and obtained path.

Using the same tool, it can be placed on a map another robots. By changing the cost of cells in which other robots are currently located, the robot recognizes them as obstacles and sets a path based on the new map. Figure 3 shows 3 example paths at different stages of movement. It's easy to note that when the road is busy the robot changes its path even if it is far away and so far it may turn out that the road will be free.When the road again makes a free robot again change the path. In some cases, the robot may follow a different path and then turn around or follow a different path.

This method always gives a result, even if there is no road crossing, the method will minimize the damage but to reach goal. The main disadvantage of this method is the exponential increase in computation with increasing the size of the map.

3. Algorithm A* for Directed Graph

Directed graph is designed for more expansive layouts. In this case, the whole area of the robot's activity is not saved, only the selected points and costs of transition between them [8]. In this method, data entry takes a long time, each vertex of the graph must be named and enter its coordinates and then enter the cost of the path between them.

For the method tests a directed graph was constructed, it is composed of 400 vertices figure 7.

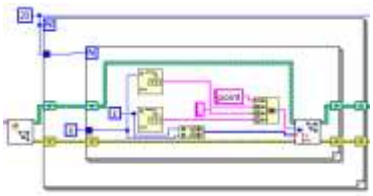

Figure 6. Vertex of the graph generator.

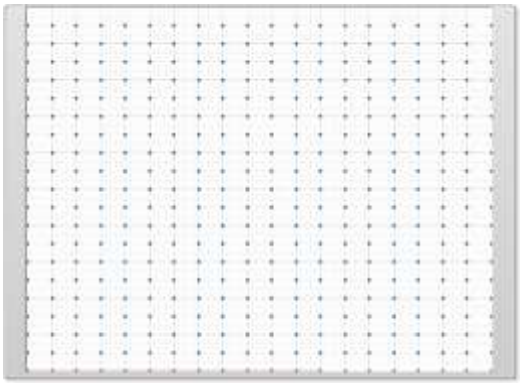

Figure 7. Vertex of the graph.

Vertices are evenly distributed in rows and columns, making it easier to identify and cost between them. A special porocedure was used to name and specify the vertex coordinates figure 6.

The problem is also to connect so many vertices in pairs and set the cost. The connection method with the nearest neighbors has been selected. Figure 8 shows the combination of each vertex with its closest neighbors. Figure 9 shows a combination of each vertex with its closest neighbors and with 16 nearest neighbors.

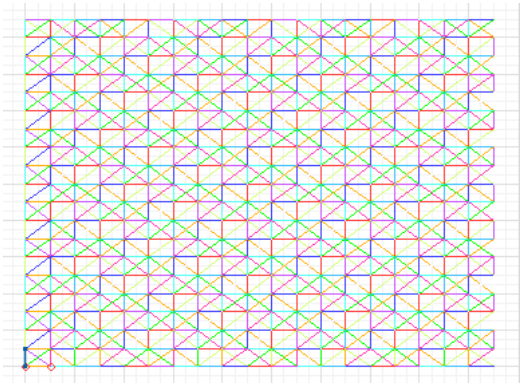

Figure 8. 8 neighbors connetion graph.

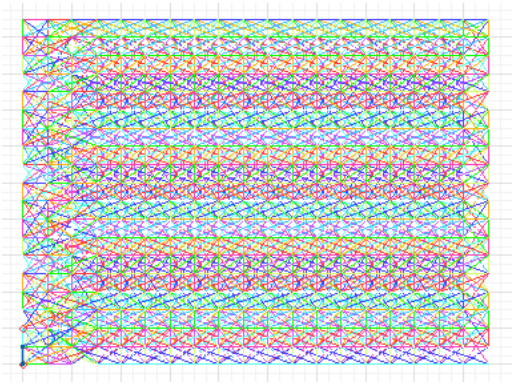

Figure 9. 24 neighbors connetion graph.

Connecting each vertex to 24 neighbors when trying to display the graph caused the computer to slow down, however, the computer has no problems calculating paths. For this reason only vertices and paths are displayed.

Figure 10 shows a section of the program that sets track costs and sets paths.

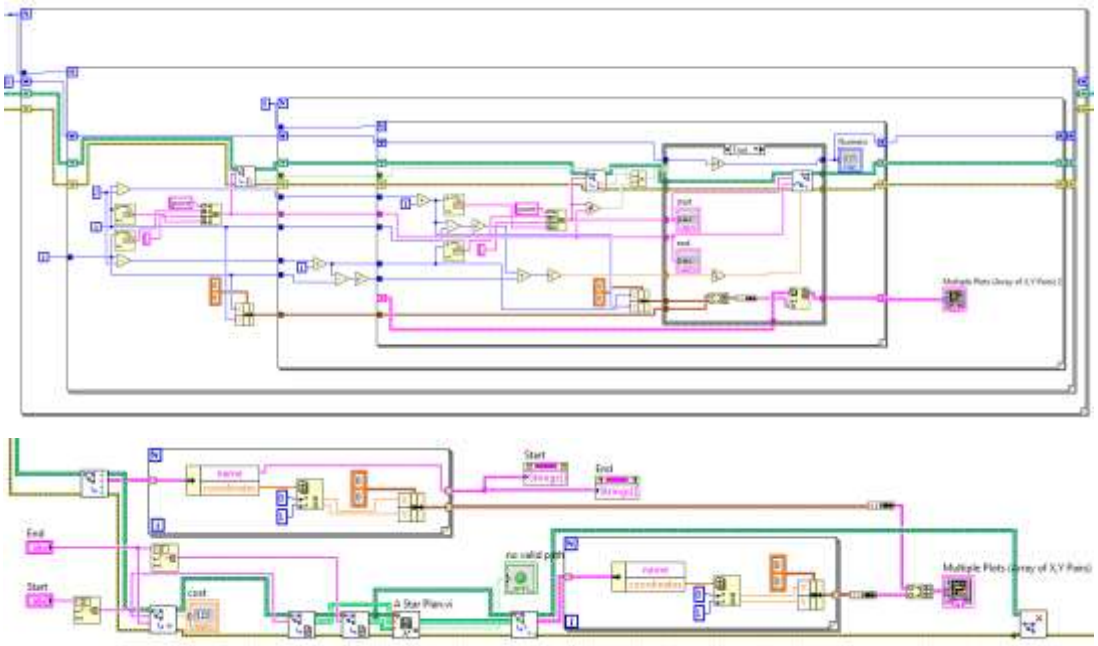

Figure 10. Part of the program responsible for costing paths and handling of the A * algorithm.

Figure 11 shows sample paths to several selected points. It is easy to note that this is a graph with 24 neighbors connected.

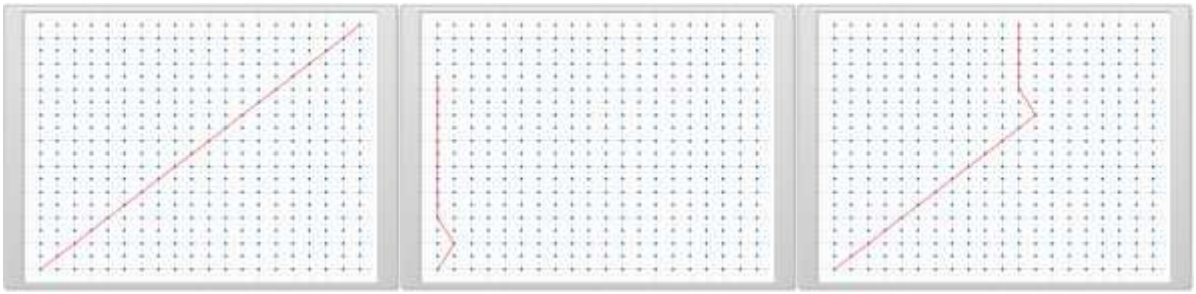

Figure 11. Sample paths generated by A * algorithm.

Not all paths seem to be optimal figure 11, as this may be due to the way the path costs are calculated.

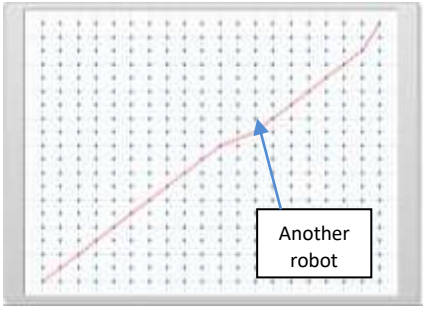

Figure 12. Path with another robot on the course (24 neighbors connected).

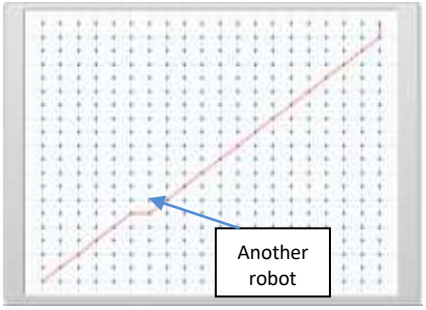

Figure 13. Path with another robot on the course (8 neighbors connected).

Comparing figures 12 and 13 can be seen a faster response of another robot to the path that appears.

4. Conclusions

Both methods are easily programmable in the Labview environment. The results are satisfactory. It is problematic to prepare the layout. There is no possibility of importing from CAD systems, making the interface will be necessary [12,13]. The tool used in the article is only for 2D systems, but the algorithm is universal enough that it can be used in 3D applications. It is very important to determine the cost of crossing between points. For occupied maps, security zones can be assumed. In the case of directed graph, you can determine the cost as a function of the distance of the points and the speed with which the robot moves.

5. References

[1] Banaś W, Sękala A, Gwiazda A, Foit K, Hryniewicz P and Kost G 2015 Determination of the robot location in a workcell of a flexible production line. *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012105
[2] Banaś W, Sękala A, Gwiazda A, Foit K, Hryniewicz P and Kost G 2015 The modular design of robotic workcells in a flexible production line. *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012099
[3] Cholewa A, Świder J, Zbilski A 2016 Numerical model of Fanuc AM100iB robot *IOP Conf. Ser.: Mater. Sci. Eng.* **145** 052002
[4] Cholewa A, Świder J, Zbilski A 2016 Verification of forward kinematics of the numerical and analytical model of Fanuc AM100iB robot *IOP Conf. Ser.: Mater. Sci. Eng.* **145** 052001
[5] Ćwikła G, Krenczyk D, Kampa A and Gołda G 2015 Application of the MIAS methodology in design of the data acquisition system for wastewater treatment plant. *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012153
[6] Dzitkowski T and Dymarek A 2015 Method of active and passive vibration reduction of synthesized bifurcated drive systems of machines to the required values of amplitudes. *Journal of Vibroengineering* **17(4)** 1578-1592
[7] Dymarek A and Dzitkowski T 2016 Inverse task of vibration active reduction of Mechanical Systems *Mathematical Problems in Engineering* ID 3191807
[8] Foit K and Ćwikła G 2017 The CAD drawing as a source of data for robot programming purposes - a review. *MATEC Web of Conferences* **94** UNSP 05002
[9] Herbuś K and Ociepka P 2016 *IOP Conf. Ser.: Mater. Sci. Eng.,* **145** 042018
[10] Herbuś K and Ociepka P 2016 *IOP Conf. Ser.: Mater. Sci. Eng.,* **145** 052010
[11] Herbuś K and Ociepka P 2015 *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012096
[12] Herbuś K and Ociepka P 2015 *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012084
[13] Hryniewicz P, Banaś W, Sękala A, Gwiazda A, Foit K and Kost G 2015 Object positioning in storages of robotized workcells using LabVIEW. Vision *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012098
[14] Hryniewicz P, Banaś W, Sękala A, Gwiazda A, Foit K and Kost G 2015 Technological process supervising using vision systems cooperating with the LabVIEW vision builder. *IOP Conf. Ser.: Mater. Sci. Eng.* **95** 012086
[15] Skołud B, Krenczyk D, Kalinowski K, Ćwikła G and Grabowik C 2016 Integration of manufacturing functions for SME. *Holonic based approach. Advances in Intelligent Systems and Computing* **527** (Springer) 464-473