Tikrit Journal of Engineering Sciences (2023) 30 (2): 142-151
DOI: http://doi.org/10.25130/tjes.30.2.15

IRAQI Academic Scientific Journals

ISSN: 1813-162X (Print); 2312-7589 (Online)
Tikrit Journal of Engineering Sciences
available online at: http://www.tj-es.com

TJES Tikrit Journal of Engineering Sciences

Generalized Procedure for Determining the Collision-Free Trajectory for a Robotic Arm

Yasser Mohammad Al-Sharo [a], Amer Tahseen Abu-Jassar [a] *, Svitlana Sotnik [b], Vyacheslav Lyashenko [c]

a Faculty of Computer Science and Information Technology, Ajloun National University, Ajloun, Jordan.
b Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of RadioElectronics, Kharkiv,Ukraine.
c Department of Media Systems and Technology, Kharkiv National University of RadioElectronics, Kharkiv, Ukraine.

Keywords:
Robot, Movement, Obstacles, Optimal Path, Trajectory.

ARTICLE INFO

Article history:

Received	04 Apr.	2023
Received in revised form	10 June	2023
Accepted	20 June	2023
Final Proofreading	17 July	2023
Available online	30 July	2023

Citation: Al-Sharo YM, Abu-Jassar AT, Sotnik S, Lyashenko V. Generalized Procedure for Determining the Collision-Free Trajectory for a Robotic Arm. *Tikrit Journal of Engineering Sciences* 2023; **30**(2): 142-151.
http://doi.org/10.25130/tjes.30.2.15

*Corresponding author:
Amer Tahseen Abu-Jassar
Faculty of Computer Science and Information Technology, Ajloun National University, Ajloun, Jordan.

Abstract: Robotic systems play an important role in the development and modernization processes of production, facilitation of labor, and human life. The robotic manipulators are outstanding among such systems. Such robots can be used for various spheres of their application. In this case, there is the manipulator's effective control problem in the working area, of which there may be various obstacles. Therefore, a procedure is required to find the optimal path for moving the robotic arm. To develop such a procedure, the literature was reviewed, and the structural diagram of the control system of such a robot and its components was summarized. It proposed a mathematical formalization of the search for the optimal path to move the robot arm, an algorithm based on a modified method of navigation graphs, to realize the more natural movement of the robot arm. Experimental studies were conducted with different numbers of objects on the path of robot arm movement, which were combined into groups. The temporal results of this process are presented in a diagram.

1.INTRODUCTION

Robots have different areas of appl ication, design features, and, as a result, classification and grouping according to functionality [1-5]. All researcher defines robotics as a special area of intellectual modernization of production, construction, and other areas of human life. Robotic manipulators (RM) can separate devices and parts of complex robotic complex to perform functions of a human hand in production and are maneuverable with the possibility of traditional movement in two-dimensional space and 3-dimensional. RMs work in productions, on a level with people, and perform various production functions. While in real-time, they can work autonomously and interact with people without delimitating the working area. It is always necessary to consider not only the working space (environment) of movement, but also obstacles in RM's way, which can affect reaching goal speed. RM devices can automatically detect obstacles in their path and then make decisions to overcome them (circumvention); obstacles can also move. Applications examples of RM are material transfer; arc and spot welding; forging and stamping; spray coating; drilling, milling, riveting, grinding, and polishing operations; assembly of mechanical, electrical, and electronic parts; product quality control; surgical operations; and agro-industrial works [6-9]. Today, besides robotic manipulator development, new designs and methods of their control that largely depend on RM functionality and their operation conditions are relevant to solve the issues of planning RM in space with obstacles. In this case, solving the robotic manipulator planning path problem is fundamental in robotics research because it affects the industrial robots' autonomy improvement in static and dynamic environments. One of the urgent problems associated with robotic manipulator modeling, considering the external environment due to the emergence of new robot designs and methods, is to find their movement's optimal path. Optimal movement, or finding the movement's optimal path, is generally understood as a rule, movement (motion) without collisions with obstacles. The planning movement of RM prehistory is needed for moving an object (RM) to achieve a given goal without collisions affecting, to some extent, production intensification. The ability to overcome obstacles will continue its basic functions; otherwise, an emergency may even affect the execution time of the technological process in which such a robot is involved. The work [10] analyzed problems associated with positioning grippers as a solution to these problems focused on kinematics. This paper presents a design and development of a robotic arm with 5 degrees of freedom for feeding the elderly or people with disabilities. Ensuring the functioning of robots in environments with obstacles has become a prerequisite for the emergence of various algorithms and programs that allow them to monitor and consider environmental limitations quickly. Quite popular are algorithms for efficient tracking of robotic arm different trajectories based on neural networks [11]. In Ref. [11], a robot manipulator can follow a cubic position trajectory. In Ref. [13] improved the path search fundamental method, such as algorithm A* in [12], to adapt to the constraints of the neighborhood. In this paper, it was considered a local path (between the current node and target node, which is planned before the next search in the neighborhood of the current node) directly taken if it is safe and has no collisions. In Ref. [13] proposed a post-processing step to optimize the resulting path by straightening the local path to reduce the number of local paths and path length. It is worth paying attention to algorithms whose search optimization is based on genetics and natural selection principles. In Ref. [14], local and global genetic algorithms were considered. The authors evaluated these methods by analyzing trajectory error, using polynomial trajectory to 3rd degree. Considering methods to find the robot manipulator motion path, it should be summarized that such methods can be divided into two main classes:

- Graph-based methods, e.g., Probabilistic Road Maps [15] and Neural Networks [11].
- Tree-based methods, e.g., Rapidly-Exploring Random Trees [16] and Guided Sampling Tree [17].

Table 1 shows the comparative characteristic of modern methods in the field of trajectory planning given in works [12-15]. Therefore, the main concept of this work is to define a trajectory planning strategy to plan the path before the start of movement without collision with obstacles considering the angular orientation of each RM link. Then an emphasis is made on the fact that movement is not angular, i.e., more natural. Planning the trajectory of movement without collisions with obstacles is the task of automatic generation of movement RM trajectories sequence to a given point along a path. As a result, it is necessary to obtain the most rational trajectory, which is, in fact, the shortest collision-free path, for the passage of which RM will spend less time. Thus, it is important for RM its ability to plan short and safe movements. From Table 1, it can be shown that the considered methods do not consider such complex indicators as links angular orientation and movement naturalness because travel time without collisions increased with the robot links numbers and

rotation angles. At the same time, it is important to conduct empirical studies to justify necessary solutions because each RM system performs its functions based on a particular implementation platform.

Table 1 Pros and Cons of Modern Methods in the Field of Trajectory Planning.

Method	Pros	Cons
A method based on A* modified algorithm for Robot Path Planning [12]	Algorithm A* is one of the best algorithms to find the shortest path. The A* algorithm determines the value of the heuristic function just before the collision phase, demonstrating a good reduction in processing time with higher speed.	If the distance between the source and target is large, implementing algorithm A* from the source and target entails checking too many neighboring nodes (a large number of coordinates) one by one, increasing the distance between the source and target and increasing processing time.
A method based on A* modified algorithm for Robot Path Planning [13].	The first advantage is the local path between the current node and the target node, which is planned before the next search in the vicinity of the current node. Also, the local path will be accepted if it is safe and has no collisions. The second advantage is to use a post-processing step to optimize the resulting path by straightening the local path to reduce the number of local paths and path length.	If the number of sample points is insufficient when constructing a graph, the success rate of the search may be lower.
A method based on genetic algorithms, i.e., one method is based on the local genetic algorithm and the other on the global genetic algorithm [14].	The method based on global genetic algorithms was proved to be more efficient, yielding less deviation with respect to desired trajectory and lower computational cost. The key advantage is that such methods can be applied even to complex problems without special methods.	Methods based on genetic algorithms refer to evolutionary computation methods, and they do not guarantee a global solution detection in polynomial time.
Semi-Labilistic Probabilistic Roadmap: Customizable by Parameters and sampling-based method [15].	This method is a stable and reliable path-planning method for robotic manipulators. Useful when the perspective of the object under study is uncertain, i.e., its end state can be affected by a wide range of factors, such as positioning errors, joint failures, and displacements of obstacles.	Although this approach is considered one of the leading approaches in motion planning, primarily for mechanical systems with many degrees of freedom in an environment with obstacles, it is not about realizing more natural movement.

2.METHODOLOGY

The above papers [10 - 15] found a robot-manipulator movement path; however, none of these works focused on RM movement smoothness along the found path and angular orientation of robot links. Therefore, to begin with, it is proposed to solve the problem of robot naturalness movement by modifying the method of navigation graphs. The initial information before finding the trajectory of movement is the information about the path's initial and final point and obstacles in the working space. Thus, the proposed solution goal is to obtain RM's movement path, with a condition of bypassing not only one obstacle, but also more complex situations with several obstacles.

2.1.Generalized Structural Diagram of RM Control System and Its Components

Scientific interest has been in planning the robot's path, i.e., with obstacles and without collisions. The key importance is given to the RM control system. Therefore, before solving the RM movement planning trajectory problem, a generalized structural diagram of the robot-manipulator control system is considered (Fig. 1). Among components of the RM control system, it should distinguish between [18]:

1. means of adaptive control devices for controlling manipulator under conditions of changing environment and parameters.
2. means of program control, designed for control by end users and solving technological problems. Such means include algorithms for control of electromechanical components of manipulator and algorithms for path interpolation, i.e., linear, circular, and spline.
3. actuators, designed to control links of the manipulator.
4. sensing tools, which allow for solving complex technological tasks requiring detailed information about the workspace state. Ideally, it should strive to use sensors that combine different types of sensing: locating, touching, slipping, and force-motion – in a single design.

A discrete model of the robot's configuration workspace (map), i.e., a set of discrete points, must also be created to move RM. To solve the robot planning trajectory problem, there are many systems of robots autonomous programming-software products that perform a 3D simulation of individual industrial robots or small robotic complexes, conducting trajectory optimization, and generation of control programs for robots, e.g., Tecnomatix ROBCAD, RobotStudio, CimStation, IGRIP, Robotworks, Robomax, Famos robotic, and so forth. [19-22].

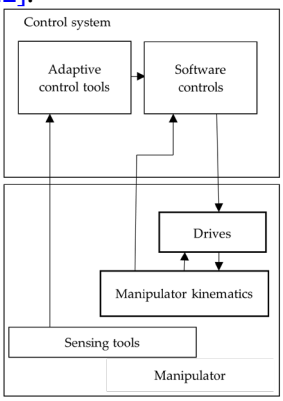
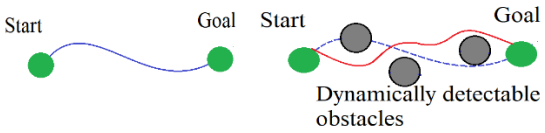


Fig. 1 The Structural Diagram of the RM Control System.

The offline programming tools (OLP) consist of several steps [23-26]:

1. Creating accurate CAD models of all components of the robotic process cell, i.e., parts, process environment, robot, tools, tooling, and so forth.
2. Simulation of robotic technological cell in the virtual 3D environment.
3. Calculation of reference points of robot-manipulator trajectory for its movement to part and execution of technological operations.

So, when considering means of motion planning, it is necessary to understand that environment can be dynamic and static and to know whether the environment consisting of one or many obstacles is fully known without information. The fundamental problem of motion planning is to obtain a collision-free path from the beginning to the goal of the RM. Therefore, it is necessary to consider global and local planning of the trajectory of the RM movement. In global path planning (global path planning, global path, route), i.e., having a starting point and an end point and working space (map) model, it is necessary to find an optimal path (Fig. 2(a)).



(a) **(b)**

Fig. 2 Global and Local Planning Example.

In local path planning (local path, local path, trajectory), i.e., having information coming from sensors, it is necessary to find the part of the path that is free from interference and as short as possible deviating from previously as little as possible deviating from the previously planned global path (Fig. 2 (b)). As a result, the optimization of RM movement trajectory is reduced to find some optimal curve in two-dimensional/three-dimensional space with forbidden areas and boundary conditions.

2.2.Mathematical Formalization of Procedure for Finding Optimal Path to RM Relocate

The trajectory planning problem result is the best or optimal path between two points in space. The criterion for choosing an optimal path depends on specific conditions. For example, it can be the closest or least energy-consuming trajectory; however, the most important task is still the possibility of the path. In this study, the navigation graph method is the most suitable for the optimal path of RM moving, as it is convenient to work with 3-dimensional space [27]. The visibility graph planner is capable of repeatedly applying old calculations when dealing with new targets, i.e., in the case of a visibility graph, when changing the location of a target or robot, it can be called an agent. It is enough only to rearrange the graph edges from the agent and target. The method of basic navigation graphs will be improved to realize more natural movement and application of joined navigation graphs instead of intersecting ones. So, a graph consists of vertices and points in 3-dimensional space and edges that connect these points, applied in Fig. 3.

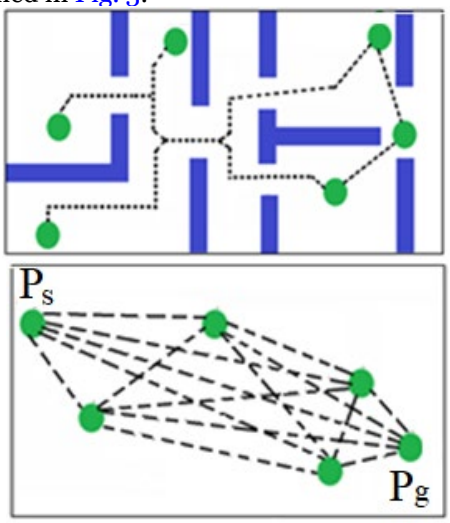


Fig. 3 Example of Building Graph for Trajectory Planning.

Fig. 3 shows examples with possible trajectories of RM movement built considering obstacles, the image of an environment (left) and a graph (right). In the beginning, there is a search for the nearest vertex of a graph as start and end points, and then the usual problem of path search in a graph between found vertices is solved, where the edges weight must be minimalist. In this research, only one object moves the manipulator's grip. It is proposed to present the procedure for finding the optimal path of RM moving:

$$G =< G_1, G_2, \dots, G_i >, \quad \textbf{(1)}$$

where G_1 is the generalized navigation graph of a particular object; $i = 1, \dots, I$, that said I – number of possible graphs. In turn, the graph can be described as set:

$$G_i =< V_i, R_j, P_k >, \quad \textbf{(2)}$$

Where

V_i is a set of graph vertices at $i = 1, \dots, I$, I – number of possible graph vertices; R_j is a set of edges at $j = 1, \dots, J$, J – number of graph possible edges; P_k is the possibility sign (information about the possibility of space around the edge), which consists of a possible set of Pp_{pos} and impossible trajectories of Pp_{ipos} movement.

Highlight conditions for successful trajectory:

1. Absence of collisions.
2. Optimality in movements quantity and quality.
3. Ensuring accurate positioning and following the necessary trajectory.
4. Minimization of time operations.

5. Optimal speed of robot drives.

Once the trajectory has been calculated and verified, it must be translated into the language of the control controller of a specific robot model to be implemented in the production environment. The search for optimal trajectory begins with the definition of starting point P_s and ending point P_g (Fig. 3). Connecting these points will result in a segment, i.e., optimal path, which will be used for calculations. It is necessary to consider the P_i – point, which sets the current position of a moving object on the trajectory, where $i = 1, \dots, I$, I – number of "current" points (at different points in time) of movement on the path to the endpoint. Suppose some path T_i on graph G_i from point P_i to point P_g of length l can be expressed as:

$$\mathrm{T}_i = v_{i_1} v_{i_2} \dots v_{i_{l+1}} \quad \textbf{(3)}$$

where $v_{i_1} = P_i$, then $v_{i_{l+1}} = P_g$.

The motion along each edge R_j in this path has a constraint that stipulates the possibility of physical passage of object between obstacles-passability. Then, let all sets of possible trajectories (paths) of RM movement:

$$\mathrm{T} = \{\mathrm{T}_1, \dots, \mathrm{T}_L\} \quad \textbf{(4)}$$

where T_n is subsets of different components of trajectory, i.e., segments connecting P_i and P_g. In this study, the object-gripper RM is considered dynamic and static objects. The difference is that the dynamic objects can change their state, i.e., position and rotation, and work in 3-dimensional working space. Then, the gripper is considered a dynamic object that can change its position or angle (state) in the course of movement. Then, model G (1.1) is proposed to be represented as:

$$G_d = \{G_i, P_i \neq \mathrm{P}_{i'} \cup U_i \neq U_i'\} \quad \textbf{(5)}$$

where P_i is the object's position at the current time; U_i is the object's rotation at the current moment; $\mathrm{P}_{i'}$ is the object's position at the moment of preliminary path search; and U_i' is the turned object at the moment of preliminary path search. The found objects G_d have changed their state, so their graphs should be listed. Then any path from P_i to P_g will intersect some sequence of navigation graphs G_d. Therefore, a search of navigational graphs is implemented. Further on, merged navigation graphs will be used instead of intersecting ones. Let P_i and P_g be identified with some vertices V_i and V_g, and the corresponding trajectory (path) will correspond to some path in the graph. Then, points of intersection with graph edges are searched, and the resulting list of points is divided into several parts. Each part of the list has its own set of points for a certain navigation graph:

$$P_c = \{P_i, Ds(P_i, P_s) < Ds(P_{i+1}, P_s), i \in 1 \dots n-1\}, \quad \textbf{(6)}$$

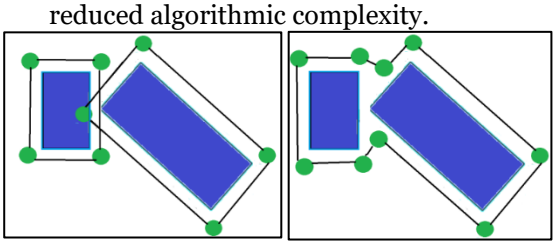
where $Ds(P_i, P_s)$ is the distance from point P_i to point P_s, and n is the number of intersection points. So, in each part of obtained new list of intersection points, a search for trajectory (path) is implemented. The parts themselves are connected by straight segments.

During the search for a path in the navigation graph, there can be two cases:

1. If there is only one intersection point (P_c) path only touches the navigation graph, then no additional path search is required.
2. If there are two intersection points, a path search between two intersection points of the navigation graph is performed by the criterion of minimum total length of edges. In this case, property $\mathrm{pp}_{\mathrm{pos}}$ of edge passability is considered.

As a result, the resulting paths are merged into one final path. Each found graph path is connected in turn, i.e., connecting extreme points with a straight line. At the time of path search, dynamic objects do not differ from static ones. However, there is one difference in calculating navigation graphs of dynamic objects (expression 5). Since, in this study, it is considered that capturing RM changes its state (position/rotation) due to what collisions may occur at the moment of collision with dynamic objects, then:

1. Restart the search procedure and consider the new state of all dynamic objects; however, at the same time, new states of dynamic objects may affect the navigation graph of static objects, and some edges may become impassable $\mathrm{Pp}_{\mathrm{Ipos}}$. So, such edges temporarily do not participate in pathfinding until the dynamic object moves.
2. The situation when navigation graphs of objects intersect, i.e., the intersection of one edge dynamic object by another, may lead to the wrong pathfinding solution. Then, the solution is a union of two graphs of these objects so that all edges are passable (Fig. 4).
3. Combining results in an adequate path with reduced algorithmic complexity.



(a) **(b)**

Fig. 4 Examples of Dynamic Objects Navigation Graphs (**a**) Before Merging and (**b**) After Merging.

Nevertheless, this study places the greatest emphasis on empirical research because it is not always possible to formalize cases where the

perspective of the object under study is uncertain.

2.3. Generalized Robotic Model

The developed simplified kinematic diagram of RM is shown in Fig. 5. A simplified kinematic scheme is proposed because such representation displays key information for this work, i.e., the rotational motion of each link. It is quite native these since it contains fewer structural elements.

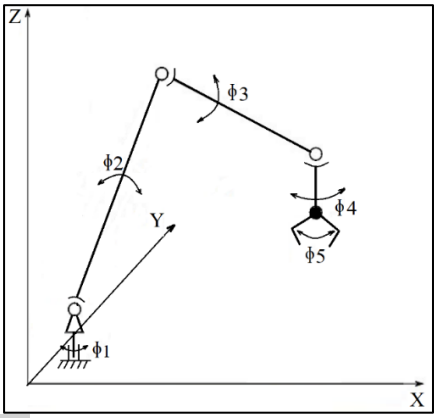


Fig. 5 Simplified Kinematic Diagram of RM.

The robot consists of the following parts: base, body, shoulder, elbow and forearm, and gripper (claw). The body moves by angle ϕ1; the shoulder moves by angle ϕ2; the forearm moves by angle ϕ3; the wrist moves by angle ϕ4; claws move by angle ϕ5. It is important to note that the time for planning a rational trajectory of movement RM depends on the presence of obstacles, the number of coordinates, and the angular orientation of the robot links.

3. RESULTS AND DISCUSSION

3.1.Project Proposals

To begin, the project of finding optimal paths in nine steps is described.

Step 1:Data input – loading workspace map including points Ps and Pg between which is needed to find the optimal path.

Step 2:Identification of objects on the path. Here images of obstacles on a map will be represented as "bypass frames" of objects, i.e., they are rectangles of arbitrary size to simplify their association.

Step 3: Determination of "bypass frame" points. Each rectangle focuses on two points (P_{max} and P_{min}), maximum and minimum points with respect to the origin. Therefore, after combining objects, if objects are encountered while moving along the trajectory, it is sufficient to find new maximum and minimum points with respect to the origin of coordinates.

Step 4: Grouping of objects (merging) is needed to make it easier to bypass objects (example in Fig. 4).

Step 5: Check the entire path for obstacles. If there are any, go to Step 2 to determine the new P_{max} and P_{min}. If not, go to Step 6.

Step 6: Construction of a navigation graph (described in detail in 3).

Step 7: Search for intersection points with obstacles (grouped) on the path between points Ps and Pg. Sort intersection points Pc by distance from Ps, i.e., path's starting point.

Step 8: Determination of optimal path for avoiding obstacles. The obstacle avoidance using graphs is computed through the minimum sum of all edges. Since, in this study, obstacles will always be in the form of rectangles, there will be two types of traversal options: if intersection points are on adjacent sides, calculating two additional points to traverse is required; if intersection points are on opposite sides, calculating one additional point to traverse is required.

Step 9: Determine the entire path of travel. As a result, resulting path segments with and without obstacles are combined into one final path.

The algorithm for solving the problem of finding the optimal path in 3-dimensional space is shown in Fig. 6.

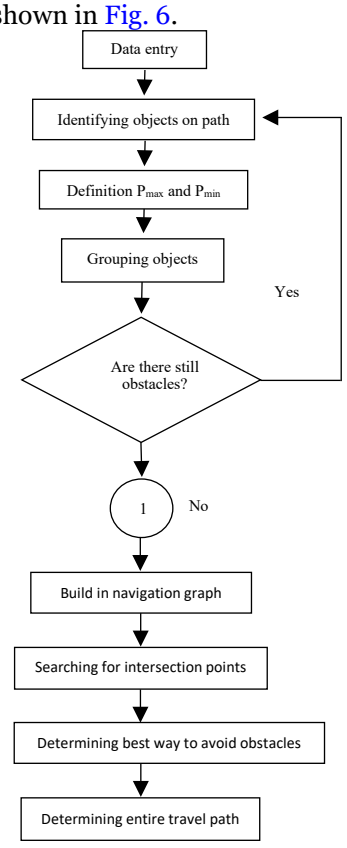


Fig. 6 Algorithm for Solving the Problem of Finding the Optimal Path.

Subsequently, the proposed algorithm was implemented in the Visual Studio environment.

3.2.Software Implementation and Experimental Research

The software was developed in a Visual Studio environment. C# was used as the language. The chosen platform is .NET Core, free and open source; it provides cross-platform. Open Toolkit library was also used since it provides several service libraries and allows using the latest versions of OpenGL, a graphics standard in the computer graphics field. A generalized block diagram of the program is shown in Fig. 7.

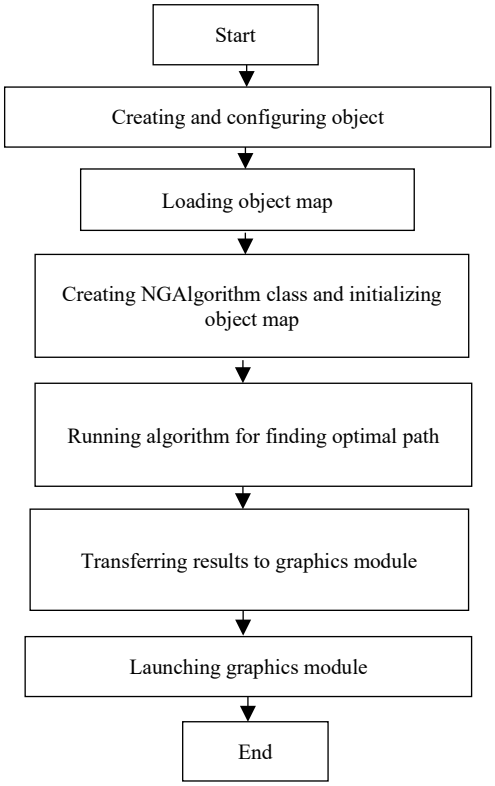


Fig. 7 Generalized Algorithm of Program.

First, create an object to playback space DMSWindow, characterized by: window size, name of window, frame refresh rate, and refresh rate of states. Next, load a map of all objects represented as an array of coordinates in the static class Maps.cs. Then, an instance of NGAlgorithm class was created in which a map of objects is passed. NGAlgorithm represents an algorithm for the graph method. To start the algorithm of finding the optimal path: call GetRoute methods that need to pass the initial and final coordinates of the path. The output is an array of points representing the optimal path for an object in a given space. Transfer results, i.e., found path and object map, to graphical module Drawing, an instance of DMSWindow class. Launch the graphical module Drawing and draw objects. Each time a state update is launched, the object is moved (Figs. 8 - 10 (a, b, and c)).

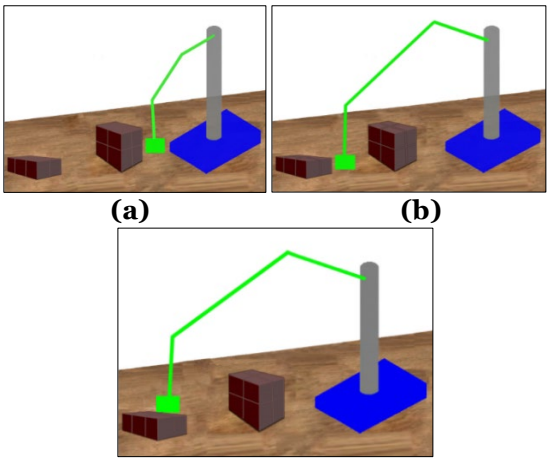


Fig. 8 Program Window (Experiment 1).

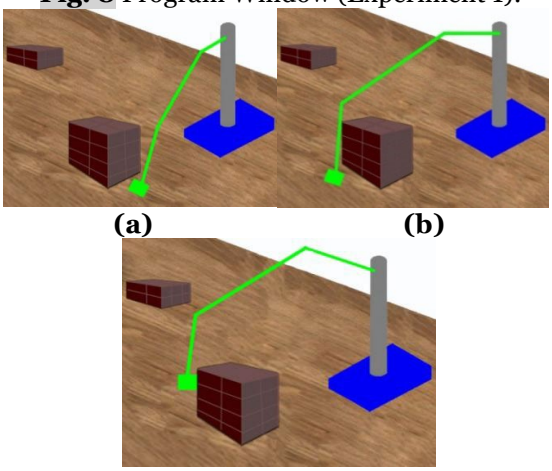


Fig. 9 Program Window (Experiment 2).

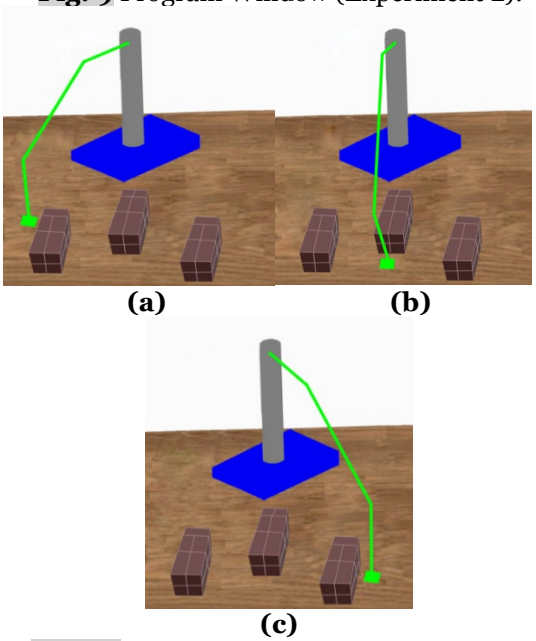


Fig. 10 Program Window (Experiment 3).

For the experiment, a different number of obstacles were selected, which were combined into groups of initial data in Table 2. The route timing is shown in Fig. 11, and there were three test runs in each experiment.

Table 2 Experimental Data.

Experiment No.	Number of objects on the workspace map	Number of obstacles groups	Constructed route (number of coordinates)
Experiment 1 (Fig. 8)	14	2	8
Experiment 2 (Fig. 9)	28	2	6
Experiment 3 (Fig. 10)	28	3	12

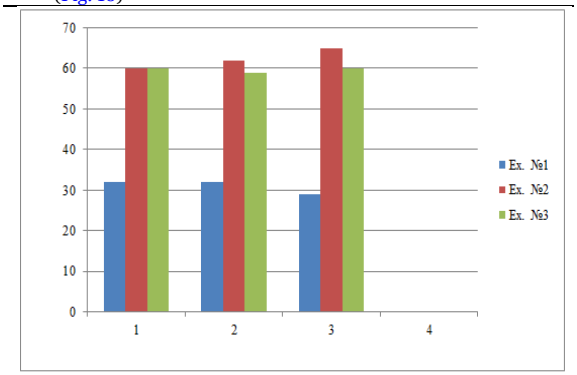


Fig. 11 Time to Build Path in Milliseconds.

Thus, it took tens of milliseconds to find a path that allowed using an intelligent decision-making system in real-time and recalculating the path for each collision with dynamic objects in space. The trajectory of robot movement is determined not only by the number and shape of obstacles, which are then grouped, but also by the number of coordinates, with their increase, and the range of angular orientation of robot links (rotation angles). Then, studies of travel time dependence at different angles of link rotation were conducted. These studies are shown in Table 3.

Table 3 Experimental Data.

Experiment No.	Number of coordinate	Time travel time, seconds	Range of link angles				
			ϕ1	ϕ2	ϕ3	ϕ4	ϕ5
Experiment 1 (Fig. 8)	8	47	30°-60°	10°-40°	20°-45°	45°-90°	0°-45°
Experiment 2 (Fig. 9)	6	30	0°-30°	10°-40°	10°-30°	0°-30°	0°-45°
Experiment 3 (Fig. 10)	12	55	30°-60°	10°-100°	20°-80°	20°-90°	0°-45°

After analyzing and evaluating obtained data, it can be said that the rotation angle of each robot link directly affected the travel time in different conditions.

4.CONCLUSIONS

This paper deals with procedure development to move a robot manipulator. A review of literary sources and relevant tools has been carried out, outlining the main ways of solving the problem. As a result, the mathematical formalization of the procedure for finding an optimal path to move the robot arm has been proposed. An algorithm for such path search based on a modified method of navigation graphs was implemented to realize more natural movement. Based on such an algorithm, different experiments were conducted, which evaluated the proposed solutions' feasibility to understand the applicability of such an approach in real conditions. In three experiments, different numbers of obstacles were selected, i.e., 14 and 28, which subsequently were combined into groups. In the course of determining the movement rational (without collision with an obstacle) trajectory, the proposed software built routes with different numbers of coordinates. It was found that more coordinates and smoother movement, but more time were required to build the route, although time variation was insignificant. Experiments were conducted to evaluate the effectiveness of developed software for planning the trajectory of the robotic arm in conditions with different numbers of obstacles in the robot's path, which proves that dividing obstacles into groups allowed for simplifying the trajectory planning task and finding a rational route for the robot that minimized movement time. In addition, the proposed software is flexible because with equal initial data (28 objects in working space), conditions and still finding the shortest path can be adapted. The robot links coordinate number and angular orientation dependence was also investigated. It was shown that with a different grouping of obstacles and their placement, fewer coordinates, and smaller angles of link rotation, faster robot travels in a given path were achieved. These results can be useful for optimizing the robot manipulator's performance in various conditions where the movement speed is a critical factor. The main condition for applying the proposed software for planning the trajectory of the manipulator robot depends on its specific implementation and the capabilities of the robot for which it is designed. The software can be used to plan manipulator robot movement trajectories in various applications, including manufacturing, medical, construction, and more. Limitations may depend on the complexity of the trajectory planning task, the number and shape of obstacles and their density, the robot characteristics, and so forth. A software model was developed in a Visual Studio environment. The language used was C# and .NET Core platform. During testing, it was concluded that the proposed solution is effective and fast and works in real-time. This solution can be used in intelligent decision-making systems for robotic systems. The next stage of such work is to create a specialized control system for RM.

REFERENCES

[1] Matarneh R, Maksymova S, Deineko Z, Lyashenko V. **Building Robot Voice Control Training Methodology using Artificial Neural Net**. *International Journal of Civil Engineering and Technology* 2017; **8**(10): 523-532.

[2] Attar H, Abu-Jassar, AT, Yevsieiev V, Nevliudov I, Lyashenko V, Luhach AK.

Zoomorphic Mobile Robot Development for Vertical Movement Based on the Geometrical Family Caterpillar. *Computational Intelligence and Neuroscience* 2022; **2022**: 3046116, (1-19).

[3] Baker JH, Laariedh F, Ahmad MA, Lyashenko V, Sotnik S, Mustafa SK. **Some Interesting Features of Semantic Model in Robotic Science**. *SSRG International Journal of Engineering Trends and Technology* 2021; **69**(7): 38-44.

[4] Zaki ND. **An Interactive Human Interface Arm Robot with the Development of Food Aid**. *Tikrit Journal of Engineering Sciences* 2012; **19**(1): 54–61.

[5] Salih TA, Nayef MZ. **New Design of Mobile Robot Path Planning with Randomly Moving Obstacles**. *Tikrit Journal of Engineering Sciences* 2013; **20**(1): 21–28.

[6] Spong MW. Seth Hutchinson, and Mathukumalli Vidyasagar. Robot Modeling and Control; Wiley: New York, NY, USA, 3, 2006.

[7] Deniz C, Cakir M. **In-Line Stereo-Camera Assisted Robotic Spot Welding Quality Control System**. *Industrial Robot: An International Journal* 2018; **45**(1): 54-63.

[8] Diodato A, Brancadoro M, De Rossi G, Abidi H, Dall'Alba D, Muradore R, Cianchetti M. **Soft Robotic Manipulator for Improving Dexterity in Minimally Invasive Surgery**. *Surgical Innovation* 2018; **25**(1): 69-76.

[9] Ghadge K, More S, Gaikwad P, Chillal S. **Robotic Arm for Pick and Place Application**. *International Journal of Mechanical Engineering and Technology* 2018; **9**(1): 125-133.

[10] Kruthika K, Kumar BK, Lakshminarayanan S. Design and Development of a Robotic Arm. *In 2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, 2016: 1-4.

[11] Elsisi M, Mahmoud K, Lehtonen M, Darwish MM. **An Improved Neural Network Algorithm to Efficiently Track Various Trajectories of Robot Manipulator Arms**. *IEEE Access* 2021; **9**: 11911-11920.

[12] Guruji AK, Agarwal H, Parsediya DK. **Time-Efficient A* Algorithm for Robot Path Planning**. *Procedia Technology* 2016; **23**: 144-149.

[13] Fu B, Chen L, Zhou Y, Zheng D, Wei Z, Dai J, Pan H. **An Improved A* Algorithm for the Industrial Robot Path Planning with High Success Rate and Short Length**. *Robotics and Autonomous Systems* 2018; **106**: 26-37.

[14] Reboucas Filho PP, da Silva SPP, Praxedes VN, Hemanth J, de Albuquerque VHC. **Control of Singularity Trajectory Tracking for Robotic Manipulator by Genetic Algorithms**. *Journal of Computational Science* 2019; **30**: 55-64.

[15] Akbaripour H, Masehian E. **Semi-Lazy Probabilistic Roadmap: A Parameter-Tuned, Resilient and Robust Path Planning Method for Manipulator Robots**. *The International Journal of Advanced Manufacturing Technology* 2017; **89**: 1401-1430.

[16] Li B, Chen B. **An Adaptive Rapidly-Exploring Random Tree**. *IEEE/CAA Journal of Automatica Sinica* 2021; **9**(2): 283-294.

[17] Cheng X, Huang E, Hou Y, Mason, MT. Contact Mode Guided Sampling-Based Planning for Quasistatic Dexterous Manipulation In 2d. *In 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021: 6520-6526.

[18] Nise NS. **Control Systems Engineering**. John Wiley & Sons, 2019.

[19] Vitolo EF. Multi-Attribute Task Sequencing Optimisation with Neighbourhoods for Robotic Systems, Doctoral dissertation, University of Naples Federico II, 2017.

[20] Kiwała S, Kazibudzki J, Muraszkowski A, Olinski M. **Fast Trajectory Planning and Programming of a Robotic Manipulator using Robotstudio Software for Inertial Sensor Testing**. *Interdisciplinary Journal of Engineering Sciences* 2016; **4**(1): 56-64.

[21] Wohlrab R, Cámara J, Garlan D, Schmerl B. **Explaining Quality Attribute Tradeoffs in Automated Planning for Self-Adaptive Systems**. *Journal of Systems and Software* 2023; **198**: 111538, (1-23).

[22] Thomas C, Stankiewicz L, Grötsch A, Wischniewski S, Deuse J, Kuhlenkötter B. **Intuitive Work Assistance by Reciprocal Human-Robot Interaction in the Subject Area of Direct Human-Robot Collaboration**. *Procedia Cirp* 2016; **44**: 275-280.

[23] Mineo C, Pierce SG, Nicholson PI, Cooper I. **Robotic Path Planning for Non-Destructive Testing–A Custom MATLAB Toolbox Approach**. *Robotics and Computer-Integrated Manufacturing* 2016; **37**: 1-12.

[24] Neto P. Off-Line Programming and Simulation from CAD Drawings: Robot-Assisted Sheet Metal Bending. *In IECON 2013-39th Annual Conference of the IEEE*

Industrial Electronics Society, 2013: 4235-4240.

[25] Huynh D. **Robotic Arm Modelling and Framework for Offline Programming**, 2019. https://www.theseus.fi/bitstream/handle/10024/266725/Huynh_Dat.pdf?sequence=4

[26] Savanevych VE, Khlamov SV, Akhmetov VS, Briukhovetskyi AB, Vlasenko VP, Dikov EN, Trunova TO. **CoLiTecVS Software for the Automated Reduction of Photometric Observations in CCD-Frames**. *Astronomy and Computing* 2022; **40**: 100605, (1-15).

[27] Irshad MZ, Ma CY, Kira Z. Hierarchical Cross-Modal Agent for Robotics Vision-And-Language Navigation. *In 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021: 13238-13246.