mathematics

Article

Path Planning of Logistic Robot Using Method of Vector Marks Tree Generation

Olga Strikuliene [1], Kastytis Kiprijonas Sarkauskas [2], Julius Gelsvartas [2], Leonas Balasevicius [2], Virginijus Baranauskas [2],* and Alma Derviniene [3]

1 Technology and Entrepreneurship Competence Center, Panevezys Faculty of Technologies and Business, Kaunas University of Technology, Nemuno St. 33-218, LT-37164 Panevezys, Lithuania; olga.strikuliene@ktu.lt
2 Department of Automation, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, Studentu St. 48-320, LT-51367 Kaunas, Lithuania; sarkauskaskastytis@gmail.com (K.K.S.); julius.gelsvartas@gmail.com (J.G.); leonas.balasevicius@ktu.lt (L.B.)
3 Department of Electronics Engineering, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, Studentu St. 48-213, LT-51367 Kaunas, Lithuania; alma.derviniene@ktu.lt
* Correspondence: virginijus.baranauskas@ktu.lt; Tel.: +370-37-300-291

Abstract: The authors of this article analyzed the problem of logistic robotics. This paper presents a method for robot navigation in a known environment. The method consists of two steps. The first step is to model the system, assign vector marks to the prominent edges of the virtual environment map, and direct the robot to reach these marks. The second step is to enable the robot to execute a specific task based on the given paths and deal with the local obstacles avoidance independently. The identification of the prominent point, the computation of the vector mark, and optimal path calculation are performed on the computer model using colored Petri nets in the software 'Centaurus CPN'. The proposed approach was extended to simulate the work of a logistic robot, which has to take boxes and deliver them to certain places in storages. The experimental investigation has shown that the simulated mobile robots with the proposed navigation system were efficiently moving along the planned path. The analysis of the vector tree reveals that it takes 0.389 s to compute and graphically represent it. The occupation of certain places in storages is visualized and shown in experimental graphics.

Keywords: robot control; path planning; vector mark; logistic robot; vector tree

MSC: 93-08

check for updates

Citation: Strikuliene, O.; Sarkauskas, K.K.; Gelsvartas, J.; Balasevicius, L.; Baranauskas, V.; Derviniene, A. Path Planning of Logistic Robot Using Method of Vector Marks Tree Generation. *Mathematics* **2024**, *12*, 73. https://doi.org/10.3390/math12010073

Academic Editors: Idilia Batchkova and Elena Koleva

Received: 2 December 2023
Revised: 19 December 2023
Accepted: 22 December 2023
Published: 25 December 2023

1. Introduction

Path planning is an active research topic relevant to many robotic applications. A broad range of algorithms has been developed to solve this problem. Therefore, comparing different algorithms is challenging. A comprehensive comparison of different path planning algorithms is presented in [1]. Another recent study categorizes path planning algorithms into meta-heuristic and conventional categories [2]. The Kiva systems robotic logistics system is used in Amazon's logistics centers. The task of warehouse robots is to lift the shelves of products and deliver them to the correct warehouse locations. The robots' working area is separated from humans [3–5]. The Fetch company develops robots to transport goods from one place to another. These robots can work alongside humans in the same space and navigate in environments with dynamic moving obstacles [6].

The most important task of any path planning algorithm is to make sure that the robot will not hit any obstacles when executing the planned path. Therefore, path planning is often divided into two separate algorithms, namely local and global path planning. The main responsibility of the global path planner is to find the path from the start location to the goal while avoiding all known obstacles. The local planner calculates the robot velocity commands that help the robot stay on the planned global path. Additionally, the local planner uses the robot sensor information to avoid unexpected obstacles. When the unexpected obstacle is avoided, the robot can return to the planned global path. The robot operating system (ROS) has one of the most popular path planning architecture implementations. This system is used in many robotic logistic tasks [7]. More recently, this system has been improved and re-implemented in ROS2 [8]. This version places a lot of emphasis on the definitions of behavioral navigation behavioral tree definitions. These behavioral trees improve the performance of the overall system in challenging and dynamic environments.

In this paper, a trajectory planning algorithm based on the generated unique vector marks tree method is applied to the modeling of a control system for a logistic mobile robot. The proposed method aims to find the shortest trajectory to the target point, considering the dimensions of the mobile robot with the load. The logistics distribution system for transported products assesses the occupancy of the racks and shelves and directs the loads accordingly.

The subsequent content of this paper consists of: path planning algorithms are revised in Section 2; Section 3 describes the vector marks tree-generated method and defines the object of research; Section 4 presents the results of the research. The future work and conclusions are summarized in Section 5.

2. Path Planning Algorithms

Path planning algorithms can be divided into four main categories, namely those based on graph, potential field, optimization, and random search. A detailed review of the path planning algorithm can be found in [9]. The choice of algorithm is highly dependent on the planning stage of the path. Graph-based algorithms are mainly sued for global path planning. Local path planning is usually implemented using potential fields of search-based algorithms.

Environment maps are needed to plan the global path planning. These maps can be created from building plans or by a robot driving in that environment. Simultaneous localization and mapping (SLAM) are a group of algorithms that can be used to create environment maps in real-time [10]. One of the most popular SLAM frameworks in recent years has been described in [11]. For example, see Figure 1.

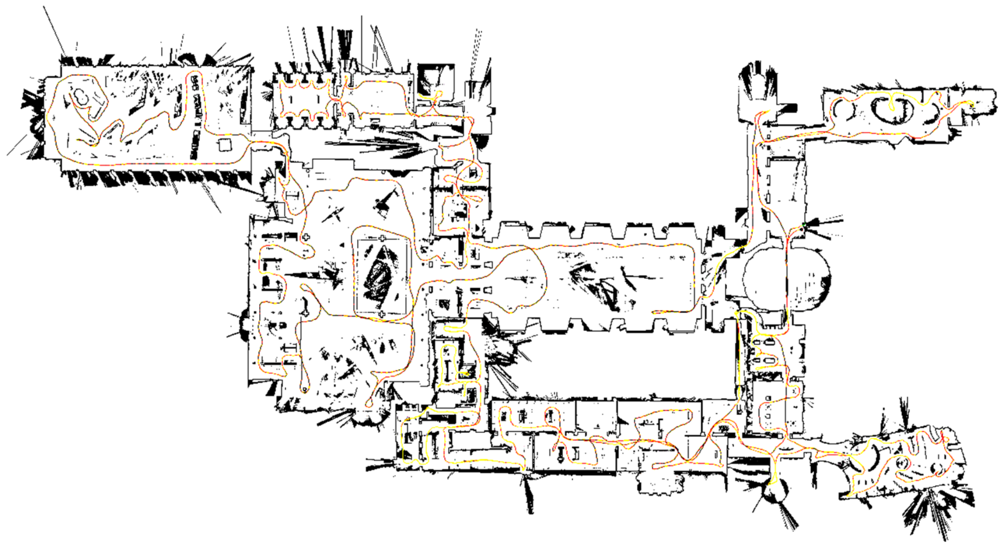

Figure 1. Example of a map created using Cartographer SLAM [11].

Environment maps are usually discretized before using graph-based path-planning algorithms. The discretization step has to be selected according to the dimensions of the robot. In this case, obstacles are marked as occupied cells on the discrete environment map.

The robot is treated as a point object when performing global path planning. Because the robot is not a point object, it can hit the obstacles when moving through the path planned for the point object. To avoid this problem, obstacles are often inflated on the global map. The inflation distance is selected according to the robot's diameter. When planning the path in the inflated environment map, the robot can always move from any free cell to all of its neighboring free cells. Obstacle inflation guarantees that the robot will never hit an obstacle when moving along a planned path [12]. The relative standard deviation of each path's distance (*RD*) mathematical computation formula follows [12]:

$$RD = \frac{\sqrt{\frac{1}{N \cdot K}\sum_{i=1}^{N \cdot K}(D_i - D_m)^2}}{D_m} \tag{1}$$

where D_i is the ith path's distance in cells, D_m is the mean distance D of each path, K is experiment's ID, and N is the number of successive goals that the algorithm must produce paths for. This metric checks the punctuality of the method concerning its produced paths.

Global path planning can be performed using the classical graph shortest path planning algorithms. However, the selected map cell size must not be too small. *A** and *Dijkstra* are some of the algorithms that are often used in global path planning (see Figure 2). The *Dijkstra* algorithm is guaranteed to return an optimal path, but it often has to perform a lot more computations. *A**, on the other hand, uses heuristics to reduce the computation requirements of the algorithm. Both algorithms can find the optimal path when the conditions are right. The biggest disadvantage of both *A** and *Dijkstra* is that planning has to be performed from scratch when an unknown obstacle is detected. This can lead to high computational loads when the robot operates in highly dynamic environments. *D* Lite* is a more efficient version of *A** that stores the intermittent planning results and can reuse this information when a dynamic obstacle is detected. *D* Lite* has been successfully used for global path planning [13].

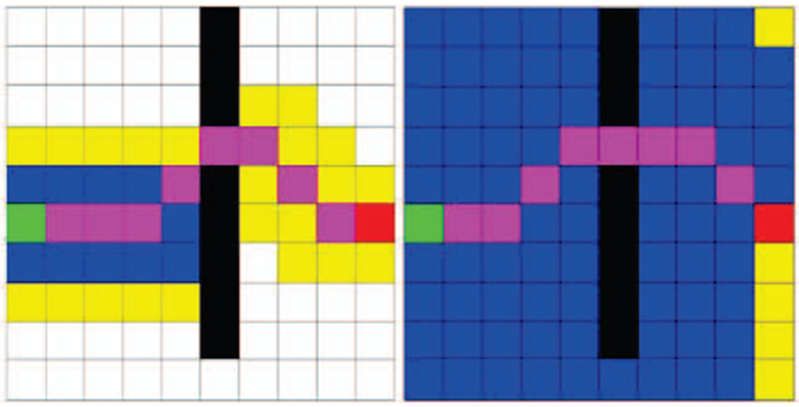
Figure 2. *Dijkstra* and *A** path-planning differences [14].

Local path planning converts the global path into velocity commands. These commands are used to calculate the robot actuator velocities. When doing global path planning, the robot only has to select between four and eight possible actions in each cell. Local path planning, on the other hand, works in a continuous space. This is because, in each time step, the robot could drive at any desired speed. Local path planning is often performed in a small window near the robot. This reduces the computational load when dealing with a continuous action space. The size of the local path planning window is usually chosen by taking the reaction time into account. This is performed so that the robot could avoid dynamic obstacles that can appear in the robot environment. Additionally, planning the local path always takes the robot's footprint into account.

As already mentioned, robots can have a continuous velocity action space. It is impractical to try to evaluate all possible robot velocities when doing local planning. One way to address this problem is to select a set of possible robot velocities and then check with one who obtains the robot closest to its goal. It is also important to check that the selected action will not cause the robot to hit any obstacles. One of the most popular algorithms that uses this method is the dynamic window approach [15] (see Figure 3).

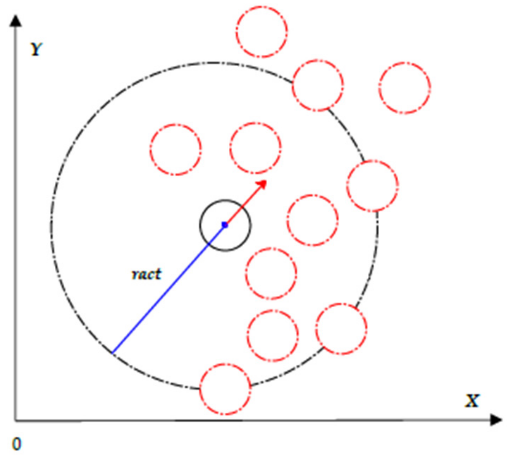

Figure 3. Dynamic window algorithm for local path planning [16].

Optimization-based algorithms search for the optimal robot trajectory. It is very important to select a good optimization criterion for these algorithms. The goal function can have one or more optimization criteria. For example, the criteria can be: trajectory length, smoothness, distance to obstacles, execution time, or energy consumption. The choice of algorithm parameters will determine how the optimized trajectory looks. Optimization-based algorithms can generate very good trajectories because they operate in a continuous space. The main disadvantage of optimization-based algorithms is that they usually have many parameters that need to be tuned to obtain reasonable algorithm performance. Another problem is the possibility that the optimization algorithm becomes stuck in a local minimum. Despite these problems, optimization-based algorithms are successfully used for robot navigation [17]. Moreover, optimization-based algorithms can be applied when controlling Ackerman drive robots [18].

Path-planning algorithms are similar to optimization-based algorithms. These algorithms create a monotonic function that decreases the movement towards the target location. Path planning is then performed by performing a decent gradient on this function. As with optimization-based algorithms, local minima are often a source of concern for these algorithms. Many different methods are proposed to help avoid these issues [19]. A detailed review of different local path planning algorithms can also be found in [20]. In recent years, learning-based approaches have also become popular [21]. These methods can achieve even higher performances when combined with modern deep neural network approaches [22]. Neural network training is often based on error minimization methods. The error that the network is trained to minimize can be chosen according to the problem to be solved and the type of neural network. One error function widely used for training neural networks is the sum squared error. The sum of squared errors is suitable for solving regression problems. The cumulative squared error is obtained by summing the errors in all the outputs of the network, for the entire data sample:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{y^k(x^n, w) - t_k^n\right\}^2, \tag{2}$$

where N is the number of input vectors, c represents the number of outputs of the network, k is kth output of the network, calculated as a function of the nth input vector x^n and weight vector w.

Path planning can also be performed for several robots simultaneously. In this case, the goal is to plan a joint plan to make sure that robots will not be obstacles to each other. These plans can also help avoid congestions. These plans can consider the robot waiting time. Simultaneous path planning is often used in logistics to find a common plan for all robots operating in a warehouse [23]. The researchers [24] also proposed a path planning method based on neural network training, which allows one to achieve a level of accuracy close to 90%. Such planning approaches can produce very good results, but their practical implementation is often very difficult. Simultaneously planning a path for a robotic fleet is even more computationally expansive than the other methods discussed so far. Another big problem is the re-planning that is needed when unexpected situations arise.

3. Vector Mark Generation Method and Research Object

This paper presents a method for finding the shortest path for mobile robots using vector marks. Vector marks are derived from the scanned outlines of the environment around the robots. The supervision system controls the robots' movement by tracking each one separately and calculating the optimal route. The vector marks help estimate the movement trajectory of the mobile robot. In order to describe the properties of vector marks, the authors describe "visibility"—a point p_d is visible from point p_0, if there exists a line segment connecting these points and all points on this line belong S_{free}:

$$\forall p \in l(p_0, p_d) \in S_{free}, \tag{3}$$

where $l(p_0, p_d)$—means the line connecting points p_0 and p_d. Before analyzing the scanned data, we need to check the contour of scanned obstacles for any changes, gaps, or breaks. These are called salient points and they indicate important abnormalities. We can find them by looking for places where the gradient direction suddenly changes. To do this, we only need to measure the tangential angles at the points we are interested in, instead of computing the exact gradient at every point. The research in this article is focused on the detection of salient points, where the shape of obstacles changes [25]. Figure 4 shows the algorithm of vector mark generation.

The length of the vector mark is an important factor. This length influences the weight coefficient of the mark, which is calculated from the end of the mark to the target point. The length of the mark is also included in the calculation of the weight coefficient of a vector mark. Lengthening the mark may lead to a situation where the mark falls into a static obstruction zone. That is, the distance between two static obstacles is less than the specified length of the vector mark. In order to avoid this situation, the algorithm that recalculates the length of the vector mark is adjusted. Using the heuristic relationship between the size of the vector mark l and the length of the jump Δl at the point of rupture found in the scanner contour, the proposed algorithm is used:

$$l = \begin{cases} 2 * g, \; if \; \Delta l \geq 4 * g \\ \quad 0.25 * \Delta l \end{cases}, \tag{4}$$

where g is the scanner distance and Δl is length of a radar beam jump for aesthesia.

The shortest path between two points is known to be a straight line. However, there are not always no obstacles between the goal and the current position points.

In this case, the optimal path is a polygonal path approximated by the shortest distance between the robot's current position and the target point. If the target point is obstructed by an obstacle, there is at least one point at the vertex of the polygonal path that allows the robot to bypass this obstacle. Obviously, there may be cases where one new point will not be sufficient to overcome the obstacle. The path through these vertices will, in the general case, be far from the shortest path. A more accurate approximation is therefore needed.

Authors use criteria λ to select the vector, which can lead to the target point in the shortest path, from visible vectors k:

$$\lambda = \min_{n=1}^{k}(W_n + \|p_{nv} - p_r\| + \|p_{nh} - p_{nv}\|)n \in \Re, \tag{5}$$

where W_n is the weight coefficient of the n-th vector mark; p_{nv} is the visible vector mark point; p_r is the robot position; p_{nh} is the apex of the vector mark.

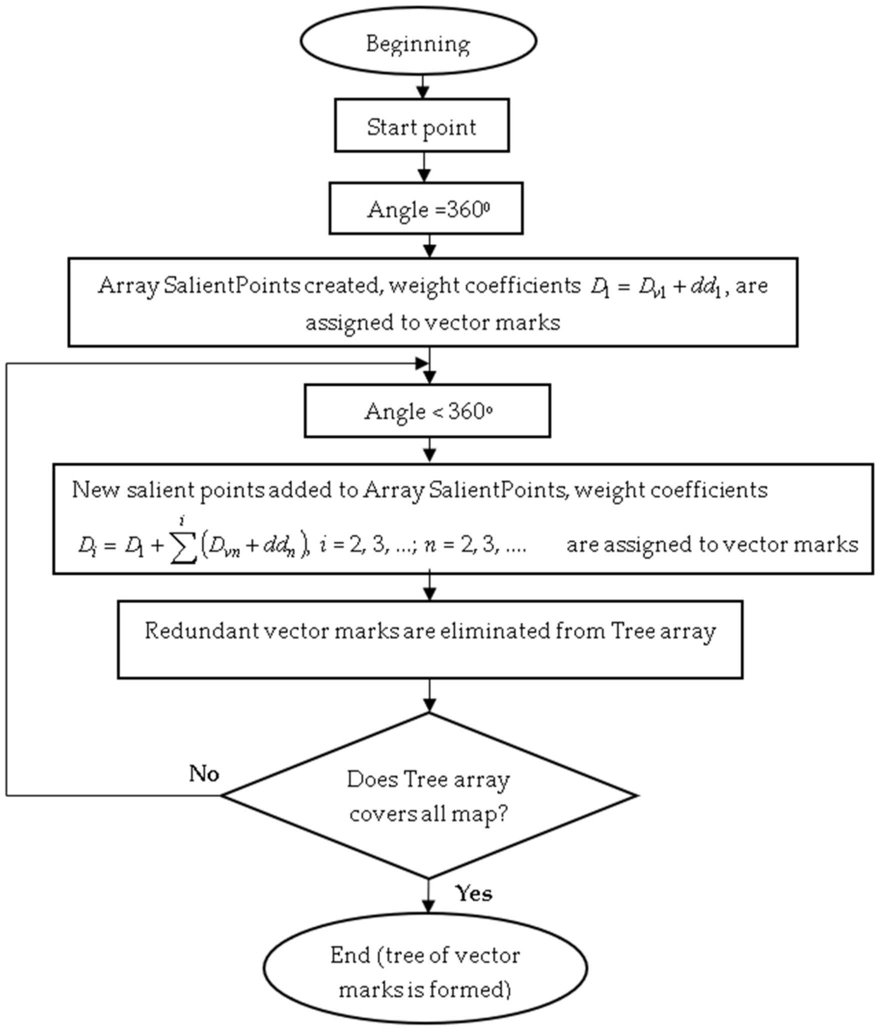
Beginning
Start point
Angle =360⁰
Array SalientPoints created, weight coefficients $D_1 = D_{v1} + dd_1$, are assigned to vector marks
Angle < 360°
New salient points added to Array SalientPoints, weight coefficients $D_i = D_1 + \sum^{i}(D_{vn} + dd_n)$, $i = 2, 3, \ldots; n = 2, 3, \ldots$ are assigned to vector marks
Redundant vector marks are eliminated from Tree array
Does Tree array covers all map?
No
Yes
End (tree of vector marks is formed)

Figure 4. Algorithm of vector mark generation.

The vector is defined by its end point, which is also the origin of the vector. The vector's length is equal to the given size, so that it can be used to locate other features of the current scene, such as gaps in the wall or door, by generating new vectors from end of it. This method helps to detect gaps and holes on parallel planes. Let the robot size be as the material point. Let the obstacle (see Figure 5) between the robot's current position "A" and the target "D" be a fragment of a circle. From the target point "D", a tangent to the obstacle is drawn and the new vertex "C" is fixed on this tangent. Thus, the path does not cross the obstacle. Obviously, this path is not the shortest path from "A" to "D". On the tangent, from point "C", a vector mark is formed at the point of contact "K", pointing to the target "D". Another tangent to the obstacle (point of contact "B") is drawn from the current robot position "A" until it crosses the vector mark "CK". When the robot passes point "B" at a fixed distance h along this tangent from point "F", a new tangent to the obstacle (point of contact "L") is drawn. The robot moves along the new tangent. This process is repeated until the robot's path is aligned with the tangent "CK". The result of this process is a polygonal path that more accurately approximates the optimal path.

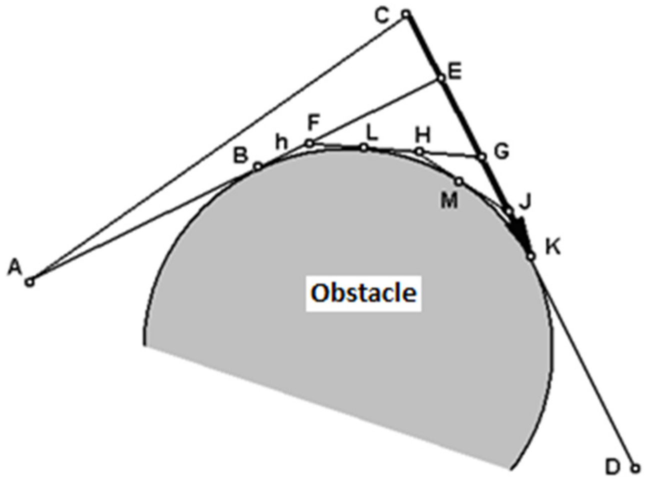

Figure 5. Approximated path and vector mark.

The accuracy of the approximation depends on the value of h. All tangents drawn from the current robot position intersect with the vector "CK". In other words, the vector "CK" is visible from the robot's current position and is a vector mark. The length of the mark must be at least as long as the distance "EK", otherwise the robot at point "A" will not see the vector mark. The total length of the vector mark must be long enough to form the tag, and at least long enough for the end of the mark to be visible. A path consisting of tangent fragments is obviously shorter than a polygonal path.

Evidently, the robot's path is close to the shortest path if the robot's movement step h is not too large. If the lengths of several different paths are similar and correspond to the shortest required path, the criterion for selecting the visible vector mark is quite simple. Figure 6 shows the case of a known environment, where a circular obstacle is avoided in a virtual environment by estimating the dimensions of the virtual robot.

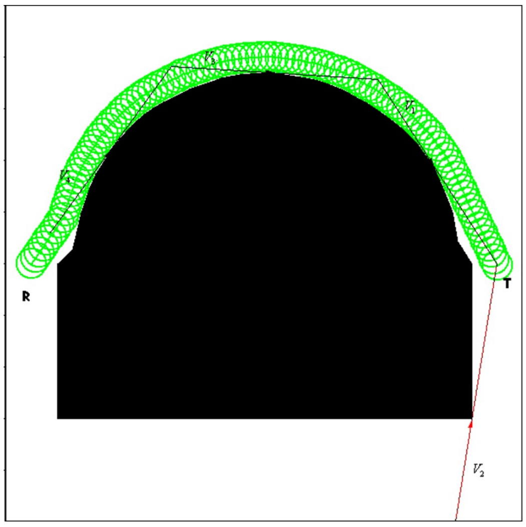

Figure 6. Path of the robot.

For the environment depicted in Figure 6, the target point is denoted by "T" and the robot's current position is denoted by "R". A global search is performed for the path between the points "R" and "T". A tree of the vector is formed from the goal point. The first generation of vector marks V_1 and V_2 is formed.

Continuing further scanning from the end point of the vector mark V_1, at the point of contact of the obstacle, a vector mark V_3 of known length is formed, equal to the length of the vector mark V_1. The next point of contact of the obstacle is found, where a vector mark V_4 is being formed. In this way, vector marks are formed at the points of contact of a circular static obstacle. In the case shown in Figure 6, the dimensions of the simulated virtual robot are known. Thus, the robot has to go from point "R" to the goal point "T" and then it has to bypass a round static obstacle. In the simulated case, a trajectory is constructed in the virtual space based on the vector tree and the robot's dimensions. As can be seen in Figure 6, the virtual robot smoothly avoids the obstacle and follows the shortest possible path according to the generated trajectory. Figure 6 shows how the approximation of the path depicted in Figure 5 is implemented in the software.

The analysis is based on a storage room where boxes of fastening details arrive stacked on pallets. They are then to be removed to the appropriate shelves. There are several rows of the double-sided racking with a gap of 2800 mm between them. The storage room is depicted in Figure 7.

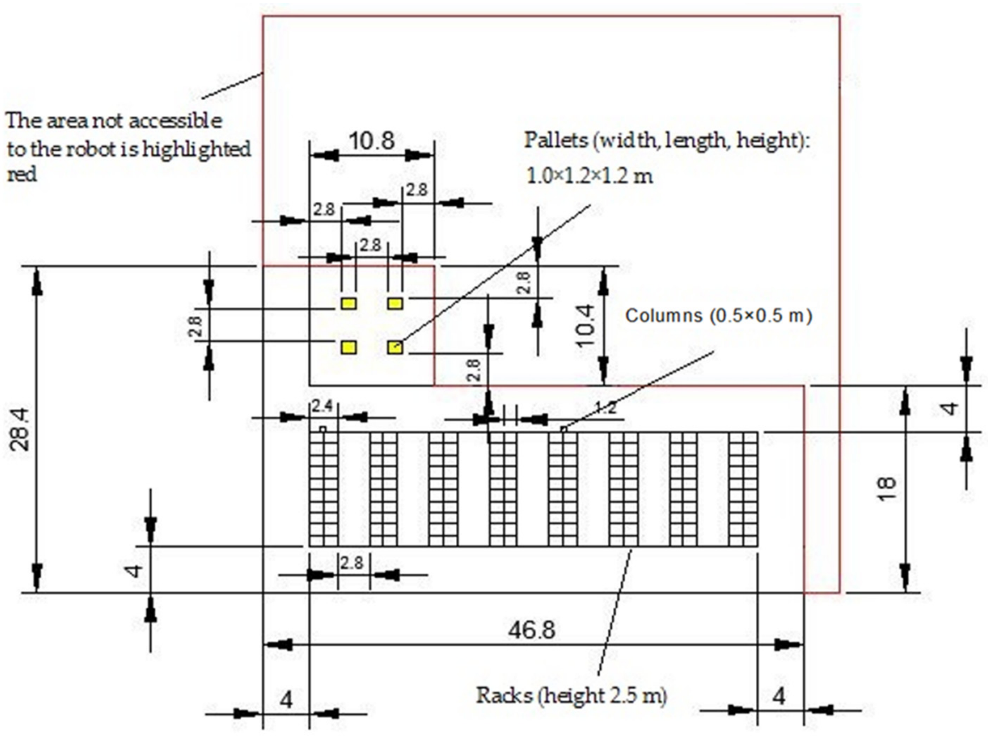
The area not accessible to the robot is highlighted red
10.8
Pallets (width, length, height): 1.0×1.2×1.2 m
2.8
2.8
2.8
2.8
2.8
10.4
Columns (0.5×0.5 m)
2.8
28.4
2.4
1.2
4
18
4
2.8
46.8
4
Racks (height 2.5 m)
4

Figure 7. Storage room.

The minimum height at which the box can be picked/placed is 200 mm, and the maximum height is 2000 mm. The dimensions of the robot chassis are 800 mm $\times$ 1000 mm. Figure 7 shows that the space between the racks is 2080 mm, so that two mobile robots can pass.

4. Experimental Research

The software 'CentaurusCPNV16' is capable of solving mobile robot path planning tasks [25]. However, the authors of this paper do not solve collisions between dynamic objects here. Collisions with a static environment are analyzed and taken into account.

A model of the room designed for the experimental studies is shown in Figure 8.

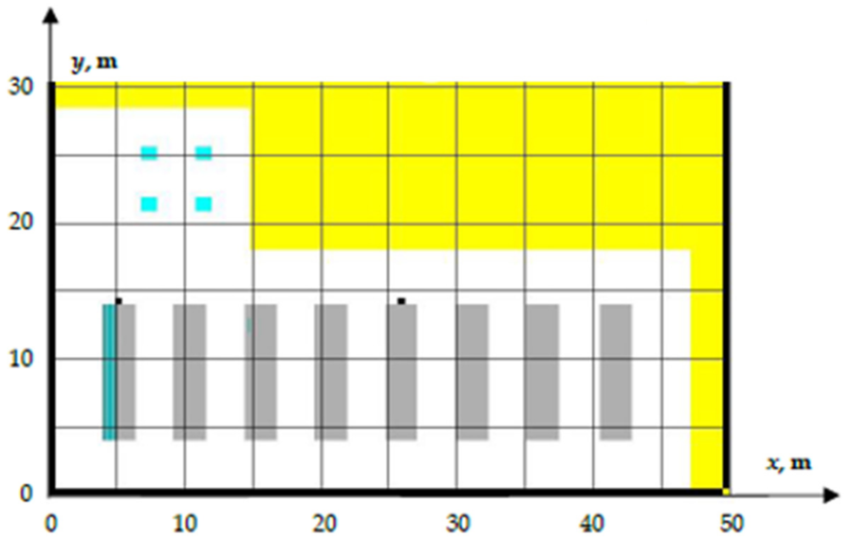

Figure 8. The model of the room.

The model has a yellow area for people; four pallets for the storage of boxes (4 blue squares); the boxes are placed in 16 racks (8 gray areas, each consisting of 2 racks). Two loading points are allocated for the mobile robot. They are marked with black squares in racks 1–2 and racks 9–10. The gaps between the racks are designed to allow a 1 m wide robot to pass through them.

The robot has to pick up the boxes from the pallets to transport and place them in the racks. Each rack can hold 60 boxes. The layout of the rack is shown in Figure 9. Figure 9 shows a general view of the rack, where the blue area indicates the available spaces. Each rack consists of 10 vertical shelves with six levels, which is the situation of the removal of boxes from the pallet to the selves of rack 1. The red color indicates the already occupied spaces, with boxes placed on three floors.

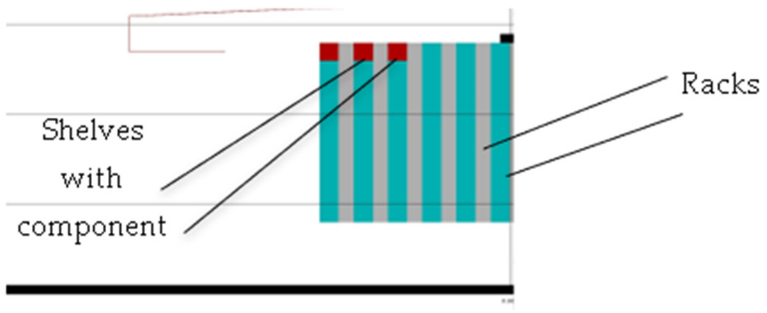

Figure 9. Rack layout and occupancy of the shelves.

The vector tree is formed by specifying its coordinates, the length of the vector mark, the radar distance insensitivity, and the radar angle of view (see Table 1).

Table 1. Parameters of the vector tree generation.

Parameter	Value
X target	7.5
Y target	22.5
Corbel	4.0
Distance gap of radar	1.0
Max generation	100.0
Minimal visible chord	1.0
View angle of radar (grad)	330.0

For the robot to transport the boxes and place them on the shelves, its trajectories need to be modeled. Based on the methodologies described above, a vector tree must be generated at the target point corresponding to the coordinates of the center of each vertical shelf of the rack. Based on this tree, the trajectory of the robot from the current location to the target point is calculated. In addition, the calculation of this trajectory considers the robot's dimensions.

According to this concept, two different vector tree systems can be used for a given system configuration:

- For delivering a load to the racks (160 vector trees);
- For arriving at the pick-up area from the place of delivery (4 vector trees).

The loads must not only be delivered to the racks but also unloaded, i.e., the reverse action must be undertaken by creating vector trees for the unloading process. In this situation, the question is whether to form databases of all possible vector trees, which would be selected for functionality based on the coordinates of the destination point, or to generate the required tree each time based on the coordinates of the destination.

The second case is more efficient as it does not require the creation and maintenance of a vector tree database. It is also necessary to consider that the preparation of the database takes additional time.

For the given room configuration (see Figure 8), the time required to generate a vector tree with modern computer technology is minimal. For example, the analysis of the vector tree shown in Figures 10 and 11 reveals that it takes 0.389 s to compute and graphically represent it.

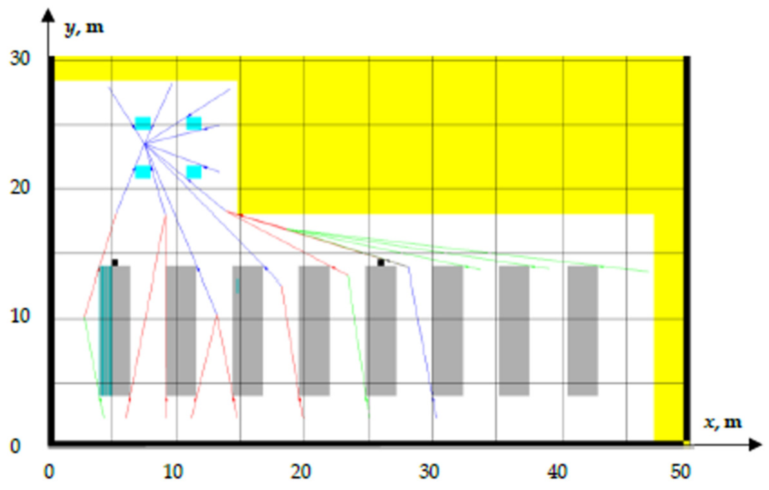

Figure 10. Maximum length of vectors limited by distances to obstacles (target point $x = 7.5$; $y = 22.5$).

Figures 12–17 show how the generated vector mark trees and the generated robot trajectories from different starting positions appear in different vector tree parameters (vector tag length, viewing angle, robot radius, and robot starting positions). The parameters, which can be changed, are shown in Table 2.

In this case (see Figure 13), the robot's path is shorter than that obtained using the minimum vector length (shown in Figure 12).

In this case, (see Figure 15), the robot's path is shorter than that obtained using the minimum vector length (see in Figure 14).

The experimental simulations and testing showed that, when generating vector trees, it is advisable to use the maximum length of the vector. This results in a shorter path for the robot to reach the goal. It can be seen in Figures 12 and 14. In these cases, more vector marks are formed, but the calculated trajectory of the robot is 8.5 proc. longer.

In this case, (see Figure 17), the robot's path is the same as the minimum vector length (shown in Figure 16).

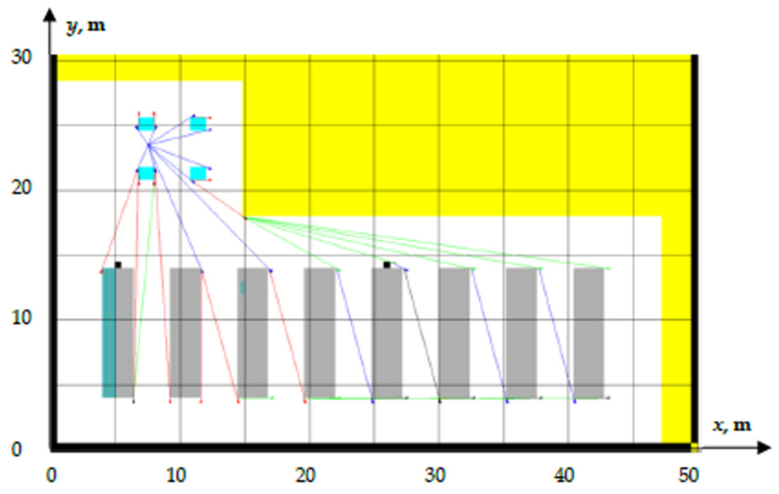

Figure 11. Minimum length of the vector (target point $x = 7.5$; $y = 22.5$. Condition: the length of the vector is assumed to be 1 cm).

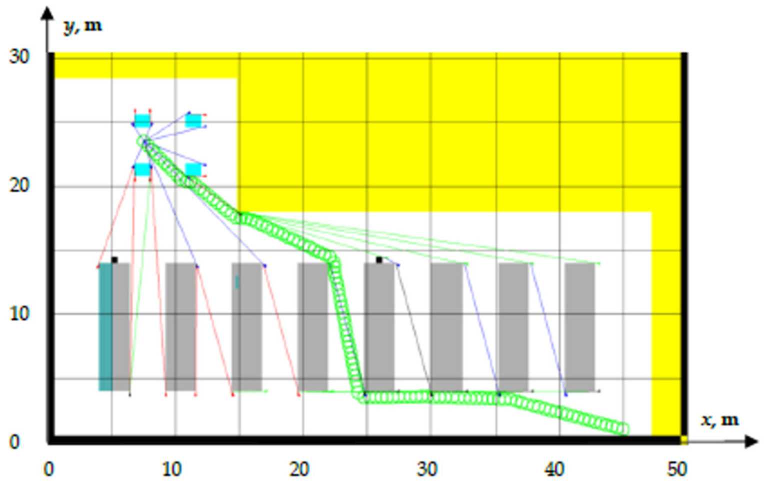

Figure 12. Minimum length of the vector (robot location coordinates $x = 45$; $y = 1$, target point coordinates $x = 7.5$; $y = 22.5$).

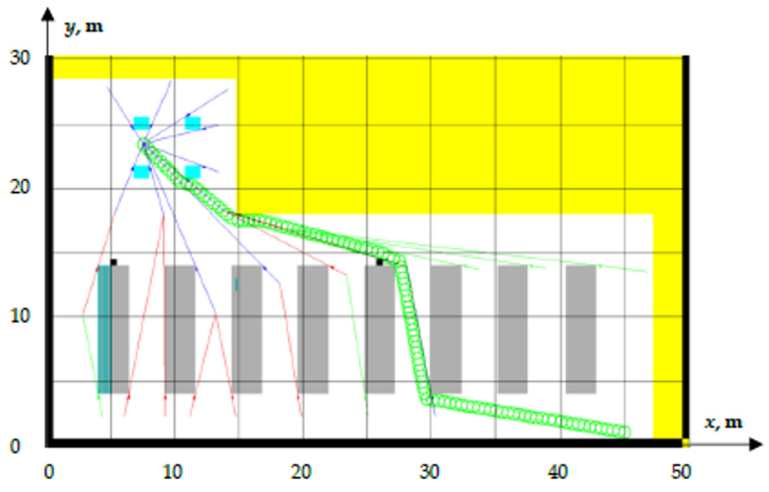

Figure 13. Maximum vector length (robot location coordinates $x = 45$; $y = 1$, target point coordinates $x = 7.5$; $y = 22.5$).

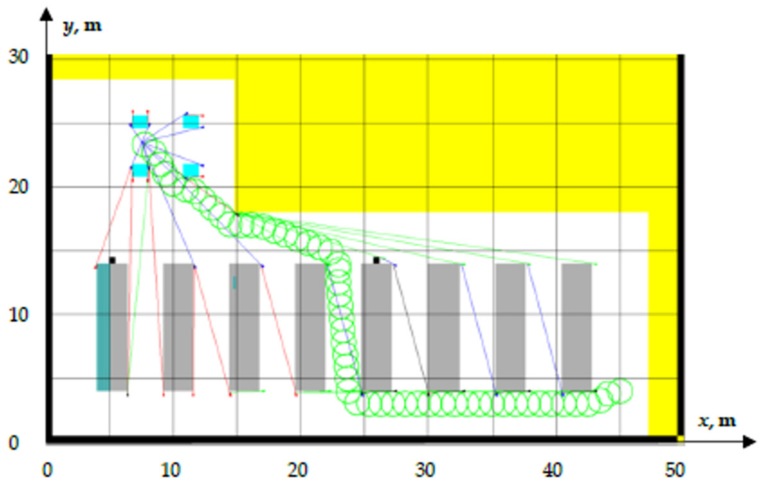

Figure 14. Minimum length of the vector (robot location coordinates $x = 45$; $y = 4$, target point coordinates $x = 7.5$; $y = 22.5$).

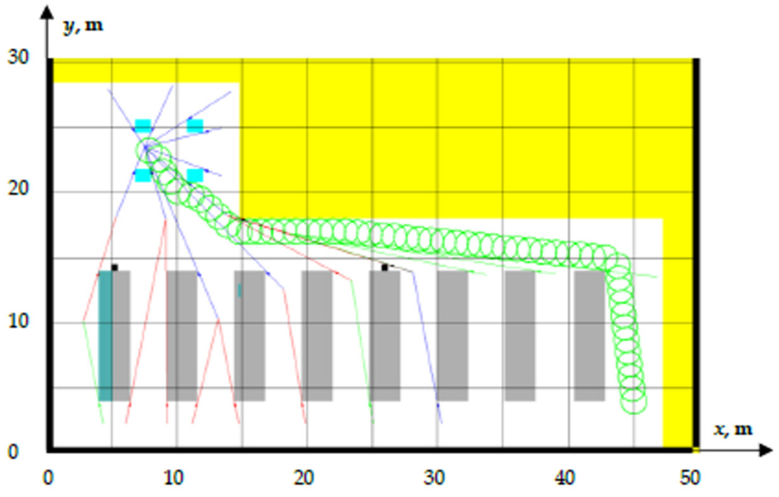

Figure 15. Maximum vector length (robot location coordinates $x = 45$; $y = 4$, target point coordinates $x = 7.5$; $y = 22.5$).

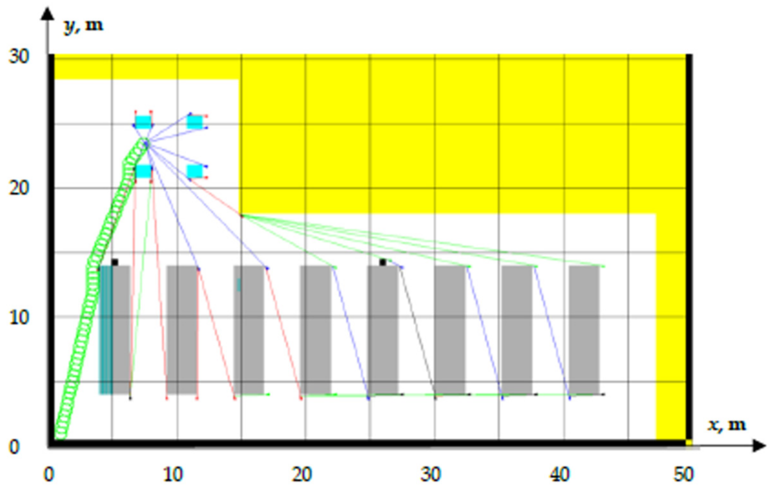

Figure 16. Minimum length of the vector (robot location coordinates $x = 1$; $y = 1$, target point coordinates $x = 7.5$; $y = 22.5$).

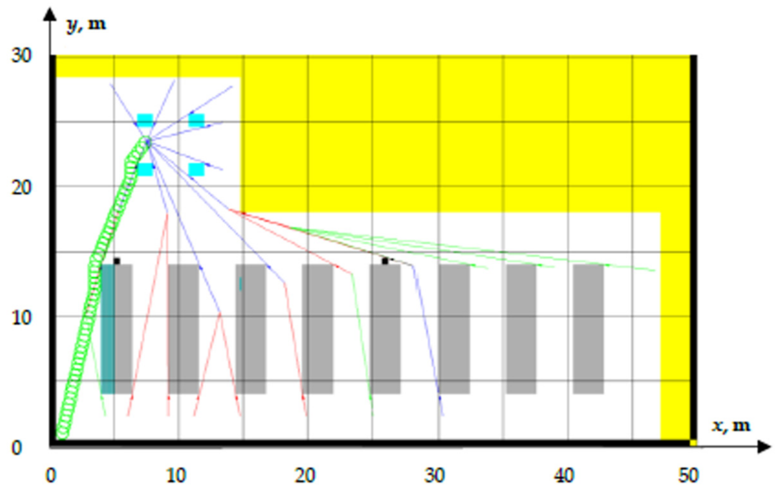

Figure 17. Maximum vector length (robot location coordinates $x = 1$; $y = 1$, target point coordinates $x = 7.5$; $y = 22.5$).

Table 2. Parameters of the trace calculation.

Parameter	Value
X target	7.5
Y target	22.5
Tolerance X	0.1
Tolerance Y	0.1
X robot	45.0
Y robot	1.0
Robot's radius	1.0

Using the maximum length of vector marks enables us to calculate the shorter path to the target point. During the testing simulations, the authors used limiting parameters: view of angle radar, different target point coordinates, different initial coordinates of the robot, and the length of the vector. A simulation-based control system for a logistic robot has been developed to simulate and represent the filling of storage shelves with boxes of a given size. The movement of logistic robots is not attached to any physical restrictions (such as tracks).

Discussion

The performance checking of the proposed algorithm and comparison with other researchers' works was performed. The experimental simulation results of the proposed method were compared with results obtained using the neural network field (NNF) and fuzzy logic methods, which are commonly used in a 2D environment. The experimental results are presented in Figure 18. Figure 18b shows that the path formed by the proposed algorithm in this paper was slightly shorter than the path obtained by the NNF, presented in Figure 18a [24]. Meanwhile Figure 18c [26] shows an example of navigation based on fuzzy logic, Figure 18d shows the experimental testing of the proposed algorithm. Furthermore, it is obvious that the proposed method showed better results (the path length is shorter) compared with fuzzy logic.

The authors can state that the proposed method of navigation is capable of bypassing obstacles of various shapes, avoids falling into local minima, and calculates the shortest possible path to the target point, as shown in Figure 18d.

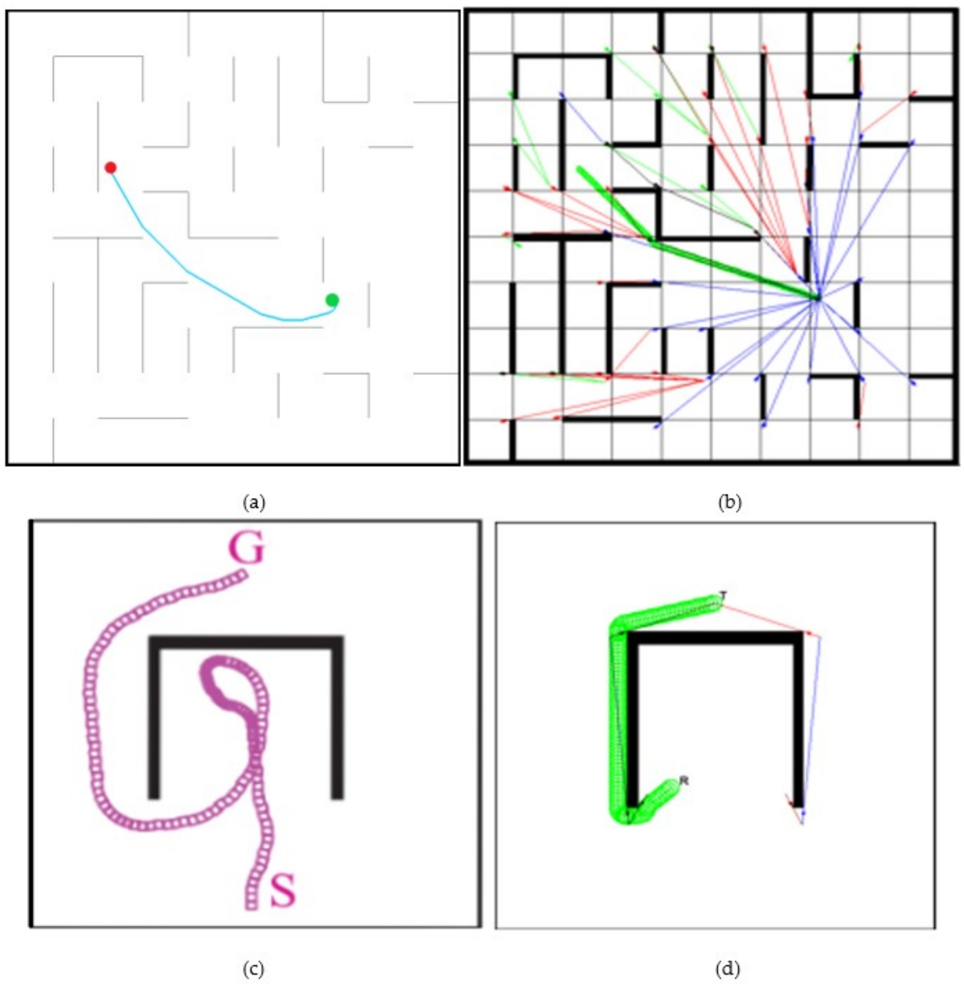

Figure 18. Experimental testing and verification. (**a**) NNF example; (**b**) proposed algorithm; (**c**) fuzzy logic example; (**d**) proposed algorithm.

5. Future Work and Conclusions

Further investigations and experimental testing of the proposed system are related to the ensuring stability of the calculated trajectory and positioning of the mobile robot. At any time, the robot might deviate from its planned trajectory, so the localization algorithm must be used to correct any deviations.

The algorithm of salient point's detection, vector mark estimation, and optimal path calculation is presented and is realized using colored Petri nets. Experimental investigations, using different modeling parameters, have shown that the usage of the maximum possible vector mark length enables the calculation of 8.5 percentages shorted path. The limitation of the radar view angle enables to avoid duplicating of salient points detection. The proposed approach was extended to simulate the work of a logistic robot, which has to take boxes and deliver them to storages.

Our experimental investigation showed that the proposed navigation algorithm is able to find the shortest path in the known 2D environment and avoids falling into local minima in various shapes of the environment. The occupation of certain places in storage is visualized and shown in experimental graphics.

Author Contributions: Conceptualization, K.K.S., V.B. and O.S.; methodology, K.K.S., V.B. and O.S.; software, K.K.S.; validation, J.G., L.B. and A.D.; formal analysis, L.B. and A.D.; investigation, K.K.S., J.G., V.B. and O.S.; resources, O.S.; data curation, V.B.; writing—original draft preparation, K.K.S., J.G., L.B., V.B., O.S. and A.D.; writing—review and editing, K.K.S., J.G., L.B., V.B., O.S. and A.D.;

visualization, V.B.; supervision, L.B. and K.K.S.; project administration, O.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors received specific funding for this study from the competition of research projects for the development of the KTU research groups (PP32/1807).

Data Availability Statement: Data are contained within the article.

Acknowledgments: Authors thank Stanislovas Kęstutis Bartkevičius for his support during the experimental testing of the proposed system.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Patle, B.K.; Babu, L.G.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
2. Ab Wahab, M.N.; Nefti-Meziani, S.; Atyabi, A. A comparative review on mobile robot path planning: Classical or meta-heuristic methods? *Annu. Rev. Control* **2020**, *50*, 233–252. [CrossRef]
3. Mountz, M.C.; Wurma, P.R. Method and System for Retrieving Inventory Items. U.S. Patent 7894933, 22 February 2011.
4. Raffaello, D.; Mansfield, P.K.; Mountz, M.C.; Polic, D.; Dingle, P.R. Method and System for Transporting Inventory Items. U.S. Patent 8280547, 2 October 2012.
5. Raffaello, D. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Trans. Autom. Sci. Eng.* **2012**, *9*, 638–639.
6. Wise, M. Fetch and freight: Standard platforms for service robot applications. In *Workshop on Autonomous Mobile Service Robots*; Fetch Robotics Inc.: San Jose, CA, USA, 2016; Available online: http://docs.fetchrobotics.com/FetchAndFreight2016.pdf (accessed on 18 October 2022).
7. Marder-Eppstein, E.; Berger, E.; Foote, T.; Gerkey, B.; Konolige, K. The office marathon: Robust navigation in an indoor office environment. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010.
8. Macenski, S.; Plaza Mart'in, F.J.; White, R.; Clavero, J.G. The marathon 2: A navigation system. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020.
9. González, D. A review of motion planning techniques for automated vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1135–1145. [CrossRef]
10. Khairuddin, A.R.; Talib, M.S.; Haron, H. Review on simultaneous localization and mapping (SLAM). In Proceedings of the 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 27–29 November 2015.
11. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016.
12. Tsardoulias, E.G. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *J. Intell. Robot. Syst.* **2016**, *84*, 829–858. [CrossRef]
13. Belanová, D.; Mach, M.; Sinčák, P.; Yoshida, K. Path planning on robot based on D Lite algorithm. In Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA), Kosice, Slovakia, 23–25 August 2018.
14. Zhu, Z.; Xie, J.; Wang, Z. Global dynamic path planning based on fusion of A* algorithm and dynamic window approach introduction to A*. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019.
15. Seder, M.; Petrovic, I. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007.
16. Li, Y.; Zhu, Q. Local path planning based on improved dynamic window approach. In Proceedings of the 40th Chinese Control Conference, Shanghai, China, 26–28 July 2021.
17. Rösmann, C.; Hoffmann, F.; Bertram, T. Planning of multiple robot trajectories in distinctive topologies. In Proceedings of the 2015 European Conference on Mobile Robots (ECMR), Lincoln, UK, 2–4 September 2015.
18. Rösmann, C.; Hoffmann, F.; Bertram, T. Kinodynamic trajectory optimization and control for car-like robots. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017.
19. Magid, E.; Lavrenov, R.; Khasianov, A. Modified spline-based path planning for autonomous ground vehicle. In Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotic (ICINCO 2017), Madrid, Spain, 26–28 July 2017.
20. Amer, N.H.; Zamzuri, H.; Hudha, K.; Kadir, Z.A. Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges. *J. Intell. Robot. Syst.* **2017**, *86*, 225–254. [CrossRef]
21. Sun, W.; Chen, X.; Zhang, X.; Dai, G.; Chang, P.; He, X. A multi-feature learning model with enhanced local attention for vehicle re-identification. *Comput. Mater. Contin.* **2021**, *69*, 3549–3561. [CrossRef]
22. Sun, W.; Zhang, G.; Zhang, X.; Zhang, X.; Ge, N. Fine-grained vehicle type classification using lightweight convolutional neural network with feature optimization and joint learning strategy. *Multimed. Tools Appl.* **2021**, *80*, 30803–30816. [CrossRef]

23. Digani, V.; Sabattini, L.; Secchi, C.; Fantuzzi, C. Ensemble coordination approach in multi-AGV systems applied to industrial warehouses. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 922–934. [CrossRef]
24. Chen, Y.; Cheng, C.; Zhang, Y.; Li, X.; Sun, L. A neural network-based navigation approach for autonomous mobile robot systems. *Appl. Sci.* **2022**, *12*, 7796. [CrossRef]
25. Bartkevicius, S.; Fiodorova, O.; Knys, A.; Derviniene, A.; Dervinis, G.; Raudonis, V.; Lipnickas, A.; Baranauskas, V.; Sarkauskas, K.; Balasevicius, L. Mobile robots navigation modeling in known 2D environment based on Petri nets. *Intell. Autom. Soft Comput.* **2018**, *24*, 241–248. [CrossRef]
26. Tao, Z.; Haodong, L.; Songyi, D. Multi-robot Path Planning Based on Improved Artificial Potential Field and Fuzzy Inference System. *J. Intell. Fuzzy Syst.* **2020**, *39*, 7621–7637.