applied sciences

MDPI

Article

Real-Time Path Planning for Strategic Missions

João Vítor R. Vasconcelos [1], Alexandre S. Brandão [1,*] and Mário Sarcinelli-Filho [2]

[1] Núcleo de Especialização em Robótica (NERO), Department of Electrical Engineering, Graduate Program in Computer Science, Universidade Federal de Viçosa, Viçosa 36570-900, Minas Gerais, Brazil; joao.v.vasconcelos@ufv.br

[2] Graduate Program in Electrical Engineering, Universidade Federal do Espírito Santo, Vitória 29075-910, Espírito Santo, Brazil; mario.sarcinelli@ufes.br

* Correspondence: alexandre.brandao@ufv.br

Received: 22 September 2020; Accepted: 30 October 2020; Published: 3 November 2020

Featured Application: Our proposal has application in a real problem, whose objective is to guide a mobile robot to a target point in an environment that changes along time. These changes are characterized by the addition of obstacles that prevent navigation or the inability to follow a previously computed route, for instance scenarios of recent hazards or natural disasters.

Abstract: Robot navigation is still an open research topic, mainly regarding applications that require online planning. In this context, this manuscript presents an implementation of the Lifelong Planning A* (LPA*) search algorithm to guide a mobile robot in an environment that changes along time, by detecting obstacles and updating the current mapping information. Firstly, simulations validate the strategy, and later experiments confirm these results considering a real application. The result is that the LPA* algorithm is able to guide the mobile robot to its target in a changing environment. Obstacles are interactively included in the scene to force the route redesign and the seeking for a new optimal solution connecting the current robot position and its target position.

Keywords: mobile robots; path planning; shortest path problem; service robots

1. Introduction

In recent decades, mobile robotics has been successfully used in many applications, such as security, industry and military missions, for instance. To perform these missions, a robot should be able to navigate and explore the environment autonomously. Often, path-planning is one of the problems to be solved, which involves the search of an optimal or sub-optimal route from the initial position to a desired one in its workspace [1].

In a dynamic environment, such as those characteristic of war scenarios or disaster zones, route changes often have obstructed passages or loss of information about some region. In this case, the decision making by the robot must take place interactively with the environment, and the robot has to update the route to its destination each time a change occurs in the environment. Thus, it disrupts momentously the previous/current computed path and starts following a new one, resulting in a local planning performed upon detecting an obstacle or loosing information about a certain zone by which it should pass.

From a military point-of-view, the authors of [2] mention that in recent years the US Army has incorporated robots to reduce the number of humans in their strategic actions. As a result, the US Army has invested heavily in research and development of autonomous systems [3]. With a similar purpose, intelligent agents have been used in natural disaster sites to minimize the losses, so that intelligent disaster rescue operations are becoming more and more common [4–6]. It is important to notice that in both scenarios the environment can change during mission execution, therefore requiring

the incorporation of a path-planning strategy that is able to update the planned path whenever an unforeseen or unexpected event occurs.

There are several algorithms available in the literature to deal with route optimization, such as Dijkstra algorithm [7], A* algorithm [8] and Lifelong Planning A* (LPA*) algorithm [9]. All of them can solve path-planning problems and find an optimal solution. The main difference among them is the complexity level of the algorithm.

All of these three algorithms deal with the path-planning problem, finding an optimal route for the robot, with different execution times. When a change in the environment occurs, such as the addition/appearance of an obstacle, if that change affects the previously found path, the Dijkstra and A* algorithms solve the path planning problem again, while the LPA* takes advantage of the already obtained solution for this new problem. Thus, this last approach addresses the problem more quickly and effectively.

In this context, this paper discusses the application of the LPA* algorithm in a real problem, whose objective is to guide a mobile robot to a target point in an environment that changes along time. These changes are characterized by the addition of obstacles that prevent navigation or the inability to follow a previously computed route, emulating scenarios of recent hazards or disasters.

After this short introduction, this paper is hereinafter organized as follows: Section 2 presents some related works, and Section 3 introduces the LPA* algorithm and how it is implemented to deal with a navigation problem. The description of the navigation environment and the results obtained with a mobile robot evolving in such an environment, as well the pertinent discussions, are presented in Sections 4 and 5, respectively. Finally, Section 6 highlights the main contributions and concluding remarks of this paper.

2. Related Works

Several strategies have been proposed to autonomously guide robots in dynamic environments. Some of them, related to this work, are commented below.

An intelligent motion planning with dynamic obstacle avoidance is proposed in [10]. Two Sugeno fuzzy logic controllers work separately to handle collision avoidance tasks. Four behavior controllers are responsible to guide the navigation. They are goal-reaching, goal searching, speed control and obstacle avoidance. This last one is split in two others, to separately deal with static and dynamic obstacles.

Oliveira and colleagues contributed with a new navigation strategy called Dynamic Planning Navigation Algorithm optimized with genetic algorithm (DPNA-GA) in [11], which ensures that the robot reaches the goal in an unknown environment containing static and dynamic obstacles. In summary, the robot follows a path composed by location goals and the destination.

Patle, in [12], uses the optimized firefly algorithm, and applies it to mobile robot navigation. For this, a zone was made around the agent representing the light intensity of a firefly and from that the algorithm is able to detect and avoid any object within that safe zone in a static environment. In [13], a genetic algorithm improves robot decisions and enables navigation in dynamic environments.

A new path planning methodology based on Ant Colony Optimization is presented in [14] for environments with static obstacles. The authors consider the obstacles in the straight segment that connect the starting point with the goal point, as critical ones, and use such an information to improve the optimization process. Then, they introduce the Ant Colony Optimization with the Influence of Critical obstacle, or simply ACOIC.

In [15], the authors deal with the path planning problem for platforms with limited sensor and processing capabilities. They assume that a mapping strategy is available, however the robot has no prior information about the environment. As the robot navigates towards the target, it stores information that enables it to generate sub-optimal routes using the A* searching algorithm. The more information the robot has about the environment, the closer to the optimal route the obtained solution is. Now, considering a known dynamic environment, the work of Luan [16] uses a search algorithm to

find an optimal way to get to the desired position, and during task execution the algorithm is able to deviate and recompute routes, thus avoiding collisions with dynamic obstacles. In addition, the hybrid navigation algorithm proves to be more efficient than others known in the literature and presents excellent performance in environments with many dynamic obstacles.

In turn, in [17], an improved genetic algorithm finds an optimal path in a dynamic environment. This work considers that the entire environment is known by the algorithm and the algorithm considers that the robot is a particle, ignoring its dimensions. The work guarantees that the algorithm finds the optimal path, better than A* and D* (which commonly find sub-optimal paths).

All the works listed in this section apply some optimization algorithm to perform a path planning. Some of them for static and other for dynamic environments. Thus, they give theoretical support to the strategy proposed in this paper.

3. The Path Planning Strategy

Figure 1 illustrates the flowchart of the path planning strategy for a mobile robot, considering the possibility of environment changes, such as the appearance of unknown or unpredicted obstacles. Algorithm 1 assists the flowchart interpretation.

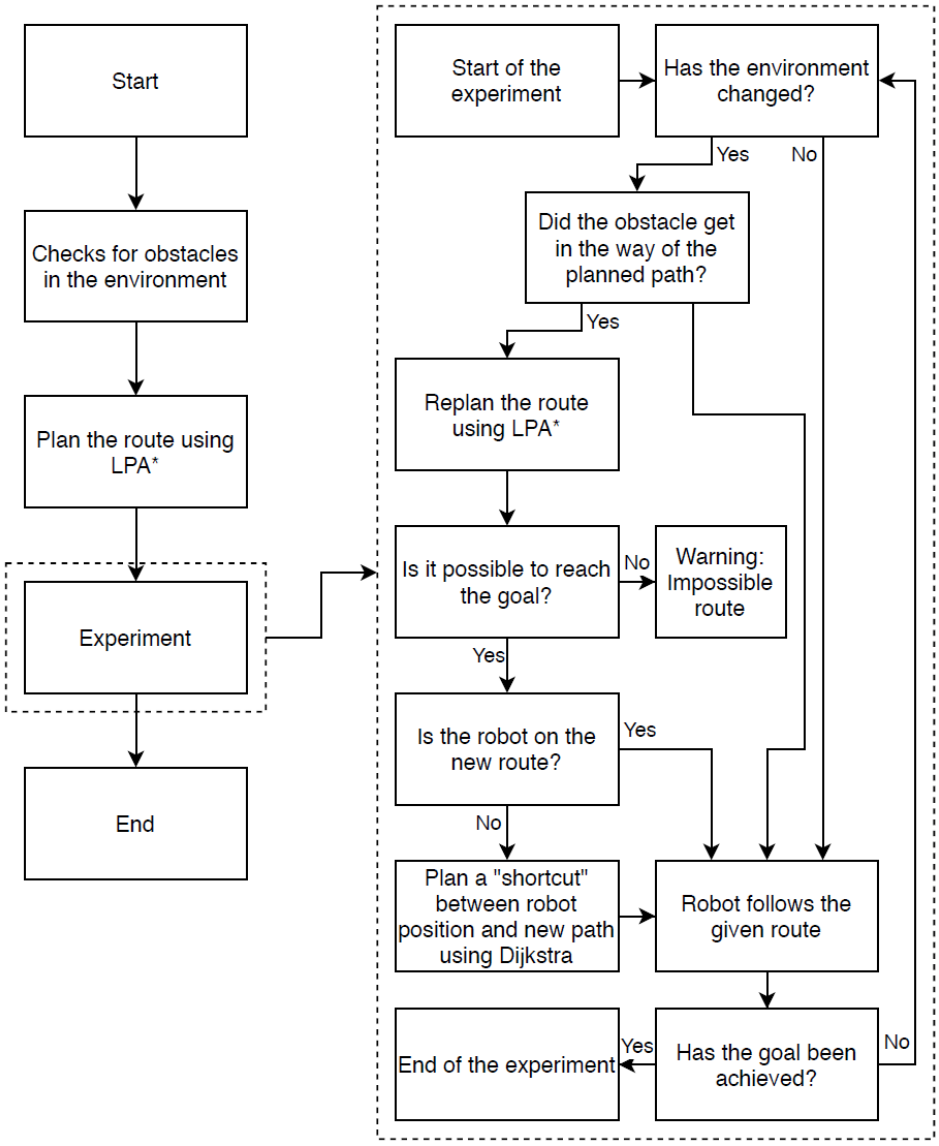


Figure 1. The route planning strategy flowchart for a mobile robot.

Initially, the presence of obstacles in the environment is verified. After checking, the LPA* algorithm [9] provides a feasible route. If there are any changes in the environment that make it impossible to follow the route previously found, the Algorithm 1 recomputes and redesigns the route. If a feasible solution cannot be found to reach the goal, the robot stops (or returns to the starting position) and sets an "Impossible Route" warning. If, after replanning, the robot is not on the calculated new route, a collision-free "shortcut" is established, using Dijkstra's algorithm. Finally, the mission is accomplished once the robot reaches the goal.

From our point of view, the main contribution of this paper is the way the LPA* and Dijkstra algorithms collaborate to find a feasible and optimal path linking the robot to its goal, whenever an unpredicted or unforeseen event occurs.

The following functions were used to implement the LPA* algorithm:

- $T(n)$ is the transition function that takes a n state and returns all successor states of n.
- $g(n)$ returns the cost between the initial S_{init} state and the n state.
- $C(n, c)$ returns the cost of the transition from state n to state c.
- $h(n)$ is a heuristic function that returns an estimation of the cost between state n and goal S_g.
- $f(n)$ returns the result of $g(n) + h(n)$.

Algorithm 1: LPA*

Input: S_{init}, S_g, *CLOSED*
Output: Route from S_{init} to S_g
if *CLOSED is empty* **then**
  Insert S_{init} in OPEN and CLOSED
else
  OPEN gets CLOSED and it is ordered based on f
  while *OPEN isn't empty* **do**
    n receives and takes first element of OPEN
    if $n = S_g$ **then**
      Returns the route between S_{init} and n
      break
    end
    i receives all successor states from state n by applying the transition function $T(n)$
    for *c ranging in each state of i* **do**
      if *c not in CLOSED* **then**
        Insert c in OPEN and CLOSED
      end
      if *c in CLOSED and* $f(c) > g(n) + C(n, c) + h(c)$ **then**
        Upgrade $g(c)$ gets $g(n) + C(n, c)$
        Upgrade $f(c)$ using the new value of $g(c)$
      end
    end
    Order based on f states in OPEN in ascending order
  end
end

4. The Description of the Navigation Environment

To perform the experiments, a 5×5 square grid map measuring 3×3 m is used, as shown in Figure 2a. The black and red dots are the initial S_{init} and the objective S_g states, respectively. Figure 2b indicates what each value of the map and graph means, which is represented in blue.

It is adopted that if the object occupies the grid space, it is treated as an obstacle. In the process of defining the successor, the center of each cell is assumed as a node. In this paper, the system of 4 nodes is adopted, i.e., 4-neighborhood. To calculate the heuristic function, the Euclidean distance will be used, described by

$$h(n) = d(n, S_g) = \sqrt{(x_n - x_{S_g})^2 + (y_n - y_{S_g})^2}, \quad (1)$$

where x_n and y_n represent the current robot position, while x_{S_g} and y_{S_g} are the goal position.

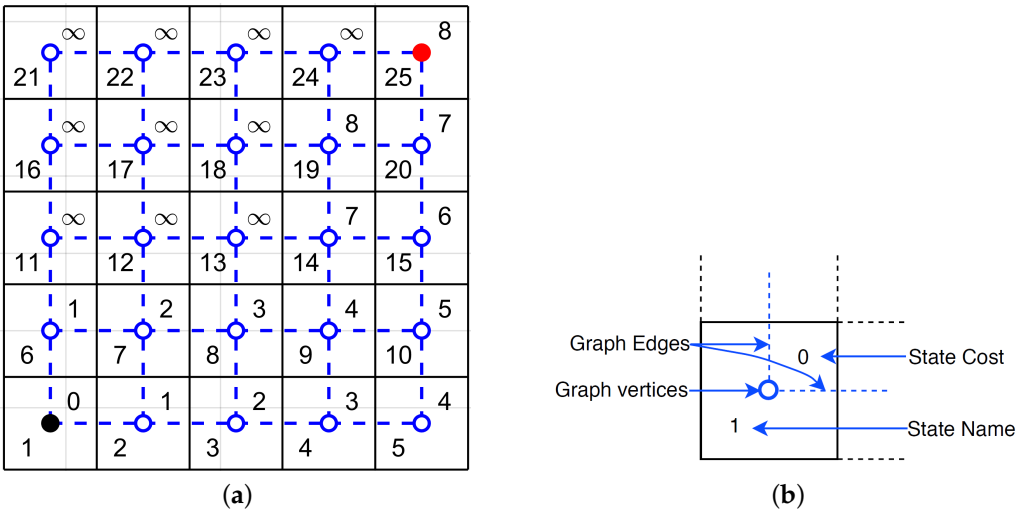


Figure 2. (**a**) Map and graph used in the experiment and (**b**) its characteristics. It is noteworthy that when the state cost is ∞, the node or state is not opened.

5. Results and Discussion

This section presents the simulation and experimental results, which validate the proposed navigation in a changeable environment. The objective is to simulate war or natural disaster scenarios, where the previously defined route must be changed due to unexpected obstructions.

Before validating the proposed algorithm in dynamic scenarios, it was compared with others that have a similar behavior.

5.1. A Brief Comparative Analysis

Considering that a comparative analysis can be made on the quality of a path planning strategy, this subsection demonstrates that the proposed strategy works in a similar (and often better) way than the proposals in [11–14]. Thus, the purpose of the following simulations is two-fold: validate the proposed strategy and compare it with others as well in a static environment.

Figures 3–5 show the comparison with five works, which aims to navigate in a structured or dynamic environment. Before starting the comparison, it is worth mentioning the following considerations: this paper performs (a) an implementation of the LPA* algorithm, (b) a Manhattan navigation system is used to describe robot navigation, (c) the entire environment and any change on it are considered known (or can be detected and reported to the planner) and (d) all simulations are performed with similar/identical environment characteristics (size, discretization, position of the objects and so on). When analyzing the figures, the routes resulted from our strategy correspond to the black lines, whereas the routes resulting from the algorithms under comparison are those in gray.

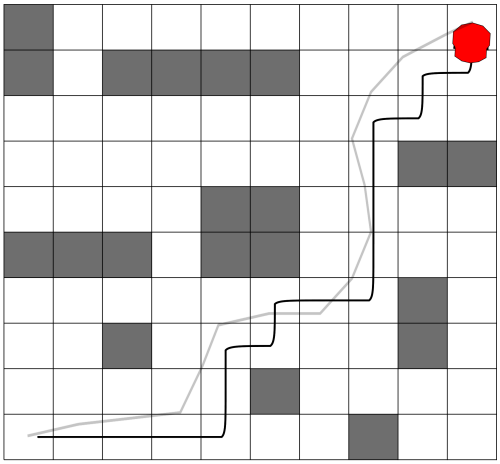
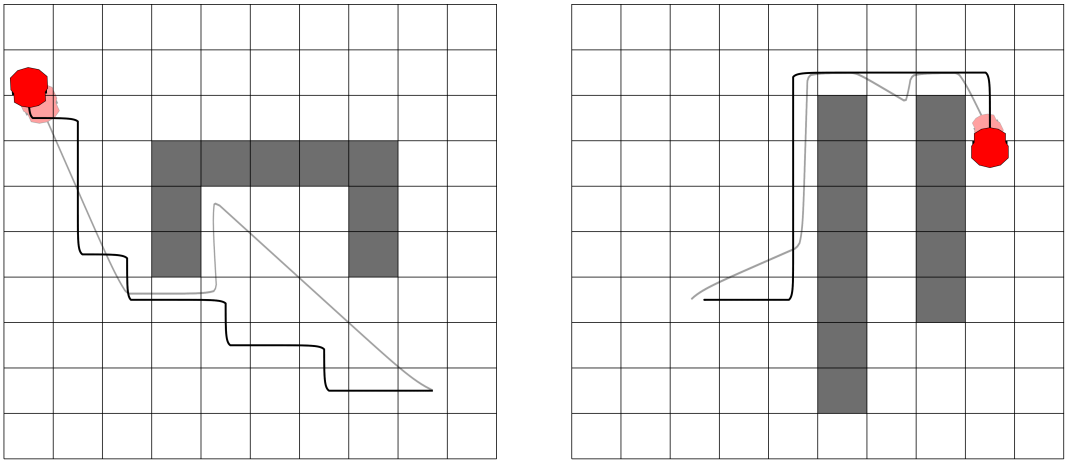


Figure 3. Comparison with Dynamic Planning Navigation Algorithm optimized with genetic algorithm (DPNA-GA) controller by Oliveira in [11].



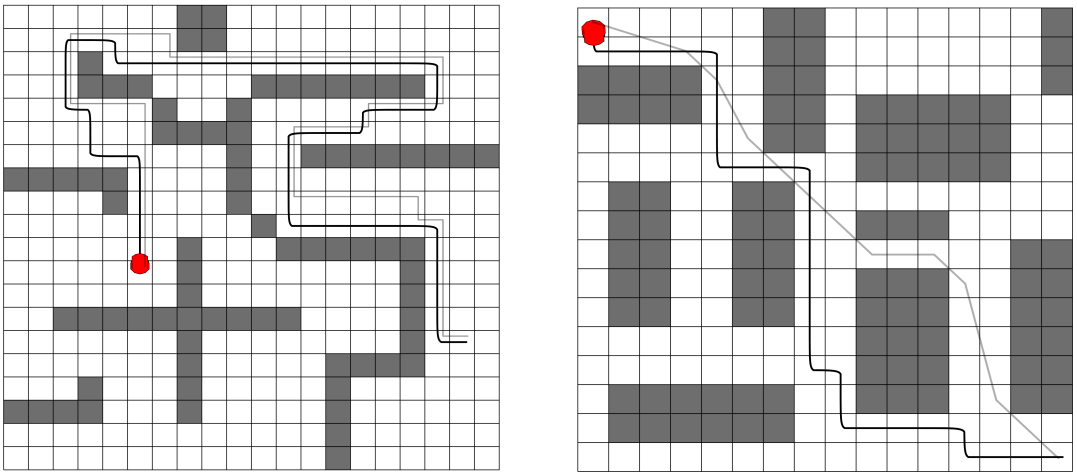
(**a**) (**b**)

Figure 4. Comparison with the (**a**) firefly algorithm by Patle in [12] and (**b**) Matrix-Binary Codes based Genetic Algorithm (MGA) controller by Patle in [13].

Figure 3 shows that the DPNA-GA has a strong influence of the obstacles on the path construction. In the same scenario, the LPA* responds similarly, but its computed route is closer to the obstacles, as shown in the figure. In this last case, notice that the agent does not perform unnecessary maneuvers to avoid the obstacles and, despite having its movements limited by Manhattan navigation, its route is still secure enough. Therefore, the LPA* algorithm is more efficient once the agent travels a shorter path and, consequently, spends less energy to reach the goal.

Figure 4a highlights that the firefly algorithm [12] performs a local optimization, which consequently makes the robot perform more movements than necessary to achieve the objective. In contrast, LPA* found the path that requires less movements to avoid the obstacle. In other words, the agent goes directly towards the target, overcoming the obstacle. Following, Figure 4b illustrates that the Matrix-Binary Codes based Genetic Algorithm (MGA) algorithm [13] is more efficient than the previous one, although still performing a reactive maneuver, i.e., the agent only avoids the obstacle after detecting it. In turn, as a result of a deliberative strategy, LPA* returns the path represented in black line, once it has the knowledge of the map previously.

Figure 5a shows the comparison with the ACOIC algorithm [14]. Notice that the traveled paths are almost identical, because both of them use the Manhattan description and work in a deliberative way. Finally, a similar behavior can be observed in Figure 5b, where the difference between the routes occurs because the IGA algorithm [17] uses 8-neighborhoods instead of 4-, as LPA* does.



(**a**) (**b**)

Figure 5. Comparison with (**a**) Ant Colony Optimization with the Influence of Critical (ACOIC) controller by Han in [14] and (**b**) Improved genetic algorithm (IGA) controller by Zhang in [17].

5.2. Scalability Analysis

In order to claim that the proposed algorithm is scalable, this section carries out simulations using a 5×5 and a 50×50 map. Other sizes of maps could be also considered, but we decide to exploit these and later explain how to deal with bigger or smaller ones. At the beginning, both maps are clean. All cells are free navigation zones. Then, the robot starts navigating, aiming to cross and reach the opposite side of the map. During navigation, the simulation code pauses six times, and blocked zones are randomly placed in the environment. After the last pause, approximately 24% of the maps are already labeled as occupied.

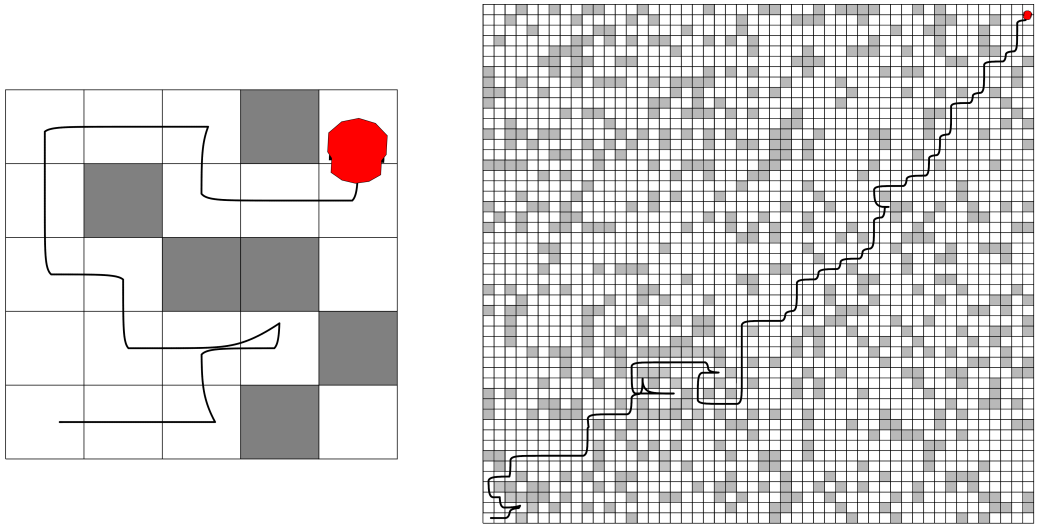
Whenever the code pauses, the robot stops, LPA* runs and returns a new optimal route joining the started position to the target position. After, Dijkstra runs and finds a "shortcut" joining the agent to the route computed by the LPA* algorithm. Then, the robot resumes its navigation according to the new computed route. Figure 6 illustrates the last stop situation and the feasible solutions found by the proposed strategy in both cases.

The computer used to run the simulations has a Gigabyte B450M motherboard, an AMD Ryzen 5 1600 3.6 GHz processor, 16 GB 3000 MHz RAM memory and runs MATLAB® software. Taking into account such hardware and software setup, the average times spent to find a feasible solution for the 5×5 and 50×50 maps are 24.6 and 58.2 ms, respectively.

As one can see, the larger the map is, the longer the time required to find a way to the destination (whenever possible) is. However, taking the mission covered in this work, it allows the robot to take the time to find the best route. After all, the work deals with an exploration mission. Hence, we assume that to take the necessary time to find a feasible solution is safer than to interrupt the search process and present a sub-optimal solution.

It is worth stressing that the scope of the work is not to propose a better search algorithm, but to present an application of a strategy to improve the use of the LPA* and Dijkstra algorithms. Further, one way out to deal with giant maps would be to subdivide them into sectors containing smaller grids. Therefore, the algorithm would have to calculate a path showing which sectors the robot should pass through, and within each sector there would be another route conducting the robot to the

next sector. Nevertheless, this strategy goes beyond the proposal of the present work, thus becoming a suggestion for its continuity.



(**a**) (**b**)

Figure 6. Simulation to check the computational cost for the code in (**a**) a 5×5 grid and (**b**) a 50×50 grid.

5.3. Experiment 1: Goal Seeking

To perform the experiments, Figure 7 illustrates the scenario to be described by a navigation grid. The mobile robot Pioneer 3DX simulates the military agent or rescuer. The environment has an Optitrack motion capture system, which emulates a GPS in real outdoor environments. It is worth noting that obstructions, obstacles and obstructions of routes in general are also detected by this tracking system in order to provide the necessary information for the route planning algorithm.

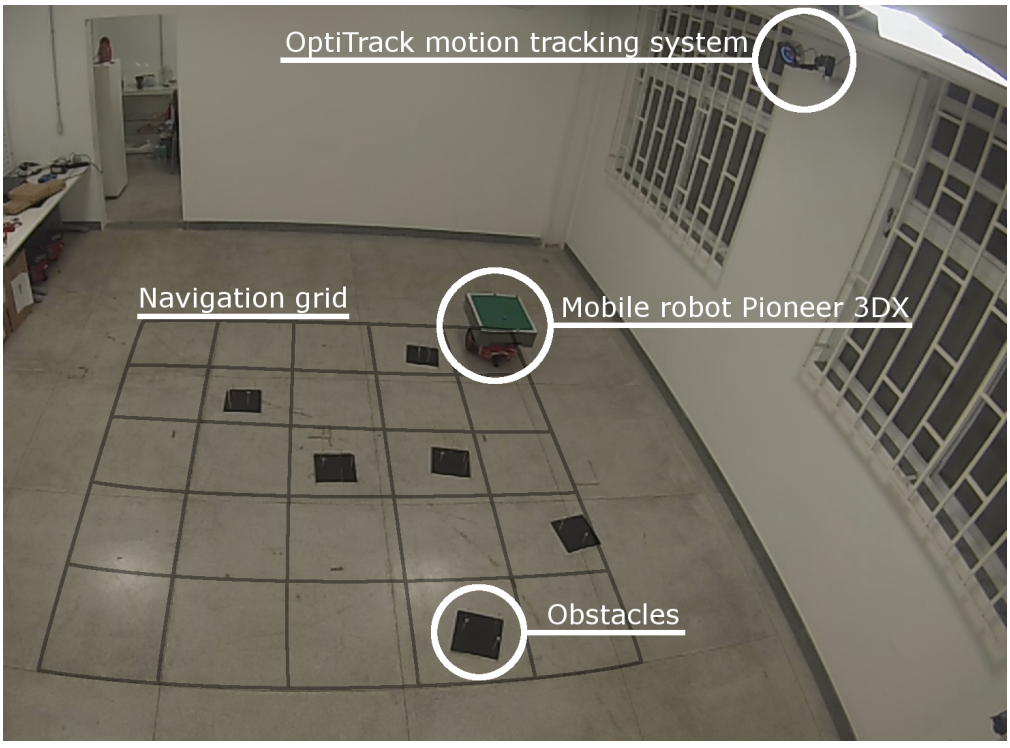


Figure 7. The scenario to be described by a navigation grid.

For the resulting figures, the solid black line indicates the path the robot followed, the solid green line indicates the route planned by the algorithm and the dashed red line indicates the route the robot used as reference.

In the sequence, the first experiment shown in Figure 8 is detailed step-by-step, highlighting the events that occur from a snapshot to another. First, there are no obstacles in the environment, as illustrated in Figure 8a. So the LPA* algorithm finds the optimal path to the goal. Next, tree obstacles are inserted at the spots 4, 10 and 13, as stressed by the X-signal on the map of Figure 8b. Consequently, the algorithm looks for another feasible solution. It is worth mentioning that the route only changes if necessary. Note that obstacles 4 and 10 require a change of route, unlike obstacle 13. In other words, LPA* algorithm replans the route whenever an obstruction corrupts the previous computed route, which does not happen when obstacle 13 is placed in the environment. Once again, note that obstacle 13 does not obstruct the route, even the one created after placing obstacles 4 and 10. Following, another event occurs blocking region 14, so the robot can no longer follow its path. As described before, the algorithm is executed and finds another path. However, the robot is not on the new route and it is necessary to reach it. Thus, a "shortcut" is created connecting the robot's current position to its nearest point on the path. It is worth stressing that this shortcut is not always direct. Instead, the Dijkstra algorithm guarantees a collision-free path that fulfills such a constraint. Figure 8c illustrates this block, replan and shortcut situation. Finally, regions 17 and 24 receive the occupied tag, as shown in Figure 8d,e, respectively. Notice the LPA* runs over again and establishes the path to be followed until the robot reaches its goal, as shown Figure 8f.

A video for the experiment is available at: https://youtu.be/YU7IyGgXclA. Through video and Figure 8, it can be concluded that the tracking system accurately captured all obstacles and the algorithm, when perceiving an obstacle, quickly redesigned the route. Thus, the robot is able to achieve the goal.

5.4. Experiment 2: Impossible Route

In this experiment, a case was implemented for which obstacles isolate the target and the robot, and a message of impossible route is defined. Figure 9 illustrates the experiment, as well the video available at https://youtu.be/tRVyPzqXLRo.

Analogously to the first experiment, the second begins with an obstacle-free environment, see Figure 9a. Notice the path is identical to the one previously. The path changes just after setting zone 5 and 13 as locked ones. Notice in Figure 9b that obstacle 5 demands a path redesign, but obstacle 13 does not. In Figure 9c, note that the obstruction occurs in an area where the robot has already passed. In that case, zone 2. As expected, the route does not suffer any change, and the robot keeps moving forward. However, when zone 9 becomes occupied, as shown in Figure 9d, LTA* looks for a new feasible route. Nevertheless, the robot is not on this route, and so, the shortcut strategy takes care of finding a path to connect them. Finally, Figure 9e illustrates the robot following the path, just before receiving the Route Impossible alert. This event happens after zones 17 and 21 are labeled as not-safe to navigate, as shown in Figure 9f. In other words, the impossible route alert indicates there is no path connecting the starting and the destination zones. Then, the robot must stop or return to its home-position. In this paper, we opted for the latter approach, as highlighted in the experiment video.

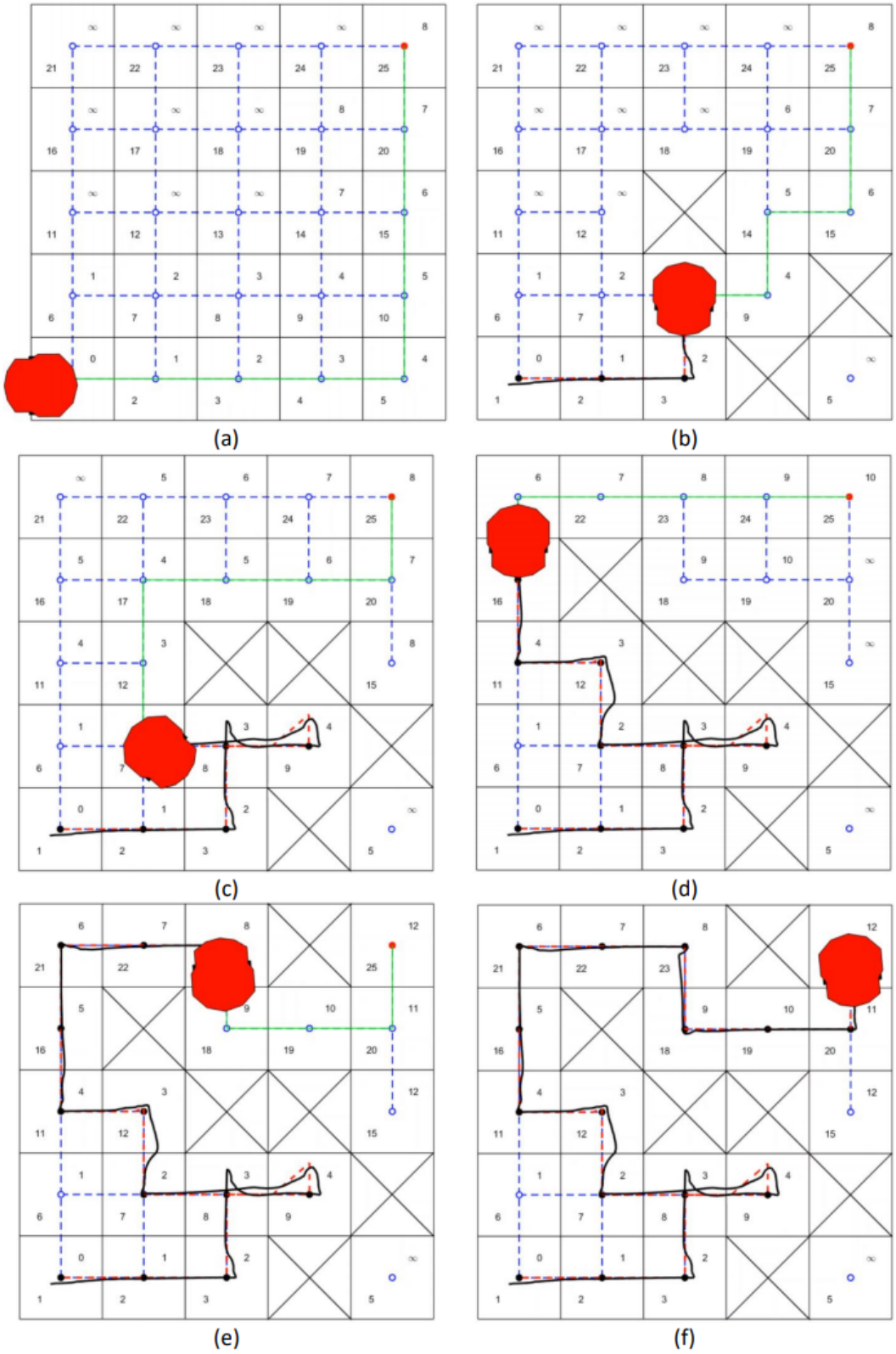


Figure 8. Experiment 1: The goal achievement. Snapshots of the route traveled, from the initial state (**a**) to the final one (**f**), passing by some intermediate stages (**b**–**e**).

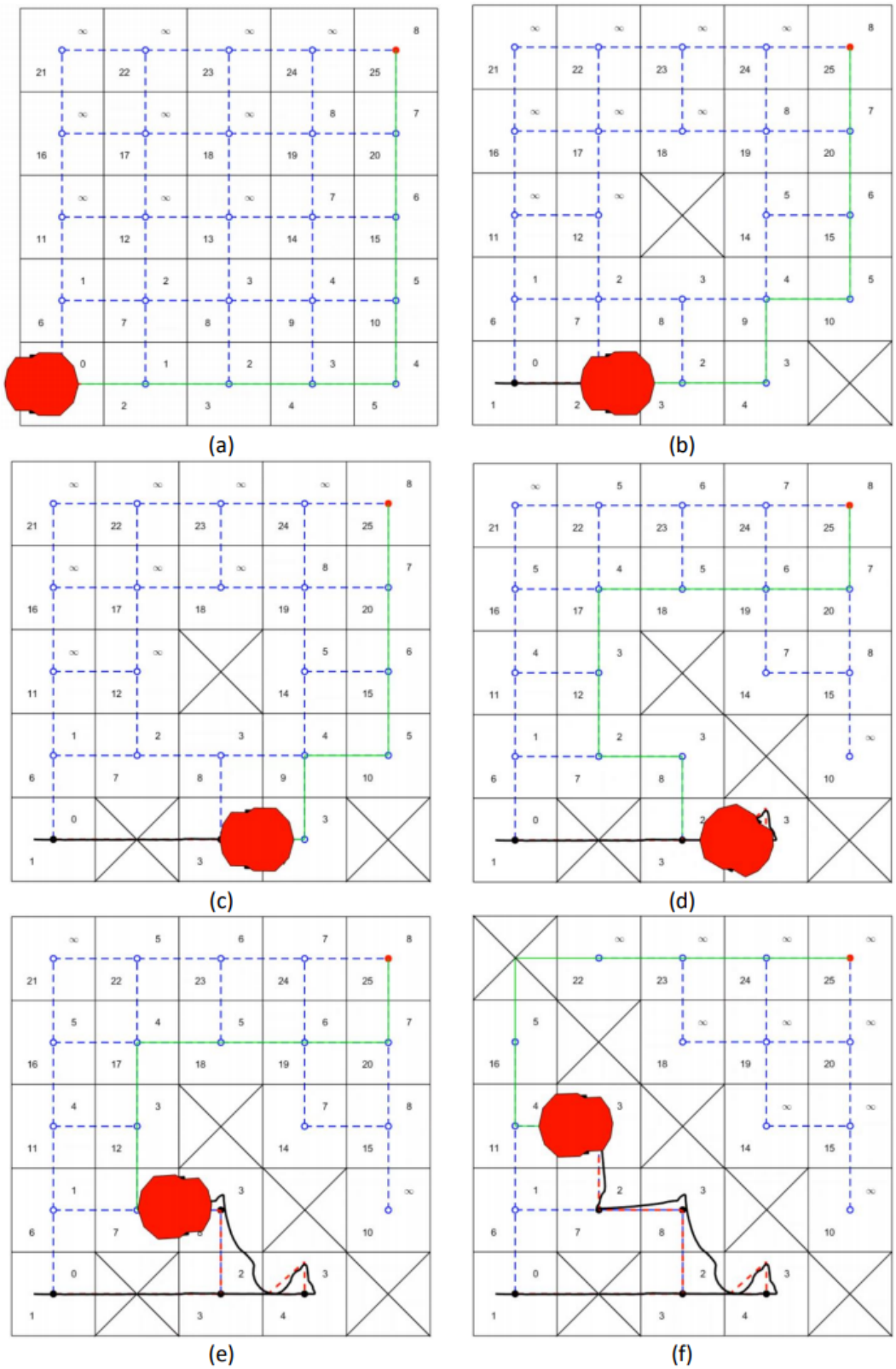


Figure 9. Experiment 2: The impossible route. Snapshots of the path traveled, from the initial state (**a**) to the final one (**f**), passing by some intermediate stages (**b**–**e**).

6. Concluding Remarks

This paper uses a combination of the Lifelong Planning A* (LPA*) and the Dijkstra algorithms to design a path planning strategy for a mobile robot in an uncertain environment. The LPA* replans the path when an obstacle appears on a previously planned path, while the Dijkstra creates a shortcut between the current robot position and the new planned path to make easier the transition of the robot to the updated path.

In simulations, the LPA* algorithm achieved the proposed objective, guiding the mobile robot in an environment that changes over time. Similarly, in real experiments the robot was, in fact, guided by the implemented algorithm, which interactively modified the planned path as unpredicted obstacles appeared. In summary, the proposal enables a path replanning and the seeking for an optimal solution that ensures the mission accomplishment, whenever a feasible route is available.

Finally, the proposed strategy is scalable and can be adapted, with some adjustments, for environments of any size. In short, the map would be subdivided into sectors containing smaller grids. At a high level, the algorithm would calculate a path that passes through sectors until it reaches its destination. At a low level, within each sector, another route would take the robot to the next sector. Nonetheless, this strategy is not addressed in this work, and stays as a suggestion for its continuity.

Author Contributions: Conceptualization, J.V.R.V., A.S.B. and M.S.-F.; methodology, J.V.R.V. and A.S.B.; software, J.V.R.V.; validation, J.V.R.V.; formal analysis, J.V.R.V.; investigation, J.V.R.V.; resources, J.V.R.V., A.S.B. and M.S.-F.; data curation, J.V.R.V.; writing—original draft preparation, J.V.R.V. and A.S.B.; writing—review and editing, J.V.R.V., A.S.B. and M.S.-F.; visualization, J.V.R.V. and A.S.B.; supervision, A.S.B. and M.S.-F.; project administration, A.S.B.; funding acquisition, A.S.B. and M.S.-F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CAPES—Coordenação de Aperfeiçoamento de Pessoal de Nível Superior grant number 88882.437236/2019-01.

Acknowledgments: The authors thank CNPq—Conselho Nacional de Desenvolvimento Científico e Tecnológico, an agency of the Brazilian Ministry of Science, Technology, Innovations and Communications, FAPES—Fundação de Amparo à Pesquisa e Inovação do Espírito Santo, and FAPEMIG—Fundação de Amparo à Pesquisa do Estado de Minas Gerais, for the support given to this research. Mr. Vasconcelos also thanks CAPES—Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, for the scholarship granted to him, which allowed him to dedicate all his time to the Master studies.

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

References

1. Zhang, H.Y.; Lin, W.M.; Chen, A.X. Path planning for the mobile robot: A review. *Symmetry* **2018**, *10*, 450. [CrossRef]
2. Ackerman, E. US Army Considers Replacing Thousands of Soldiers With Robots. IEEE Spectrum. 2014. Available online: https://spectrum.ieee.org/automaton/robotics/military-robots/army-considers-replacing-thousands-of-soldiers-with-robots (accessed on 30 October 2020).
3. Sapaty, P. Military robotics: Latest trends and spatial grasp solutions. *Int. J. Adv. Res. Artif. Intell.* **2015**, *4*, 9–18. [CrossRef]
4. Bhondve, T.B.; Satyanarayan, R.; Mukhedkar, M. Mobile rescue Robot for Human body detection in rescue operation of disaster. *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.* **2014**, *3*, 9876–9882.
5. Tuna, G.; Gungor, V.C.; Gulez, K. An autonomous wireless sensor network deployment system using mobile robots for human existence detection in case of disasters. *Ad Hoc Netw.* **2014**, *13*, 54–68. [CrossRef]
6. Narayanan, R.G.L.; Ibe, O.C. A joint network for disaster recovery and search and rescue operations. *Comput. Netw.* **2012**, *56*, 3347–3373. [CrossRef]
7. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
8. Holte, R.C.; Drummond, C.; Perez, M.B.; Zimmer, R.M.; MacDonald, A.J. Searching with abstractions: A unifying framework and new high-performance algorithm. In Proceedings of the Biennial Conference—Canadian Society for Computational Studies of Intelligence, Banff, AB, Canada, 16–20 May 1994; pp. 263–270. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.3023&rep=rep1&type=pdf (accessed on 22 September 2020).

9. Koenig, S.; Likhachev, M.; Furcy, D. Lifelong planning A*. *Artif. Intell.* **2004**, *155*, 93–146. [CrossRef]
10. Nasrinahar, A.; Chuah, J.H. Intelligent motion planning of a mobile robot with dynamic obstacle avoidance. *J. Veh. Routing Algorithms* **2018**, *1*, 89–104. [CrossRef]
11. De Oliveira, Á.V.; Fernandes, M.A. Dynamic planning navigation strategy for mobile terrestrial robots. *Robotica* **2016**, *34*, 568–583. [CrossRef]
12. Patle, B.; Parhi, D.R.; Jagadeesh, A.; Kashyap, S.K. On firefly algorithm: Optimization and application in mobile robot navigation. *World J. Eng.* **2017**. [CrossRef]
13. Patle, B.; Parhi, D.; Jagadeesh, A.; Kashyap, S.K. Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot. *Comput. Electr. Eng.* **2018**, *67*, 708–728. [CrossRef]
14. Han, J.; Park, H.; Seo, Y. Path planning for a mobile robot using ant colony optimization and the influence of critical obstacle. In Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management, Detroit, MI, USA, 23–25 September 2016.
15. De Oliveira, G.C.R.; de Carvalho, K.B.; Brandão, A.S. A hybrid path-planning strategy for mobile robots with limited sensor capabilities. *Sensors* **2019**, *19*, 1049. [CrossRef] [PubMed]
16. Gia Luan, P.; Thinh, N.T. Real-Time Hybrid Navigation System-Based Path Planning and Obstacle Avoidance for Mobile Robots. *Appl. Sci.* **2020**, *10*, 3355. [CrossRef]
17. Zhang, X.; Zhao, Y.; Deng, N.; Guo, K. Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 91. [CrossRef]