A New Distance Diffusion Algorithm for a Path-Planning Model based on Cellular Automata

1st Samuel C. S. Nametala
Faculty of Computing
Federal University of Uberlândia
Uberlândia, Brazil
samuel.nametala@gmail.com

2nd Luiz G. A. Martins
Faculty of Computing
Federal University of Uberlândia
Uberlândia, Brazil
lgamartins@ufu.br

3rd Gina M. B. Oliveira
Faculty of Computing
Federal University of Uberlândia
Uberlândia, Brazil
gina@ufu.br

***Abstract*—Cellular automata (CA) are bio-inspired approach that have been recently investigated to several applications including robotics. An improved model based on CA rules is proposed and evaluated for path-planning in autonomous robots. The objective is to build a short collision-free path from the robot's starting position to the target, trying to avoid unnecessary turns as much as possible. CA rules are used to enlarge obstacles, avoid their concavities and spread the distance from each cell to the target. The robot route is planned using the information of the distance of each free cell to the goal. Experiments were carried out to confirm the efficiency of the new techniques employed. Simulations with the navigation of a e-puck robot using the Webots platform have shown promising results confirming that the model is able to plan smooth and short routes.**

***Index Terms*—Cellular Automata, bio-inspired computing, distance diffusion, path-planning, autonomous robotics**

I. Introduction

In autonomous robotics, one of the most investigated problems is the path-planning task [1]. It aims to find the best trajectory between the robot position and the target, while avoiding collisions. Route-maps [2], cell decomposition [3] and potential field [4] are traditional approaches to deal with this problem. Nonetheless, bio-inspired techniques such as neural networks [5] and cellular automata [6]–[14] have also been successfully applied to the path-planning due to its decentralized structure and low computational cost. Such kind of approach could be applied, for example, for path planning of warehousing and logistics [15].

This work proposes to improve the path-planning model based on local CA rules for robotics introduced in [7] and refined in posterior models [12] and [13]. In the later references, the focus was on improving the navigation while the robot transverses the environment. On the other hand, here our emphasis is on the initial step of the model, when the route is built. Considering the path planning problem, two types of errors can hinder the successful execution of the task: systemic, more linked to the characteristics inherent to the robot (such as disproportionate wear of the wheels); and non-systemic, more related to the environment (such as slippery floors). Systemic errors are satisfactorily reduced with the use of odometry, while non-systemic errors are more difficult to deal with. The changes proposed in the previous models were more concerned with correcting the inaccuracies related to the employment of odometry during navigation. Our proposal is to minimize the occurrence of both error types by reducing the distance and the number of rotations on the planned route. Therefore, the improvements proposed here are orthogonal and complementary to those discussed in previous works.

The main objective is to reduce the traveled distance and the number of rotations of the planned route in a model based on cellular automata rules and distance diffusion. The major modifications proposed here refers to the route calculus and they can be summarized by: (i) to differentiate diagonal and cardinal moves by penalizing the first type when calculating the route; and (ii) to apply a new step of rules designed to avoid concave regions resultant of obstacles. Several experiments were performed showing that the routes planned using the new model tends to be shorter and with a smaller number of rotations than those calculated by previous models. Simulations performed in Webots platform show that an e-puck robot was able to successfully navigate from the starting point to the goal based on the route planned using the new model.

The remainder of this paper is organized as follows. Section II introduces CA basic concepts and highlights its use in robot path-planning and previous models [7], [12], [13]. Section III describes the proposed model and highlights the implemented changes to the previous models. Experiments and simulations using Webots platform are discussed in Section IV. The conclusion and future work are presented in Section V.

II. Cellular Automata in Path-Planning Task

Path-planning aims to calculate and optimize the route for the robot navigation through the environment from the it current position (origin) to the destination (target). Many research based on CA have been successfully applied to the robot path-planning problem, since they are discrete models and its unique decentralized architecture allows the generation of highly distributed solutions even in complex scenarios [6]–[14], [16]. Cellular automata have been studied as a possible model for several biological systems. They also have been investigated as a distributed strategy for multi-agent systems. CA are composed by simple components (the lattice of cells) and local interactions (the transition rule) [17]. Cellular space is a D-dimensional regular lattice with N cells, each one follows

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

the same connection pattern with its close neighbors. CA are mostly characterized by their transition rule, which determines the state of the cell i at time $t+1$ depending on its own state and the states of its neighborhood at time t [18]. Therefore, the cells interact with others in a local and synchronous way. CA-based models have been proposed for different robotics tasks [19]–[22]. In [11] a classification scheme was presented which divides the previous CA-based robotic models into six distinct approaches, where the employment of local transition rules to perform distance diffusion in path-planning is one of these approaches.

Distance Diffusion [7], [12], [13] consists to use a CA rule to calculate the distance between each free cell and the goal. In [7] the first model that uses distance diffusion was proposed. In this model, the planning algorithm is divided into phases. Initially, all obstacles are virtually enlarged to avoid collisions. Posteriorly, the distance of each free cell from the target is calculated using the CA transition rule. Based on the computed distance, the algorithm chooses the shortest route from the robot position to the goal. The models proposed in [12] and [13] are both based on [7]. However, these posterior models focus mainly on aspects of the robot navigation and they kept the routing calculus similar to [7]. The model in [12] focus on recalculating the route on-fly to compensate navigation errors while in [13] the focus are on odometry refinements and a procedure that tries to keep the robot in the center of the cell.

The environments for robots navigation are better modeled by two-dimensional CA, which commonly employ two types of neighborhood: von Neumann and Moore [23]. von Neumann neighborhood is formed by the central cell and its four neighbors in the cardinal directions, whereas the diagonal cells are also included in the Moore neighborhood, totalizing 9 cells. Based on previous works [7], [12], [13], we adopt the Moore neighborhood in our CA-based path-planning model.

III. Proposed Model

We propose a new path-planning model based on cellular automata rules, which is an improvement of the models discussed in [7], [12] and [13].

The environment is discretized in a 2D space, which represents the cellular automata lattice. Path-planning algorithm computes a route between an initial point (robot position) and the target cell (goal) using a distance diffusion approach. This diffusion is based on the application of local cellular automata rules over the lattice by some time steps. The robot is considered as a single non-oriented point not subjected to kinematics and dynamic laws. The algorithm receives a map of the environment based on a preprocessed image as input, which contains walls, obstacles, robot and goal positions. The lattice is divided into square cells in such way that the robot occupies an unique cell. Each cell state has seven possible values: free, initial, goal, wall, obstacle, virtual_wall and virtual_obstacle. Initially, only the first five states are possible describing the information captured by the image. The other two can be created by the algorithm when calculating the route as explained following.

Fig. 1 shows an overview of the proposed approach. An external device is responsible to capture the image of the environment, processing it and sending the corresponding map to the robot, which starts the path planning process. In the first step, based on the received map, internal obstacles and walls are enlarged using OE and WE rules, respectively. In the second step, the ECR rule is successively applied to shade the concave regions of obstacles, while there are neighborhoods of free cells corresponding to some triggering patterns. The distance to the goal is spread in step 3 using the GDLS and GDDS rules. In the fourth step, a free-collision route is obtained based on the GD values. Once the route is built, the path planning is finished and the robot controller is started, which will move the robot towards the goal. This process is repeated for each N robot moves (we use $N = 5$), by updating the route as the robot is moving toward the goal.

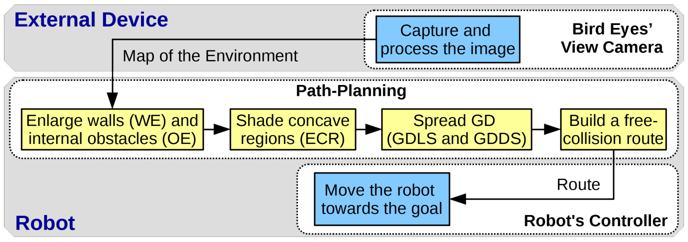

External Device
Capture and process the image
Bird Eyes' View Camera
Map of the Environment
Path-Planning
Enlarge walls (WE) and internal obstacles (OE)
Shade concave regions (ECR)
Spread GD (GDLS and GDDS)
Build a free-collision route
Route
Move the robot towards the goal
Robot
Robot's Controller

Fig. 1. Path planning flowchart.

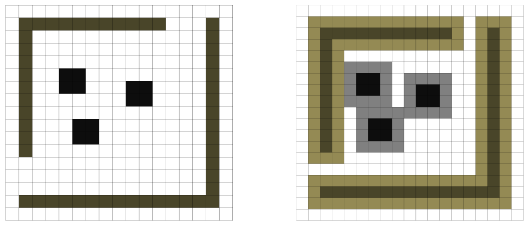
As highlighted in Fig. 1, the path-planning algorithm is divided into four main phases: (i) obstacles and walls enlargement; (ii) obstacles' concave regions shading; (iii) goal distance (GD) spreading; and (iv) route calculus based on the GD values for the free cells. They are detailed as follows.

The first phase provokes the enlargement of the walls and internal obstacles identified in the preprocessed image to compensate the effects of the dynamics not considered in the model and to avoid obstacle collisions during the robot navigation. In the present model, two similar CA rules are used to enlarge walls and internal obstacles, but keeping the differentiation between them. This phase was also presented in the previous models, however they are simpler than the one proposed here since they use the same state for walls and internal obstacles. The enlargement rules are defined by:

Obstacle Enlargement rule (OE): *IF* the central cell is free *AND* one of its neighbors is obstacle or virtual_obstacle, *THEN* the next value of the central cell is virtual_obstacle.

Wall Enlargement rule (WE): *IF* the central cell is free *AND* one of its neighbors is wall or virtual_wall, *THEN* the next value of the central cell is virtual_wall.

Both rules are applied for a fixed number of time steps (x), which depend on the cellular space resolution and robot size. In our experiments, we used $x = 1$ as used in [12], which results in an enlargement of one cell in obstacles and walls thickness. In a case of conflict, OE has more priority than WE (to be trigged). Fig. 2(a) presents the initial CA lattice resulting from the discretization of the environment, showing free (white), wall (green) and obstacle (black) cells. Fig. 2(b) shows them after the application of the rules, adding virtual_wall (green) and virtual_obstacle (dark grey) states.



(a) Initial lattice ($t = 0$) (b) Enlarged lattice ($t = 1$)

Fig. 2. Enlargement of walls and internal obstacles through the application of OE and WE rules by 1 time step: (a) original CA lattice ($t = 0$) and (b) CA lattice after the enlargement process ($t = 1$).

The second phase provokes an additional enlargement when one internal obstacle or a cluster of them defines a concave region which is not possible to be traversed by the robot to achieve the goal or there is risk of collision. These concave regions are shaded by CA rules (that is, their internal free cells are also transformed in virtual_obstacle states), since they represent an useless effort for the robots. Therefore, they can be avoided in the path planning. Besides, since the distance to the goal spreading is calculated just for free cells, it diminishes the computing in the next phase. An important consequence of this concave region shading is that the resultant number of rotations of the robots in the planned route tends to decay as shown in the experiments of Section IV. This phase is the major contribution of the present work and it was not presented in previous models discussed in [7], [12] and [13].

The CA transition rules that enlarge the obstacles in concave regions are trigged by eight possible neighborhood patterns for a free cell (Fig. 3) . Therefore, there are 8 CA rules that can be applied to each free cell in this phase. However, they are condensed in a totalistic description below:

Enlargement of Concave Region rule (ECR): *IF* the central cell and some one neighbor corner are free (F) *AND* the opposite neighbor corner with at least its three adjacent neighbors are obstacle or virtual_obstacle (O/V), as shown in one of the eight patterns presented in Fig. 3, *THEN* the next value of the central cell is virtual_obstacle.

ECR rule is consecutively applied for a non-fixed number of time steps, until no free cell in the lattice can be trigged by one of the 8 patterns. Fig. 4(a) shows the environment after one step application of the ECR shading rule over the free cells of the lattice presented in Fig. 2(b) (the final result of the first phase). The cells in light gray represents those changed to virtual_obstacle due to the application of ECR rule. Fig. 4(b) and Fig. 4(c) show the successive increment of new virtual obstacles in the environment after the application of the ECR rule by 2 and 3 times steps, respectively. Fig. 4(d) show the environment after the application of ECR by 7 time steps. This is the final configuration of the lattice, since no free cell will trigger ECR rule and this phase is finished.

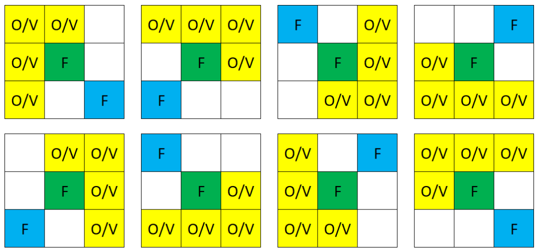
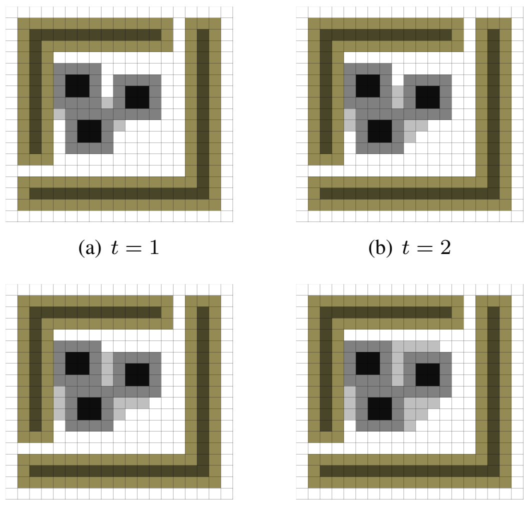


Fig. 3. Triggering patterns of the ECR rule: the central cell (green) must be free, one neighbor corner cell (blue) must be free and its opposite corner with at least three adjacent cells (yellow) must be obstacle or virtual_obstacle. The states of the remaining neighborhood cells (white) are not relevant to trigger the cells, that is, they can be free or any kind of obstacle or wall.



(a) $t = 1$ (b) $t = 2$

(c) $t = 3$ (d) $t = 7$

Fig. 4. Lattice after the application of ECR rule in different time steps (t). The colors represent free (white), wall (green), obstacle (black), virtual_wall (light green) and virtual_obstacle (gray) cells, where the enlargement resultant from ECR application (shading of concave regions) are in light gray.

Comparing the final enlarged lattice (Fig. 4(d)) with the initial one (Fig. 2(a)), it can be seen that the original three independent obstacles results in a compact region, which will be avoided in the route computation.

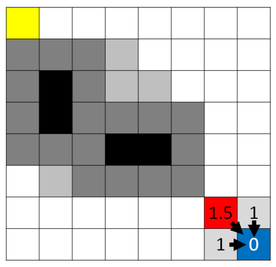
CA rules on the third phase spreads the distance to the target (called Goal Distance - GD) for each free cell. Free cells states change to numbers that quantify their distance to the goal cell (target). This is the major phase of the path-planning models proposed in [12] and [13]. However, in these previous models the distance in each move was considered equivalent to one robot step without distinction between lateral and diagonal moves, although the later takes longer distance. Here, our model considers the difference between they. It is considered that the distance is K_L when a robot makes a lateral move (from the central cell to a neighbor vertically or horizontally) and K_D when it makes a diagonal move (from the central cell to a neighbor on one of its diagonals), being K_L and K_D parameters of the algorithm. The process starts by setting the GD value of the goal cell to 0. The rules updates only free cells when at least one of its neighbors has its GD value changed in the last time step. Thus, they are applied for several time steps until no updating is possible. GD spreading rules are defined by:

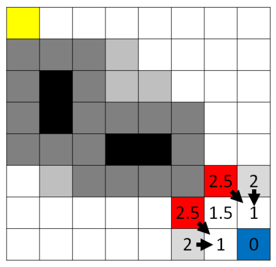
Goal Distance Lateral Spreading rule (GDLS): *IF* the central cell is free *AND* its neighbor with the lowest GD value v is in a vertical or horizontal position in relation to it, *THEN* the next value of the central cell is $v + K_L$.

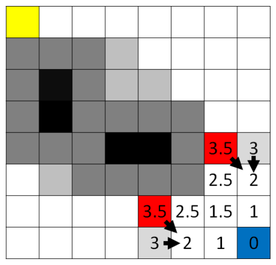
Goal Distance Diagonal Spreading rule (GDDS): *IF* the central cell is free *AND* its neighbor with the lowest GD value v is in a diagonal position in relation to it, *THEN* the next value of the central cell is $v + K_D$.

In the experiments here we considered $K_L = 1$ and $K_D = 1.5$ (to approximate $\sqrt{2}$). GD is calculated for all free cells by spreading the distance starting from the goal cell. GDLS and GDDS rules are applied for several time steps, while there is at least one free cell that triggers the rule or until the neighborhood of the robot's initial position is reached. In a case of conflict, GDLS has more priority than GDDS rule to be trigged. The neighbors of the robot position will be accessed if there is a collision-free route from the initial cell to the goal. However, the spreading could be prematurely stopped (before the initial cell is reached) when there is no possible route. The differentiation between moves proposed in the present work tends the robot to use lateral than diagonal moves, because the later are more expensive. It results in a final route with the shortest distance and smaller number of rotations as one could see in the experiments of Section IV.

Fig. 5 shows an example of goal distance spreading using GDLS and GDDS rules. The black arrows relate the cells being updated to their lowest-value neighbors. Starting from GD of the goal cell (in blue) equal to 0, Fig. 5(a) presents the values propagated to its neighbors after triggering GDLS and GDDS rules. The cells updated using GDDS are represented in red while the ones defined by GDLS are in light gray. For example, the neighbor with the lowest GD value for the red cell is 0 in an diagonal position. Therefore, GDDS was applied to this cell obtaining 1.5 ($v = 0 + K_D = 1.5$). For the gray cells, the neighbor with the lowest GD value is also 0, but they are in the vertical and horizontal positions. Thus, GDLS was applied obtaining 1 for both ($v = 0 + K_L = 1$). On the next time step (Fig. 5(b)) GD was propagated using the cells calculated in the previous step. One could see that free cells have their values calculated by applying GD rules: two gray cells (GDLS) with GD equal to 2 ($v = 1 + K_L = 1$)) and two red cells (GDDS) with GD equal to 2.5 ($v = 1 + K_D = 1.5$)). Fig. 5(c) highlights the GD updating for additional four cells: gray (GDLS) with value 3 ($v = 2 + K_L = 1$) and red (GDDS) with value 3.5 ($v = 2 + K_D = 1.5$). The GD spreading continues applying GDLS and GDDS rules until a neighbor of the robot position (in yellow) is reached: the cell with GD equal to 11. The final lattice with GD values after 10 time steps is shown in Fig. 5(d).


(a) $t = 1$


(b) $t = 2$


(c) $t = 3$

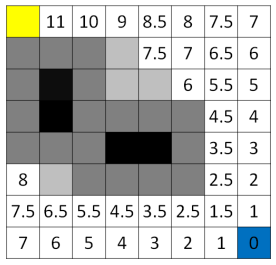
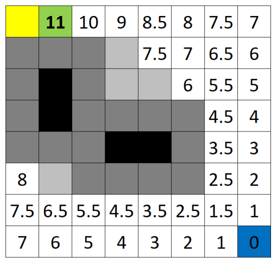

(d) $t = 10$

Fig. 5. Lattice after the application of GD rules in different time steps (t).

The forth and last phase uses GD values of the free cells to construct a free-collision route from the current robot position to the target cell. The cells are just chosen in a sequential way from the initial cell (in yellow) to the goal (in blue). Although it is very similar to this process in the previous models ([7], [12] and [13]), there is a slight modification in our algorithm. Previous models begins from the neighbor cell of the robot position and the algorithm just define the next cell to build the route by choosing one neighbor of the last chosen cell that decrements 1 in the value of GD at each step. Our algorithm needs to find and choose the neighbor with the lowest GD value and it can be decremented by 1 or 1.5. In a case of conflict (that is, two or more cells with the lowest GD value), the algorithm chooses the cell with the lowest euclidean distance to the goal. Fig. 6 shows the initial and final steps of the route building process, where the cells chosen to compose the robot's trajectory are in green.


(a) Initial step ($t = 1$)

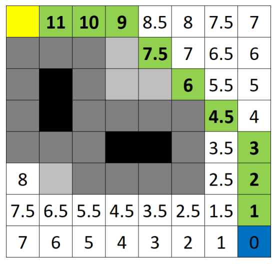
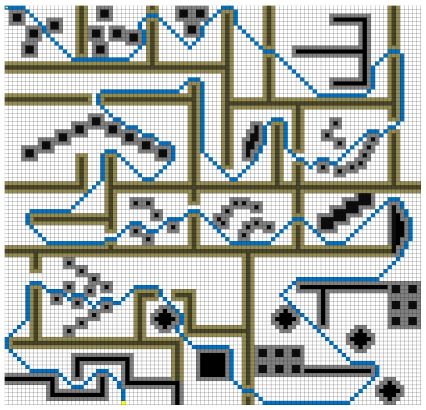

(b) Final step ($t = 10$)

Fig. 6. Steps of a robot's route building process.

The new combined techniques help the path-planning algorithm to find the shortest path with a smaller number of rotations, which consequently reduces the systemic and non-systemic errors during the robot navigation.

IV. Experiments and Results

We carried out some experiments in a computational environment implemented aiming to show the behavior of the proposed distance diffusion model and to confront this with the previous algorithm used in [7], [12] and [13]. For these simulations, we used two environments with 100×100 cells. Fig. 7 shows both environments after the enlargement of obstacles and walls. Each one has several rooms with walls, narrow doors and some internal obstacles.


(a) E1

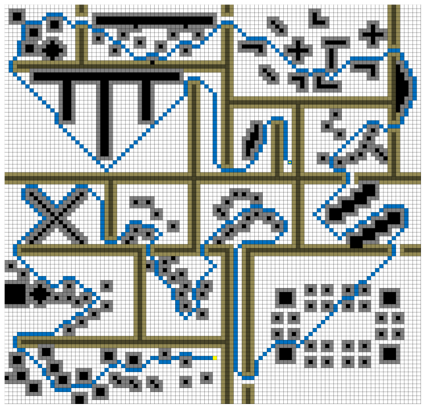
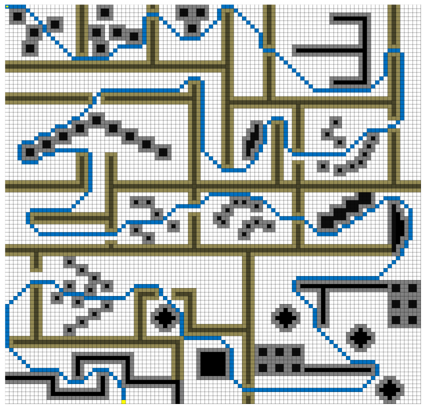

(b) E2

Fig. 7. Virtual environments used in the experiments. The routes in blue were calculated using the original path-planing discussed in [7], [12] and [13] (PATHP): (a) E1: distance = 738 and no. of rotations = 158; (b) E2: distance = 747 and no. of rotations = 211.

Aiming to clarify the contribution of each major modification of the investigated model, two versions of the proposed path-planning algorithm was elaborated. The first model differs between diagonal and lateral moves (GDLS and GDDS rules) but does not apply the shading of concave regions in the obstacles (ECR rule). It was called PATHP+D (path-planning with diagonal differentiation). The second one is a full version of the proposed model described in Section III. That is, in addition to using the GDLS and GDDS rules, it also shades the concave regions using the ECR rule. It was called PATHP+D+S (path-planning with diagonal differentiation and the shading of the obstacles' concave regions). The two versions were confronted with the results of the original algorithm for route calculus used in [7], [12] and [13] (called here PATHP) to verify the performance of the new model.

We run each version of the path-planning algorithm in both environments E1 and E2 by choosing appropriate initial position and goal cells that makes the robot to transverse all the rooms to find the target. Fig. 7(a) shows the route obtained for E1 using the original algorithm (PATHP): total distance = 738 and total number of rotations = 158. Fig. 8(a) and Fig. 8(b) present the routes calculated using PATHP+D and PATHP+D+S in the same environment, respectively. PATHP+D decreases the total distance from 738 to 702 and also reduces the number of rotations from 158 to 144 in E1. On the other hand, PATHP+D+S was able to achieve an extra reduction of rotations, totalizing 122 spins, with the same total distance of PATHP+D (702). Similar results were obtained in E2 with the calculated routes presented in Fig. 7(b), Fig. 9(a) and Fig. 9(b). PATHP+D decreases the total distance from 747 to 697 and also reduces the number of rotations from 211 to 150, while PATHP+D+S was able to achieve an extra reduction of rotations to 132.


(a) PATHP+D

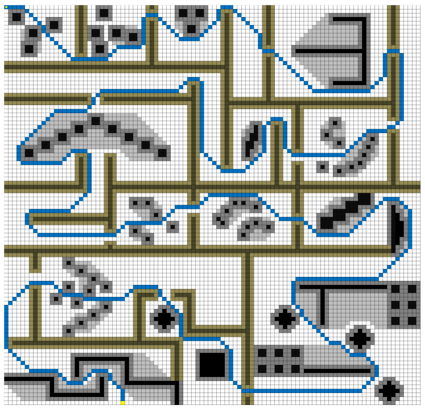
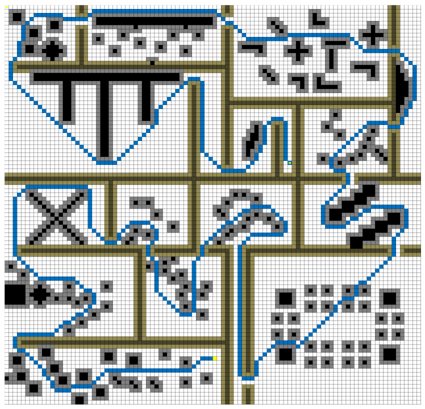
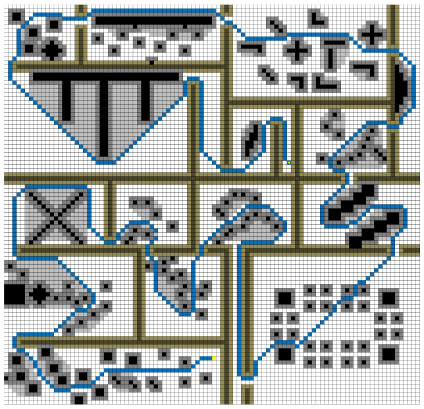

(b) PATHP+D+S

Fig. 8. Routes calculated for E1 using two versions of the proposed path-planing algorithm: (a) PATHP+D: distance = 702 and no. of rotations = 144; (b) PATHP+D+S: distance = 702 and no. of rotations = 122.



(a) PATHP+D

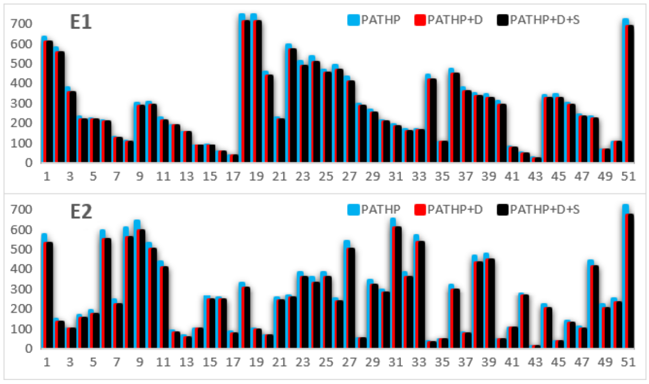


(b) PATHP+D+S

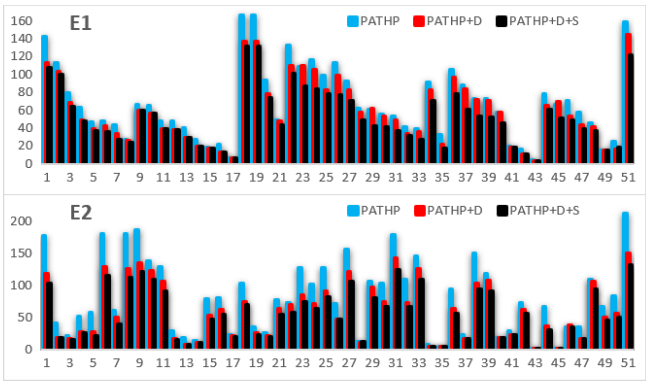
Fig. 9. Routes calculated for E2 using two versions of the proposed path-planing algorithm: (a) PATHP+D: distance = 697 and no. of rotations = 150; (b) PATHP+D+S: distance = 697 and no. of rotations = 132.

It highlights the skill of each modification in the proposed algorithm. The differentiation between diagonal and lateral moves enables find routes with shorter total distance and also reduces the number of rotations since it avoids diagonal moves. However, the shading of concave regions makes possible to reduce the number of rotations, returning smoothly robots' trajectories in the case that there are other routes with the same distance.

Subsequently, a series of 50 runs was performed for each environment, in which a pair of positions (initial, goal) is randomly generated. The objective is to quantify the proposed model improvements in terms of traveled distance and number of rotations. The results are presented individually in Fig. 10 while Fig. 11 shows the total results by summing the values of all the 50 runs for each environment. The full version of the model (PATHP+D+S) applied to E1 was able to reduce 4.3% of the total distance and 23.2% of the total number of rotations when compared with the original PATHP. For the environment E2, it was able to reduce 5.1% of the total distance and 25.7% of the total number of rotations. Comparing the versions of the proposed algorithm, the addition of concave regions shading returned the same distance in all the 100 runs and it achieves an extra reduction of 10,4% in E1 and 8,4% in E2. Also, the number of rotations in 73 runs decreased due to the concavity shading and in just 2 runs this number was slightly increased.

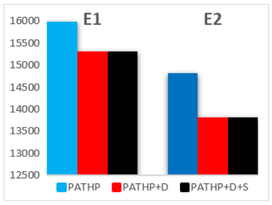


(a) Distance

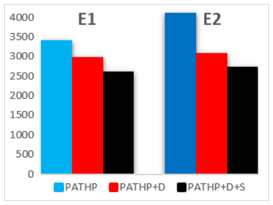


(b) Number of Rotations

Fig. 10. Performance per run of the investigated approaches using PATHP (blue), PATHP+D (red) and PATHP+D+S (black)



(a) Total Distance



(b) Total Rotations

Fig. 11. Total results for the investigated approaches using PATHP (blue), PATHP+D (red) and PATHP+D+S (black)

Fig. 12 shows in detail changes in the route planning when the PATHP+D or the PATHP+D+S algorithms are used instead of PATHP. The original route (PATHP) is shown in blue while the red route was built using PATHP+D and the black route was planned using PATHP+D+S. One can see that PATHP+D+S is able to build smoother routes by avoiding the obstacles concave regions. It is important to note that the shading step will turning free cells in virtual objects by applying the ECR rule step by step while there is a possible alternative path to cross the robot. For example, in Fig. 12(a) the concave region below the obstacles was covered towards the vertical enlarged wall until at least one free cell remain between the shadow area and the wall. This behavior is a consequence of the triggering patterns of the ECR rule showed in Fig. 3: the central cell will change to virtual obstacle only if the opposite corner is a free cell, ensuring the existence of at least one free path. On the other hand, Fig. 12(b) shows a situation in which the concave region was completely covered, since there is no other obstacle nearby. In this, the planned path is even more linear avoiding zigzag moves and reducing spins.

After the confirmation about the positive effects to build the planned route, we performed new simulations using Webots platform [24] to evaluate the navigation of the robot in executing this route. For this, we implemented the strategies proposed in [13] to navigate a single e-puck robot after the route planning. E-puch is a cylindrical compact mobile robot with $7cm$ of diameter [25]. In the navigation simulations, only the robot's position sensors (left and right wheel sensors) are used for the odometry computation. The resultant model was applied in a third environment, named E3, that is shown in Fig. 13. It has 70×45 cells and each cell is $7cm \times 7cm$. Therefore, each robot step takes $7cm$ if it is a lateral move and approximately $10cm$ if it is diagonal move.

Since the experiments here are simulated, a second e-puck robot was added to emulate the behavior of a device that captures and processes the image of the environment, sending the map to the navigating robot, to plan its route. Thus, in the emitting robot, the data after processing were added manually in a static manner, according to each environment. The locations of each obstacle, wall, robot and goal correspond to (x, y) positions of their respective cells in the environment map. A matrix stores the map, which is sent from the emitting robot. The connection between both robots is made with bluetooth inside the Webots platform [20].

Fig. 13 presents the planned routes calculated for E3 using the original PATHP and the proposed PATHP+D+S. It is possible to observe that the shading step together with diagonal differentiation generate a route that avoid to pass in the concavity regions formed by nearby obstacles returning a more linear trajectory, with less direction changes. Besides, the planned route using PATHP+D+S is shorter. Numerically, PATHP (Fig. 13(a)) calculated a route with total distance equal to 235 and number of rotations equal to 54, while PATHP+D+S (Fig. 13(b)) generated a route with total distance equal to 225 and number of rotations equal to 38.

Fig. 14 shows the actual trajectory of the robot in Webots when it navigates according to the planned route calculated using the proposed algorithm PATHP+D+S. It is possible to observe that the robot was able to achieve the goal cell starting from its initial position, making a collision-free trajectory very close to that planned. The video of this experiment can be found at [26]. We believe that the improvements in the route planning step mixed with the strategies proposed for robot control in [13] return an efficient model for path-planning and navigation of robots. The resultant path-planner is able to build short and smooth routes, whereas the control model makes the robot to execute its trajectory very close to the planned route.

The video of the simulation based on the route planned using PATHP can be found in [26] as well. The e-puck robot was also able to access the goal using a collision-free trajectory. However, the navigation using PATHP+D+S was almost 1 minute faster than using PATHP (14'15" against 15'05"), a consequence of its smoother route.

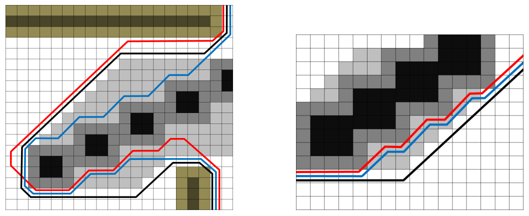
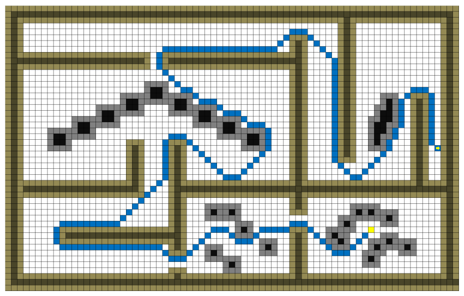

(a) (b)

Fig. 12. Details of two parts of the route calculated for environment E1 using PATHP (in blue), PATHP+D (in red) and PATHP+D+S (in black).


(a) PATHP

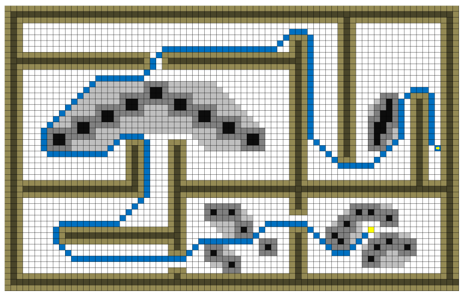

(b) PATHP+D+S

Fig. 13. Planned routes calculated for environment E3 using (a) PATHP: distance = 235 and no. of rotations = 54; (b) PATHP+D+S: distance = 225 and no. of rotations = 38.

V. Conclusion and Future Work

A new path-planning model based on cellular automata rules and distance diffusion is proposed for autonomous robotics. This model is an improvement of the models discussed in [7], [12] and [13]. The major modifications refers to the route calculus and they can be summarized by: (i) to differentiate diagonal and cardinal moves by penalizing the first type when calculating the route; (ii) to apply a new step of rules designed to avoid concave regions resultant of obstacles; and (iii) to differentiate walls and internal obstacles. The first modification causes generating routes that avoid diagonal moves when they are more expensive to the total displacement. The second makes the robots performing smoothly routes when

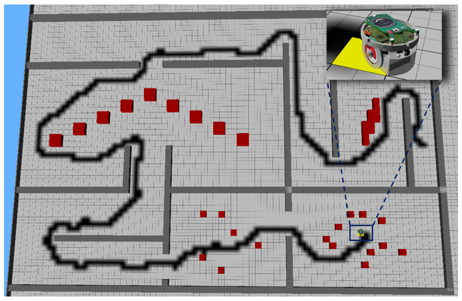


Fig. 14. Webots simulation showing the robot navigation on environment E3, using PATHP+D+S for route calculus. The corresponding video to this simulation is available at [26].

shading concave regions avoiding them. The third modification is needed to apply the shading rules for internal obstacles because in the previous models there is no difference between them and walls.

Our experiments have shown that the application of diagonal differentiation together with the shading rules return routes with short distance and smooth trajectories, that is, the robots make less rotations to traverse obstacle regions than the original routes planned in the previous models [7], [12] and [13]. Besides, Webots simulations have shown that the proposed model is able to calculate routes that can be successfully traversed using an e-puck robot and the navigation strategies presented in [13], indicating that this may be an efficient approach for robots planning and navigating.

Ongoing research aims to conduct experiments using real robots as performed in [13]. For this, we need to integrate our approach to a system for processing images captured from a bird eyes' view camera during navigation. The processed image generates a lattice map of the environment [13]. Comparative analysis with other path planning approaches, such as A* and D*-lite algorithms [27], will be carried out in future work. We also intend to investigate a hybrid path planning model integrating CA model and evolutionary robotics approach [28] aiming to refine several model parameters to give the robot more flexibility and robustness to face dynamic environments.

Acknowledgment

The authors thank FAPEMIG, CAPES and CNPq for the financial support.

References

[1] R. C. Arkin, "Behavior-based robotics," *MIT press*, 1998.

[2] Y. Zhang, N. Fattahi, and W. Li, "Probabilistic roadmap with self-learning for path planning of a mobile robot in a dynamic environment," in *Conf. Mechatronics and Automation*, 2013, pp. 1074–1079.

[3] C. Ramer, S. Reitelshofer, and J. Franke, "A robot motion planner for 6-dof industrial robots based on the cell decomposition of the workspace," in *Symposium on Robotics*, 2013, pp. 1–4.

[4] Y. Jianjun, D. Hongwei, W. Guanwei, and Z. Lu, "Research about local path planning of moving robot based on improved artificial potential field," in *Chinese Control and Decision Conf.*, 2013, pp. 2861–2865.

[5] C. Luo, J. Gao, Y. L. Murphey, and G. E. Jan, "A computationally efficient neural dynamics approach to trajectory planning of an intelligent vehicle," in *Int. Joint Conf. on Neural Networks*, 2014, pp. 934–939.

[6] P. G. Tzionas, A. Thanailakis, and P. G. Tsalides, "Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata," *Trans. Robotics and Automation*, vol. 13, no. 2, pp. 237–250, 1997.

[7] C. Behring, M. Bracho, M. Castro, and J. Moreno, "An algorithm for robot path planning with cellular automata," *Int. Conf. on Cellular Automata for Research and Industry*, pp. 11–19, 2000.

[8] F. M. Marchese, "A directional diffusion algorithm on cellular automata for robot path-planning." *Future Generation Computer Systems*, vol. 18, no. 7, pp. 983–994, 2002.

[9] K. Ioannidis, G. C. Sirakoulis, and I. Andreadis, "A cellular automaton collision-free path planner suitable for cooperative robots," *Panhellenic Conf. on Informatics*, pp. 256–260, 2008.

[10] A. Akbarimajd and A. Hassanzadeh, "A novel cellular automata based real time path planning method for mobile robots," *J. of Engineering Research and Applications*, vol. 1, no. 4, pp. 1262–1267, 2011.

[11] G. B. S. Ferreira, P. A. Vargas, and G. M. B. Oliveira, "An improved cellular automata-based model for robot path-planning," *Conf. on Towards Autonomous Robotic Systems*, pp. 25–36, 2014.

[12] G. M. B. Oliveira, P. A. Vargas, and G. B. S. Ferreira, "Investigating a cellular automata model that performs three distance diffusion on a robot path planning," *European Conference on Artificial Life*, pp. 271–278, 2015.

[13] L. G. Martins, R. d. P. Cândido, M. C. Escarpinati, P. A. Vargas, and G. M. Oliveira, "An improved robot path planning model using cellular automata," in *Conf. on Towards Autonomous Robotic Systems*, 2018, pp. 183–194.

[14] G. M. B. Oliveira, R. G. Silva, G. B. Ferreira, M. S. Couceiro, L. R. Do Amaral, P. A. Vargas, and L. G. A. Martins, "A cellular automata-based path-planning for a cooperative and decentralized team of robots," in *IEEE Congress on Evolutionary Computation*, 2019, pp. 739–746.

[15] K. C. T. Vivaldini, J. P. M. Galdames, T. B. Pasqual, R. M. Sobral, R. C. Araújo, M. Becker, and G. Caurin, "Automatic routing system for intelligent warehouses," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2010, pp. 1–6.

[16] L. E. Parker, B. Birch, and C. Reardon, "Indoor target intercept using an acoustic sensor network and dual wavefrontpath planning," *IEEE Int. Symposium on Intelligent Robots and Systems*, pp. 278–283, 2003.

[17] G. Oliveira, P. De Oliveira, and N. Omar, "Improving genetic search for one-dimensional cellular automata using heuristics related to their dynamic behavior forecast," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, vol. 1. IEEE, 2001, pp. 348–355.

[18] M. Mitchell, "Computation in cellular automata: A selected review," *Non-standard Computation*, pp. 385–390, 1996.

[19] D. A. Lima, C. R. Tinoco, and G. M. Oliveira, "A cellular automata model with repulsive pheromone for swarm robotics in surveillance," in *Int. Conference on Cellular Automata*. Springer, 2016, pp. 312–322.

[20] C. R. Tinoco, D. A. Lima, and G. M. Oliveira, "An improved model for swarm robotics in surveillance based on cellular automata and repulsive pheromone with discrete diffusion," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 1, pp. 53–77, 2019.

[21] D. A. Lima and G. M. B. Oliveira, "A probabilistic cellular automata ant memory model for a swarm of foraging robots," *14th Int. Conf. on Control, Automation, Robotics and Vision*, pp. 1–6, 2016.

[22] C. R. Tinoco and G. M. B. Oliveira, "Heterogeneous teams of robots using a coordinating model for surveillance task based on cellular automata and repulsive pheromone," in *IEEE Congress on Evolutionary Computation*, 2019, pp. 747–754.

[23] P. Sarkar, "A brief history of cellular automata," *ACM Computing Surveys*, vol. 32, no. 1, pp. 80–107, 2000.

[24] Cyberbotics, "Webots simulator," 2017, https://www.cyberbotics.com.

[25] "E-puck robot," 2017, http://www.e-puck.org.

[26] "Video: Bio-inspired computing lab," 2020, https://www.youtube.com/channel/UC4uDX-7nXDGYl4Hu5IVCJdw?disablepolymer=true.

[27] Y. D. Setiawan, P. S. Pratama, S. K. Jeong, V. H. Duy, and S. B. Kim, "Experimental comparison of a* and d* lite path planning algorithms for differential drive automated guided vehicle," in *Recent Advances in Electrical Engineering and Related Sciences*, 2014, pp. 555–564.

[28] G. Oliveira, R. Silva, L. Amaral, and L. Martins, "An evolutionary-cooperative model based on cellular automata and genetic algorithms for the navigation of robots under formation control," in *7th Brazilian Conf. on Intelligent Systems*, 2018, pp. 426–431.