IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 3, 2020, accepted November 26, 2020, date of publication December 1, 2020,
date of current version December 10, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3041802*

# A Supervisory-Based Collaborative Obstacle-Guided Path Refinement Algorithm for Path Planning in Wide Terrains

**MOHAMED G. B. ATIA[1], (Graduate Student Member, IEEE), HAITHAM EL-HUSSIENY[2,4], AND OMAR SALAH[3]**

[1]Department of Mechatronics and Robotics Engineering, Egypt-Japan University of Science and Technology, Alexandria 21934, Egypt
[2]School of Science, Engineering, and Environment, University of Salford, Salford M5 4WT, U.K.
[3]Department of Mechanical Engineering, Faculty of Engineering, Assiut University, Asyut 71516, Egypt
[4](*On leave*) Department of Electrical Engineering, Faculty of Engineering (Shoubra), Benha University, Cairo 11239, Egypt

Corresponding author: Omar Salah (omar.salah@aun.edu.eg)

**ABSTRACT** Robotic exploration of wide terrains, such as agricultural fields, could be challenging while considering the limited robot's capabilities in terms of sensing and power. Thus, in this article, we proposed OGPR*, an Obstacle Guided Path Refinement algorithm for quickly planning collision-free paths utilizing the obstacles existing in the environment. To tackle the issue of exploring wide terrains, a supervisory-based collaboration between the quadcopter and a mobile robot is proposed. The quadcopter is responsible for streaming subsequently live two-dimensional images for the environment under discussion while planning safe paths for the ground the mobile robot is planning safe paths to manoeuvre. Numerical simulations proved the significant performance of the proposed OGBR* algorithm when compared to the state of the art algorithms exist in the literature.

**INDEX TERMS** OGPR, path planning, tracking, mobile robot, collaborative.

## I. INTRODUCTION

Planning feasible paths for robots deployed into working environments is crucial in many different applications, ranging from the exploration of unknown environments [1], [2], navigation [3], [4], surveillance [5], [6], or cleaning [7], [8]. The aim is to find as optimal as possible an obstacle-free path starting from a certain point in the environment, usually the current robot position, and towards a predefined target point. Thus, the working environment is usually assumed to be known, or at least partially known beforehand via an exploration step conducted before the path planning step or through a given map for the environment under discussion. The performance of any path planning approaches is commonly assessed against one of the evaluation metrics, e.g. the time that is taken by the algorithm to find the feasible path, the length of the obtained path, and the time that will be taken by the robot to travel from the starting point to the target point [9]. In wide terrains, such as agricultural fields, the involvement of a team of robots to achieve the required task is often recommended over single robot systems [10], [11]. This is due to many reasons: first, a team of robots can fulfil a single task in less time when compared to that of a single robot [12], [13]. Furthermore, using several robots introduces a share of capabilities where each robot can do a specific navigation task depending on its resources, and in the end, all these achievements will be combined [14]. Meanwhile, teams of robots, therefore, tackle the known single point of failure issue and can be more fault-tolerant than a single robot. Thus, addressing the problem of collaborative path planning is promising, where a team of more than a single robot is collaborating in finding feasible obstacle-free paths in wide terrains. Path planning approaches are divided into two categories [15], traditional methods and soft computing methods. The traditional methods do not demonstrate intelligence in the path planning technique, unlike the soft computing methods that do. The output path of traditional methods is deterministic and it requires exact input data and is called on several types of algorithms like the node-based algorithms and sampling-based algorithms [16], [17]. Some examples of the node-based algorithms are, Dijkstra's algorithm [18]–[21], A* [22]–[24], LPA* [25], Theta* [26], [27] and

The associate editor coordinating the review of this manuscript and approving it for publication was Saeid Nahavandi.

D* [28] which split the environment into several nodes. Moreover, the sampling-based algorithms have many examples like RRT [29]–[31], RRT* [32]–[34], Artificial Potential Field [35], [36] which has the disadvantages of high complexity and local minima, and Roadmap algorithms [37]–[39]. The completeness is one of the advantages of these traditional methods in addition to that increasing the problem size increases the execution time. On the other hand, soft computing methods have many examples like particle swarm optimization algorithm [31], [40], [41] which is ten times faster than the Fuzzy logic [42]–[44], Genetic algorithm [45]–[47] which is used to avoid the local minima and the high execution time problems, Memetic algorithm [48], neural networks [49]–[51] and Ant colony [52], [53]. Although the solutions of the soft computing methods suffer from the unpredictability, the uncertainty and near-optimal path instead of exact solutions, these methods can be used in environments that higher artificial intelligence is needed and not enough information and noisy information is provided.

Although the advance of the mentioned path planning techniques, the issue of collaborative path planning for a team of robots to explore a wide terrain still needs to be addressed. Thus, in this article, we proposed a Collaborative Obstacle-Guided Path Refinement OGPR* algorithm for planning collision-free paths in wide terrains. A quadcopter is streaming multiple images for the area under investigation and the proposed OGBR* algorithm plan safe paths for a ground mobile robot to navigate this environment. One demand application of this algorithm is the inspection of agricultural fields, where the area is significantly wide for a single mobile robot to navigate. Meanwhile, the ground robot could be responsible for picking fruits, an inspection of flowers, etc.

The proposed OGPR* is an enhancement of our previous OGPR [54] algorithm was the collaboration between two robots in planning safe paths is considered. The key contributions of OGPR* are listed as follows:

- Online path planning technique where from live stream images a real-time computation is done to find the feasible obstacle-free paths.
- Optimization-based based on Particle Swarm Optimization (PSO) is utilized to quickly find feasible paths while considering obstacles in the environment into account.
- A cooperative approach in planning between aerial and ground robots.

The proposed OGBR* algorithm in this study is described in Section II. The performance evaluation is explained in detail in Section III while numerical simulations are achieved in Section IV. The results and their discussion are presented in Section V. Finally, the conclusions from this research are summarized in Section VI.

## II. MATERIALS AND METHODS

### A. PROBLEM STATEMENT

In this article, we have developed OGPR* algorithm which is an extension and improvement of the OGPR algorithm [55] to generate an optimized short path between two points in remarkably shorter time compared to the A* [56], [57] algorithm as discussed in the results.

OGPR* algorithm has several advantages over the standard OGPR algorithm as follows; OGPR algorithm requires the previously known equations of the shapes of the obstacles, OPGR* algorithm is simply developed without previous knowledge of the shapes of the obstacles to developing their equations. Moreover, one of the best advantages of OGPR* algorithm is that its input is the image and the orientation of the two points only while the OGPR algorithm requires a previous shape recognition step for the knowledge of the different shapes of the obstacles of the environment and their sizes and parameters which would make the algorithm more complicated unlike the simple OGPR* algorithm. Also, OGPR algorithm requires several complex mathematical operations unlike the OGPR* algorithm is very simple as it will be described in the next sections.

On the other hand, OGPR and OGPR* algorithms have the same concept and use the swarm optimization step, only the difference between them is the method of applying the concept.

The OGPR method [54], [55] has a mathematical strategy to generate an optimized path between two points. It consists of two stages: the first stage is the pre-optimization stage, in which an initial path is developed between the start and the target points guided with the edges of the obstacles. To develop this pre-optimization path, a straight line is developed based on the Heaviside unit step [58] between the start and target points. further, a mathematical equation is developed based on the Heaviside step function to describe the complete shape of obstacles in the environment. Then, intersection points between the basic line and the obstacles equations are calculated to define the active obstacles that the path collide with them. By combining the parts of the basic line outside the obstacles and one side of the edges each active obstacle, several pre-optimization paths are obtained.

In the second stage, the pre-optimization path is refined using particle swarm optimization to make the path shorter in several iterations each which replaces the longest possible part of the path by collision-free straight lines. In other words, the cost function searches for two points on the path that have the longest length between them, and the constrain function is a condition to test if the straight line between these two points is not intersecting any obstacles.

The OGPR method is based on a mathematical representation of the system that makes it fast [54], [55] to find the optimized path compared with the state of the art A* algorithm. On the other hand, the OGPR requires the shape types of each obstacle and its parameters (e.g. for the circle shape, it needs the diameter and the centre point) that puts some limitations on the method and may require a previous shape recognition step to define the obstacle shapes, their parameters and their locations in the environment that would increase the complexity of the method. Another limitation of the OGPR, despite its simple concept, is its complexity of the

mathematical representation that makes it more complex to be programmed.

All these drawbacks in the OGPR method have motivated the authors to use develop the OGPR* method for better representation and to be used in different applications special in online path planning detection. The OGPR* method is very similar to the concept of the OGPR method but it has been applied in matrices to form an image to simplify the method, have deeper investigations of the capabilities of the concept, and also to eliminate the previous limitations of the OGPR method. Similar to the OGPR method, the OGPR* consists of two stages; the pre-optimization stage and the optimization stage respectively. these stages are discussed in detail in the following subsections.

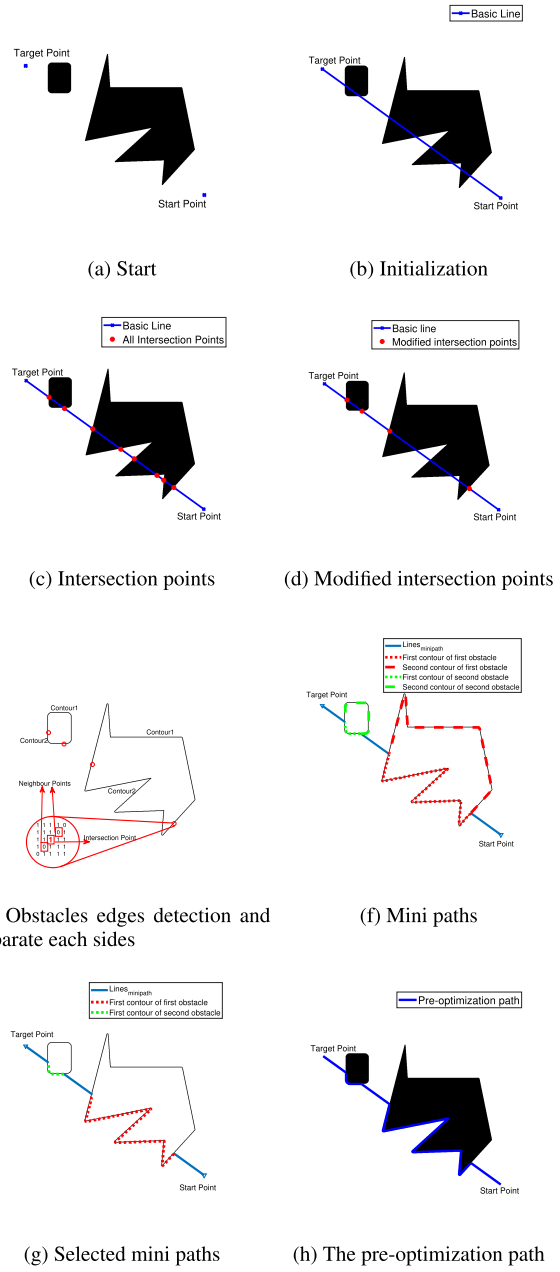
### B. THE PRE-OPTIMIZATION STAGE

In the pre-optimization stage, the initial collision-free path is generated depending on combinations of mini paths that come from the structure of the obstacles and the straight line between the start and target points. In the following, the algorithm of the initial collision-free path generation is described in detail:

- Step 1 (Initialization):
  The inputs to the algorithm, as described in $L1-L2$ (line 1 and line 2) of Algorithm 1, is the start point, target point, and a binary image as shown in Fig. 1(a). After that in $L3$, edge detection of the obstacles is processed. Then, in $L4$, the basic line is developed which is a straight line between the start and target pixels, as shown in Fig. 1(b), Further, the abscissa and ordinate of the pixels of the basic line are the vectors $\{X_{BasicLine}, Y_{BasicLine}\}$ respectively.
- Step 2 (Intersection points):
  Since $\{X_{BasicLine}, Y_{BasicLine}\}$ pixels store the state of the free space or the obstacles, the toggle of the states reveals the intersection points between the basic line and the obstacle. In case there is no obstacles lie on the basic line between the start and target points, the OGPR path is the basic line and the algorithm ends as described in $L5-L7$. Further, the basic line intersects each obstacle in two intersection points to divide it into two paths clockwise and counterclockwise directions. But, in the case of the concave-shape obstacles, more than two intersection points may result for this obstacle as shown in Fig. 1(c). So, the dummy intersection points between the two essential intersection points are eliminated, in $L9$, as the intersection points that fall on the edge pixels of the same obstacle and they are not the first neither the last intersection point determined as shown in Fig. 1(d).
- Step 3 (Generation of the mini paths):
  After the intersection points are obtained, their states are toggled, which divides the obstacle edge to two mini paths as shown in Fig. 1(e). These two mini paths represent the possibilities for the clockwise and counterclockwise directions to avoid the obstacle paths and generated by tracing the two neighbours of the toggled intersection point that fall on the obstacle edge as described in $L10-16$.
- Step 4 (Assembly of the pre-optimization path):
  The pre-optimization path is a combination of mini paths generated from the obstacles edges and straight lines between the intersection points to connect these mini paths. So, the obtained path is assembled by first adding the straight line between the start point and the closest intersection point on the first obstacle. Then, the mini path of the first obstacle. After that, the straight line between the further intersection of the first obstacle and the closest intersection point on the following mini path and so on as described in $L17-18$. Finally, a straight line between the further intersection point on the final obstacle and the target is added to the pre-optimization path in $L20-21$ as shown in Fig. 1(f-g). Although the

**Algorithm 1** The Pre-Optimization Step Algorithm

1: **Input**: Start point, target point, binary image of environment
2: $O$ = binary image of environment
3: $B$ = Edge detection of $O$
4: $\{X_{BasicLine}, Y_{BasicLine}\}$ = Basic Line pixels between start and target points
5: $\{X_{Intersection}, Y_{Intersection}\}$ = end points of $\{X_{BasicLine}, Y_{BasicLine}\}$ in $O$
6: **if** $X_{Intersection}$ is empty **then**
7: OGPR* path = $\{X_{BasicLine}, Y_{BasicLine}\}$
8: **else**
9: Remove in-between points on the same edge of $\{X_{Intersection}, Y_{Intersection}\}$
10: $\{X_{Intersection}, Y_{Intersection}\}$ remove from $B$
Pre-optimization
11: path = Start point
12: **for** i := 1 **to** size of $X_{Intersection}$ **step** 2 **do**
13: Find two neighbours to $\{X_{Intersection_i}, Y_{Intersection_i}\}$ that fall on each edge
14: contour1 = Trace edge of a neighbour in B
15: contour2 = Trace edge of the other neighbour in B
16: contour = min(contour1, contour2)
17: $Line_{minipath}$ = straight line between $\{X_{Intersection_{i-1}}, Y_{Intersection_{i-1}}\}$ and $\{X_{Intersection_i}, Y_{Intersection_i}\}$ points
Pre-optimization
18: path = [ Pre-optimization path; $Line_{minipath}$; contour ]
19: **end for**
20: $Line_{minipath}$ = straight line between $\{X_{Intersection_i}, Y_{Intersection_i}\}$ and Target
Pre-optimization
21: path = [ Pre-optimization path; $Line_{minipath}$]
22: **end if**
23: **Output**: Pre-optimization path




(a) Start (b) Initialization

(c) Intersection points (d) Modified intersection points

(e) Obstacles edges detection and separate each sides (f) Mini paths

(g) Selected mini paths (h) The pre-optimization path

**FIGURE 1.** OGPR* Pre-optimization step.

OGPR* selects the shortest mini path of the two mini paths on the obstacle in $L16$ to develop the shortest pre-optimization path, this selection is critical, and the shortest pre-optimization path does not guarantee to develop the shortest optimized path. So, in situations where the time is not critical, it is worth to investigate not only the shortest pre-optimization path but a number of the pre-optimization paths based on the possible directions of a mini path to select the shortest optimized path. to solve this issue, A sampling factor has been developed and discussed in the sections. which represents the ratio of the number of the pre-optimization paths developed from the algorithm and needed to be investigated to the total number of the pre-optimized paths.

Fig. 1 reveals an example of the OGPR* algorithm working steps to develop the pre-optimization path.

### C. THE OPTIMIZATION PROCESS

The initial optimized path is equal to the pre-optimization path which can be represented by a two-column matrix containing all its points, as shown in Eq. 1 where $n$ is the number of the path points.

$$\text{Initial optimized path} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_n & Y_n \end{bmatrix} \tag{1}$$

The path before being fully optimized, have many large portions of the path that can be replaced by a straight line to make it much shorter. It is assumed that $Q$ and $W$ are the indexes of the first point and endpoint in the path matrix that the portion fall in between, and it can be replaced by a free-collision straight line as shown in Eq. 2.

$$\text{Current optimized path} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_Q & Y_Q \\ X_{Q+1} & Y_{Q+1} \\ \vdots & \vdots \\ X_{W-1} & Y_{W-1} \\ X_W & Y_W \\ \vdots & \vdots \\ X_n & Y_n \end{bmatrix} \tag{2}$$

So, the optimization stage replaces the points between the index $Q$ to the index $W$ by the points of a free-collision straight line between these two points. The cost function is shown in Eq. 3:

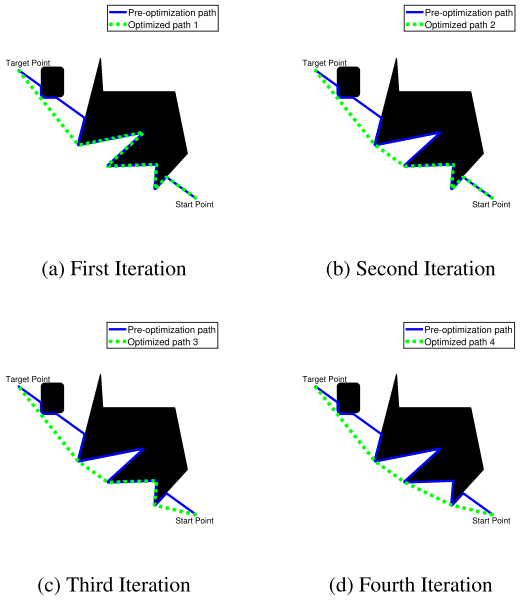
$$\text{Cost function} = Q - W \tag{3}$$

Furthermore, there are two constraints of the straight line between the two indexes $Q$ and $W$ that the line does not intersect any of the obstacles and it has not been chosen in the previous optimization loops. The constraints are be represented by the logical function $M$, if at least one of the condition is satisfied then the output is logical 1 which means that the selected $Q$ and $W$ points are unsuitable and otherwise, it is logical 0 and the line is suitable. The penalty function is expressed by Eq. 4 which converts the constrained function to unconstrained one [59]:

$$\text{Penalty function} = Q - W + k * M \tag{4}$$

where $k$ is the penalty factor that has a large value so that it takes effect when the $M$ is satisfied and makes the penalty function value larger and thus the optimization algorithm avoids this state.

Finally, the optimization stage uses particle swarm optimization algorithm [60]–[62] to search for the best values of the penalty function Eq. 4 and the optimization stage keeps working in a loop until it produces the most optimized path.




(a) First Iteration (b) Second Iteration

(c) Third Iteration (d) Fourth Iteration

**FIGURE 2.** OGPR* optimization iterations.

Fig. 2 illustrates an example of the iterations of the OGPR* algorithm to develop the optimized path. In the first iteration, the largest portion is close to the target point and is replaced by a free-collision straight line. Then, the second iteration replaces the largest portion which is in the middle by another straight line. After that, in the third iteration, the largest portion happens to be close to the start point and is replaced by a straight line. Finally, the path is optimized by replacing the largest portion by a straight line and there are no remaining portions in the path that can be replaced by a free-collision straight line.

The optimization step is described in the next algorithm:

**Algorithm 2** The Optimization Step Algorithm

1: **Input**: Pre-optimization path, binary image of environment
2: $O$ = binary image of environment
3: **Initialize**: Optimized path = Pre-optimization path
4: **Initialize**: Lines = [0 0]
5: **Initialize**: Q = W = 1
6: **while** Value is negative **do**
7: Optimized path (Q) to Optimized path (W) = straight line
8: Q, W = Swarm Optimization of the objective in Eq. 4
9: Value = value of Eq. 4 at the same Q, W
10: Lines = [Lines; Q W]
11: **end while**
12: **Output**: Optimized path

## III. EVALUATION OF OGPR* ALGORITHM

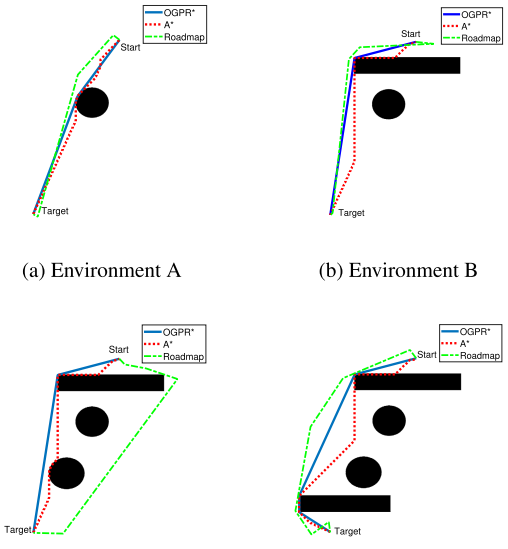
OGPR* algorithm is developed to generate the shortest path between two points. It chooses the shortest pre-optimization path as an initial path for the optimization but it is not necessarily the shortest pre-optimization path should generate the shortest possible path especially in complex environments. Also, optimization of all pre-optimization paths and selecting the shortest optimized path between them can take much more time.

The number of the pre-optimization paths is $2^n$ as $n$ is the number of the obstacles, that is why, we developed OGPR* selection ratio factor which represents the number of pre-optimization paths used to generate the shortest optimized path and falls in the range of $[1, 2^n]$.

In this section, the effects of (the size and complexity of the environment, and OGPR* selection ratio) factors are discussed on OGPR* algorithm with comparison to A* algorithm and the probabilistic roadmap [63] in terms of time execution and path length.

### A. THE SIZE AND COMPLEXITY OF THE ENVIRONMENT

The size of the environment and its complexity surely affect the performance of path planning algorithms as increasing them leads to increasing the time cost [64], [65]. The complexity can be expressed as the number of obstacles between the start and target points. Two types of environments are studied in this research, the cluttered environment and the corridor environment each which is divided into four environments( A, B, C and D) that changes its complexity from A to D as shown in Fig. 3 and Fig. 4.




(a) Environment A (b) Environment B

(c) Environment C (d) Environment D

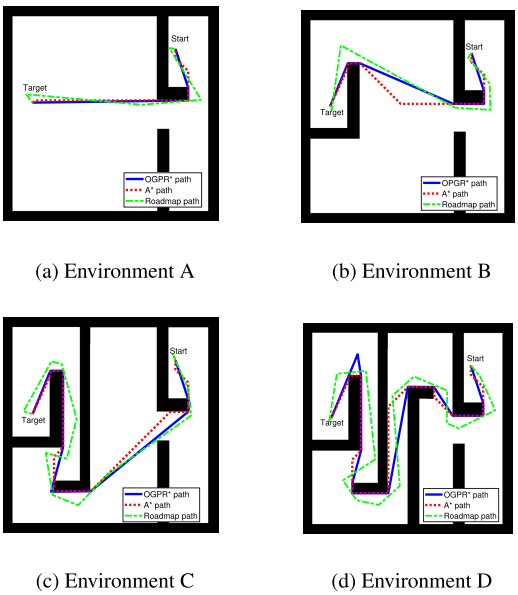
**FIGURE 3.** Cluttered environments to evaluate OGPR* versus A*.

**TABLE 1.** Time in seconds of cluttered environments A, B, C and D.

| | MS | 63*63 | 125*125 | 250*250 | 500*500 | 1000*1000 |
|---|---|---|---|---|---|---|
| Env. A | OGPR* | 0.074538 | 0.073815 | 0.073204 | 0.07817 | 0.11523 |
| | A* | 0.042865 | 0.153895 | 1.329094 | 18.433805 | 333.062615 |
| | Roadmap | 0.266832 | 0.3081 | 0.4173 | 0.6394 | 1.12527 |
| Env. B | OGPR* | 0.081251 | 0.071405 | 0.087892 | 0.092261 | 0.130737 |
| | A* | 0.040469 | 0.129353 | 1.103804 | 14.957442 | 264.685791 |
| | Roadmap | 0.252824 | 0.314429 | 0.420543 | 0.6433 | 1.13839 |
| Env. C | OGPR* | 0.086916 | 0.088092 | 0.089379 | 0.105075 | 0.148306 |
| | A* | 0.047767 | 0.134373 | 1.06095 | 13.843585 | 246.52376 |
| | Roadmap | 0.233148 | 0.29988 | 0.40052 | 0.60827 | 1.121064 |
| Env. D | OGPR* | 0.096301 | 0.098342 | 0.12654 | 0.125655 | 0.172702 |
| | A* | 0.058304 | 0.259541 | 2.719614 | 39.439068 | 694.218762 |
| | Roadmap | 0.2496 | 0.30306 | 0.4164 | 0.63472 | 1.106199 |

**TABLE 2.** Length in pixels of cluttered environments A, B, C and D.

| | MS | 63*63 | 125*125 | 250*250 | 500*500 | 1000*1000 |
|---|---|---|---|---|---|---|
| Env. A | OGPR* | 57.2779 | 113.6756 | 227.5286 | 454.9773 | 910.1666 |
| | A* | 61.3553 | 121.7107 | 243.4214 | 486.8427 | 973.6854 |
| | Roadmap | 62.7141 | 125.9346 | 251.2884 | 537.9077 | 964.9287 |
| Env. B | OGPR* | 64.6754 | 129.0818 | 259.0538 | 517.8347 | 1037.2 |
| | A* | 68.3848 | 136.3553 | 273.8823 | 547.7645 | 1096.7 |
| | Roadmap | 71.3971 | 142.6817 | 305.0327 | 564.0855 | 1241.5 |
| Env. C | OGPR* | 64.6754 | 129.0818 | 259.0538 | 517.8347 | 1037.2 |
| | A* | 68.3848 | 136.3553 | 273.8823 | 547.7645 | 1096.7 |
| | Roadmap | 74.5118 | 143.364 | 310.1920 | 604.8605 | 1230.0 |
| Env. D | OGPR* | 72.3463 | 144.9726 | 290.3888 | 580.8653 | 1164.9 |
| | A* | 77.598 | 155.3675 | 311.3209 | 622.6417 | 1247.3 |
| | Roadmap | 79.7365 | 172.2276 | 351.423 | 681.0264 | 1324.7 |




(a) Environment A (b) Environment B

(c) Environment C (d) Environment D

**FIGURE 4.** Corridor environments to evaluate OGPR* versus A*.

Firstly, the time cost of the cluttered environment is shown in Table 1 where the difference between the size of the environment affects slightly the time cost of OGPR*. Furthermore, the complexity of the environment also slightly affects the time cost of the OGPR*. On the other hand, the time cost of A* algorithm dramatically increases with the size of the environment and requires much more time than OGPR* in large environments which makes it difficult to be used without resizing the environment or decreasing its resolution to be bigger than one-pixel unit.

Moreover, Table 2 illustrates the path lengths of the four environments in different sizes where 'MS' represents the map size in pixels. There is no doubt that OGPR* path is shorter than A* path in all the environments.

Secondly, the corridor environment is compared to the A* algorithm as well to deeply investigate the complex environments and its effect on the two algorithms. Tables 3 and 4 shows the detailed behaviour of the OGPR* and A* in terms of the time cost and the length of the path for different sizes of the environments respectively.

The results show that the average reduction percentage of the time cost of the A* using the OGPR* is about 99.95% in $1000 \times 1000$ pixels environment. Although the OGPR* in corridor environment consumes much more time than itself in the cluttered environment and On the contrary, the A* in corridor environments consumes less time than itself in cluttered environments, the OGPR* algorithm has much less time than the A* in cluttered environments with size bigger than $250 \times 250$ and in corridor environments with size bigger than $500 \times 500$. Interestingly, mostly the path length of the OGPR* is much less than the one of the A* and the roadmap with an average of about 1% and 14% respectively in both the cluttered and corridor environments. Although, the roadmap used in this comparison has the smallest possible maximum number of nodes property to develop the lowest execution

**TABLE 3.** Time in seconds of corridor environments A, B, C and D.

| | MS | 63*63 | 125*125 | 250*250 | 500*500 | 1000*1000 |
|---|---|---|---|---|---|---|
| Env. A | OGPR* | 0.120 | 0.211 | 0.446 | 0.574 | 0.6493 |
| | A* | 0.0318 | 0.076 | 0.5139 | 5.3716 | 84.93 |
| | Roadmap | 0.24933 | 0.304611 | 0.3830 | 0.5955 | 1.06535 |
| Env. B | OGPR* | 0.141 | 0.2808 | 1.304 | 1.596 | 2.208 |
| | A* | 0.0233 | 0.26 | 2.370 | 31.886 | 495.84 |
| | Roadmap | 0.246712 | 0.2908 | 0.389413 | 0.579527 | 1.070804 |
| Env. C | OGPR* | 0.3717 | 1.164 | 1.366 | 1.665 | 1.866 |
| | A* | 0.069 | 0.3741 | 4.249 | 52.989 | 852.96 |
| | Roadmap | 0.243728 | 0.30164 | 0.3907 | 0.594061 | 1.01261 |
| Env. D | OGPR* | 0.5096 | 1.530 | 1.99 | 2.29 | 2.535 |
| | A* | 0.0528 | 0.376 | 3.729 | 49.269 | 777.39 |
| | Roadmap | 0.235545 | 0.287813 | 0.3815 | 0.589926 | 1.052773 |

**TABLE 4.** Length in pixels of corridor environments A, B, C and D.

| | MS | 63*63 | 125*125 | 250*250 | 500*500 | 1000*1000 |
|---|---|---|---|---|---|---|
| Env. A | OGPR* | 63.3535 | 122.71 | 245.371 | 487.25 | 972.9 |
| | A* | 64.8995 | 124.72 | 249.28 | 496.497 | 991.99 |
| | Roadmap | 70.5297 | 145.8280 | 310.2472 | 548.8087 | 1179.3 |
| Env. B | OGPR* | 72.85 | 144.18 | 289.265 | 575.4 | 1146.6 |
| | A* | 77.769 | 153.639 | 306.86 | 610.002 | 1218.8 |
| | Roadmap | 88.3355 | 179.6338 | 353.0360 | 703.3723 | 1471.5 |
| Env. C | OGPR* | 125.955 | 259.24 | 512.176 | 932.20 | 1864.9 |
| | A* | 120.98 | 241.48 | 424.9 | 954.97 | 1910.7 |
| | Roadmap | 143.7943 | 296.3440 | 549.0743 | 1154.2 | 2246.8 |
| Env. D | OGPR* | 152.44 | 307.496 | 605.4 | 1201.4 | 2365.7 |
| | A* | 146.84 | 290.367 | 576.42 | 1146.3 | 2293.1 |
| | Roadmap | 180.2982 | 373.4828 | 746.5772 | 1467.0 | 2843.8 |

time possible and increasing this maximum number develops shorter paths, they still are longer than the OGPR* and the A* and the execution time increases dramatically of the roadmap algorithm.

Accordingly, OGPR* path planning is much more advanced and effective than A* in terms of time cost and is more effective than the road map in the path length.

### B. THE OGPR* SELECTION RATIO

The OGPR* selection ratio, Eq. 5, refers to the ratio between the number of pre-optimization paths used as initial paths for optimization which later generate a set of optimized paths and the shortest one is the OGPR* path.

$$\text{The OGPR* selection ratio} = \frac{\text{number of possibilities}}{2^n} * 100\% \quad (5)$$

where $n$ is the total number of intersected obstacles.

To demonstrate the effect of the OGPR* selection ratio on the path length and time cost, a complex environment with a size of 1000*1000 pixels as shown in Fig. 5 is used for this comparison between different selection ratios and with A* algorithm also. Furthermore, fig. 6 shows that the time cost increase almost gradually with the increase of the selection ratio (SR) but the path length decreased by 3.869% at the selection ratio of 25% and kept steady after that.

Thus, OGPR* doesn't need to test all the possibilities of pre-optimization paths and the corresponding optimized paths since it takes longer time which is a disadvantage in time-critical situations. In situations where the path length is much critical than the time cost, the OGPR* do test all the possibilities of pre-optimization paths and the corresponding optimized, and in situations where they both are critical, part of the possibilities of the selection ratio can be examined to choose the shortest path in suitable time cost since it doesn't need to test all the selection ratios to get the shortest path.

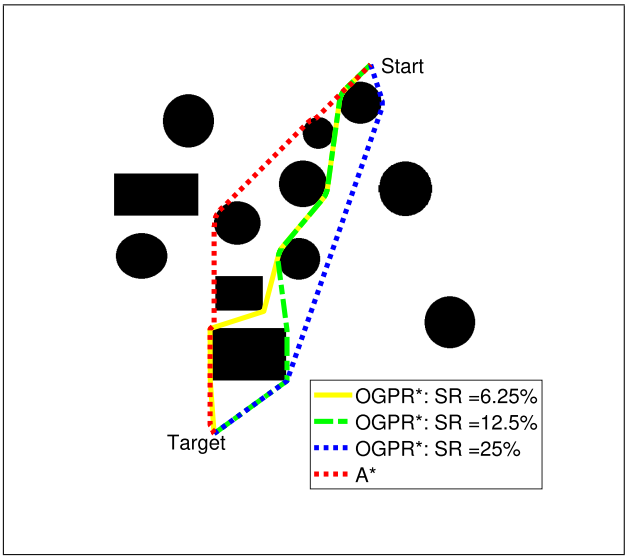
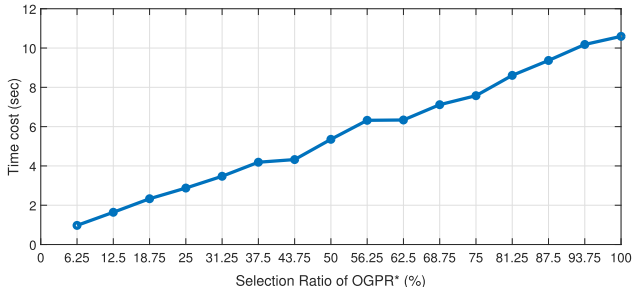
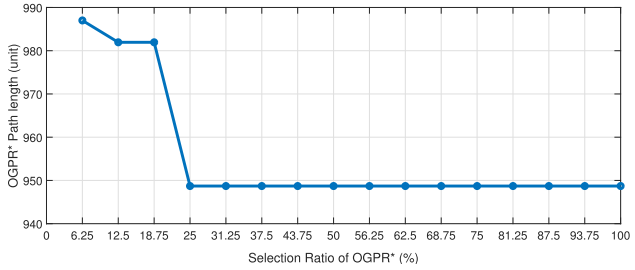



**FIGURE 5.** OGPR* path and A* path.




(a) Time cost relation




(b) Path length relation

**FIGURE 6.** Effect of Selection Ratio in OGPR*.

On the other hand, the time cost of the A* algorithm is 239.367 seconds which is so much higher than the OGPR*, but A* path length is 984.46 pixels which is slightly shorter than OGPR* of selection ratio of $\frac{1}{16}$ and longer than other OGPR* of selection ratios.

## IV. SIMULATION IN V-REP ENVIRONMENT

In this study, the simulation environment is built-in V-REP simulation because its simulation is quite close to the real-time [66] and it could be similar to the outer circumstances of exploration in the real world where the robot has to explore a large area and the resolution of the quadcopter camera only permits a small space in each time. So, the environment is divided into two terrains and the robot is controlled to reach the closest green point which represents the target of the first terrain and then move to the second green point which is the final target. The first terrain contains two black spheres represent the obstacles that the mobile robot needs to avoid in his way to the green target. While the second is made up of seven small black spheres that serve as obstacles. further, The quadcopter has a high resolution (520*520) camera to detect the obstacles and the position of the mobile robot which is mainly a simple pioneer robot with no sensors integrated, since the vision system of the quadcopter is used instead of the feedback sensors.

To validate and test the OGPR* algorithm, two sets of experiments, Static and Dynamic obstacles are designed to explore an environment while avoiding obstacles using an aerial robot and a ground mobile robot. The evaluation process uses A* algorithm as the reference and compares its results with the OGPR* algorithm.

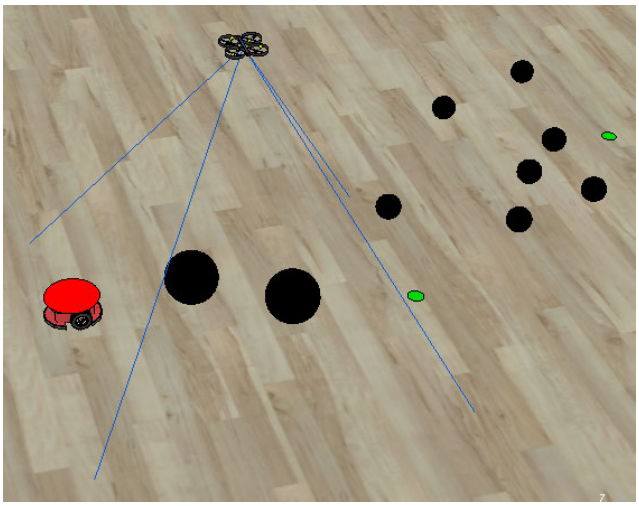


**FIGURE 7.** Static Environment.

In the simulation process, a collaborative strategy between the quadcopter, and the mobile robot is proposed as shown in Fig. 12. in this strategy, the quadcopter is the master, where all the path planning calculations are done on its processor, while the mobile robot just follows the commands coming from the quadcopter. Furthermore, the position of the mobile robot in the image of the quadcopter camera is only identified using its red colour and the target is painted green and using colour recognition the pixel position of the robot and the target are known the whole time within the image.

Although the process of controlling the position of the mobile robot requires the knowledge of its current position and direction and the required position and direction of the path in the environment, the system does not require the exact position of the robot since the robot and path are defined as nodes in the image and the error between the V-rep environment and the image is compensated by the PI controller gains. Using the inverse kinematics of the pioneer [67], the error in the direction between the robot and the path is determined and is controlled as well as the position error.

The required positions of the robot are given from the path planning of the robot using the OGPR* or A* algorithms. Interestingly, a PI controller performed efficiently driving the robot through the whole path to the target. After the robot finishes the first terrain and reaches its target, it sends a signal to the quadcopter which then flies horizontally to a suitable position in the middle of the second terrain to track the robot during its motion. Then, the loop starts again, and a new path is generated between the robot position and the target. Ultimately, this process can be repeated in situations where a large set of terrains is presented and a few certain targets in each of them. Fig. 8 shows the process of collaboration between the mobile robot and the quadcopter.

## V. RESULTS AND DISCUSSION

The results of the simulation are divided into static obstacles and the dynamic obstacles to evaluating the response of the

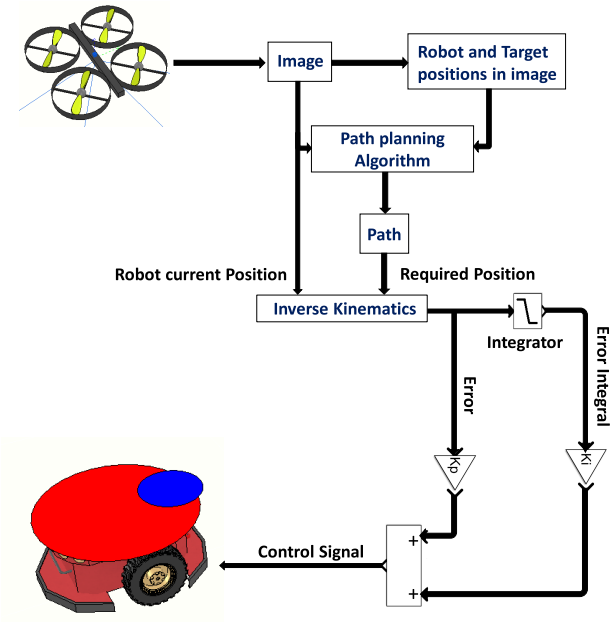



**FIGURE 8.** Collaborative behaviour.

OGPR* algorithm in critical situations in which the time cost is crucial as well as the path length.

### A. STATIC OBSTACLES ENVIRONMENT

The pioneer mobile robot and a quadcopter system are used as a platform to test the validity of the OGPR* path. The quadcopter is fixed in a suitable position n in the middle of the terrain. The size of the map is $512 \times 512$ which the resolution of the quadcopter camera. Fig. 9(a-c) shows snapshots from the camera in the first terrain during the path tracking of the robot to the OGPR* path. The static environment consists of two terrains which the target point of the first terrain in the start point of the second terrain. When the robot reaches the target in the first terrain in Fig. 9(c), the quadcopter moves to another suitable position to the second terrain that includes the pioneer and the second target. Fig. 9(d-f) shows the tracking of the pioneer to the OGPR* path in the second terrain. Similarly, Fig 10(a-c) shows the robot tracking of the A* path between the start and target points in the first terrain and Fig 10(d-f) shows the robot tracking of the A* path in the second terrain.

The cost time of OGPR* algorithm is 0.139006 and 0.1102 seconds respectively for the first and second environments, while the cost time of A* is 14.733410 and 8.8575 seconds for them respectively which is so much higher. On the other hand, A* algorithm is used for planning a path in the same static environment.

Furthermore, the total path travelled by the robot using A* algorithm is 884.4544 length unit which is also higher than the path length generated by the OGPR* algorithm which is 840.7855 length unit. The maximum velocity of the Right wheel and the left wheel using OGPR* is 2.84 m/s and 3.02 m/s respectively. Similarly, the maximum velocity of

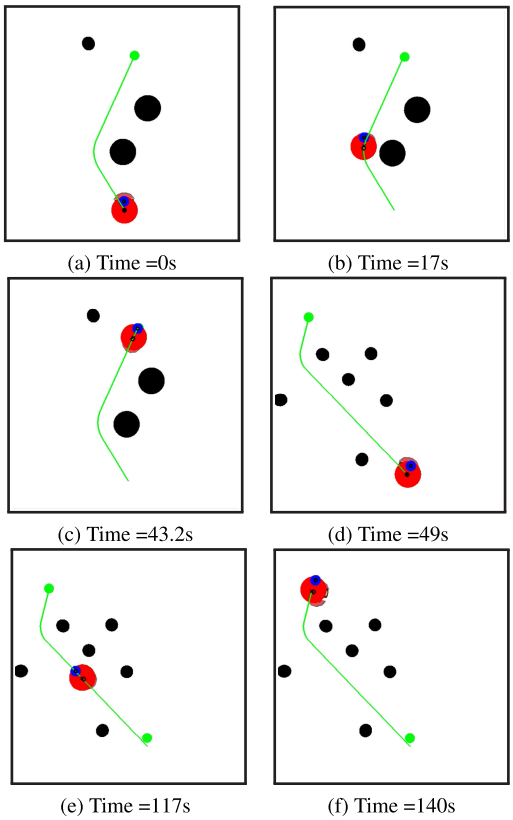



**FIGURE 9.** Static environment using OGPR* algorithm: (a),(b) and (c) are first terrain and (d),(e) and (f) are the second terrain.

the Right wheel and the left wheel using A* is 2.99 m/s and 3.17 m/s respectively. The estimated path generated of the algorithm and the actual path travelled by the pioneer robot are shown in Fig. 11. This means that the robot requires more time to plan the path and also reaching the target and energy in the A* algorithm because it has much longer execution time and path length than these of the OGPR* algorithm.

### B. DYNAMIC OBSTACLES ENVIRONMENT

The environment with dynamic obstacles is used for the evaluation of the OGPR*, to describe the behaviour of the mobile robot in moving obstacles environment where the time cost is critical and needs to be small as possible to update the free-collision path before any collisions. In the simulation, the algorithm is repeated in a loop for every short time to update the robot path in each step it takes.

#### 1) $512 \times 512$ ENVIRONMENT

The OGPR* algorithm is used to map and guide the pioneer robot while avoiding moving obstacles. The map resolution is $512 \times 512$ pixels. In this environment, one obstacle is moving cutting the path of the robot, and another static obstacle as shown in Fig. 12.

Because of the small value of the OGPR* algorithm time cost, it successfully guided the robot from the start point to the target point without any collision with the moving obstacles.

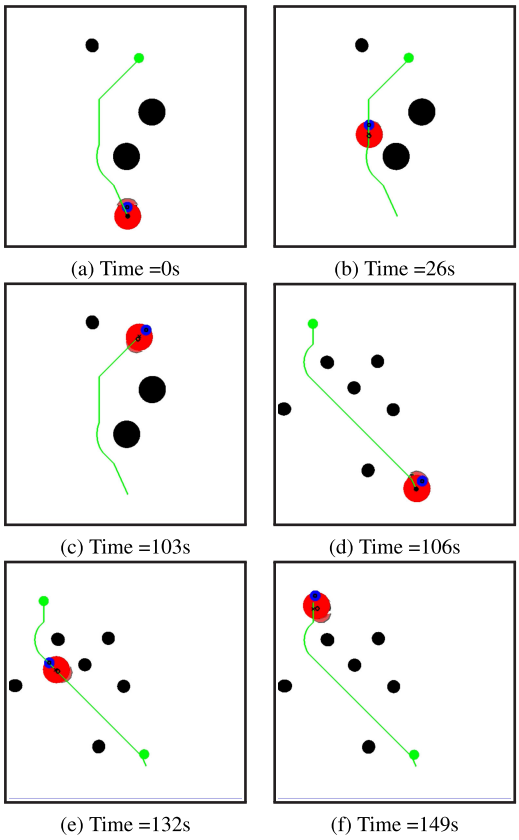



**FIGURE 10.** Static environment using A* algorithm: (a),(b) and (c) are first terrain and (d),(e) and (f) are the second terrain.

On the other hand, the A* algorithm was so slow that the robot takes almost 16 seconds to know its next move so that it can't avoid any obstacle directed to its path.

### 2) SCALING THE ENVIRONMENT

One disadvantage of the A* algorithm is that its execution time increases greatly by increasing the size of the map. In this experiment, the map is resized to minimize the size of the map and then execute the A* algorithm and getting the A* path and then resizing the path to its original map to reduce the run time of the algorithm. The same steps are done to the OGPR* algorithm to compare the two algorithms concerning small size maps in a moving environment. Fig. 13 shows the moving path planning using A* algorithm after scaling the environment to 20 % of its size. Although the robot takes about 0.0523 second to generate the first straight free-collision path, the path is harsh and has sharp edges due to the scaling. That is why it is not recommended to scale down the environment to use the A* algorithm.

Furthermore, increasing the scaling factor of the environment greatly affects the path tracking and path generation of the robot using A* algorithm. Fig.14 shows the moving path planning using A* algorithm with a scaling factor of 60%.

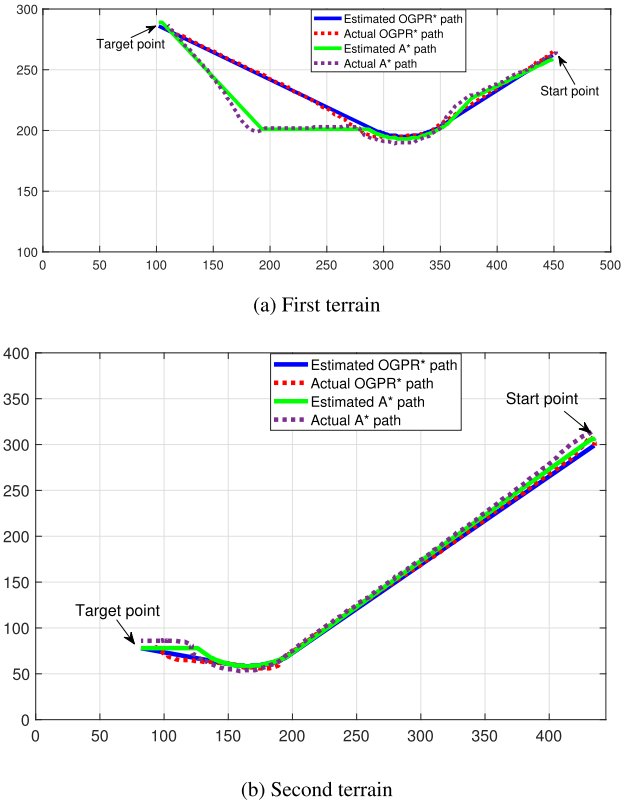



**FIGURE 11.** The path of the estimated path and the actual path.

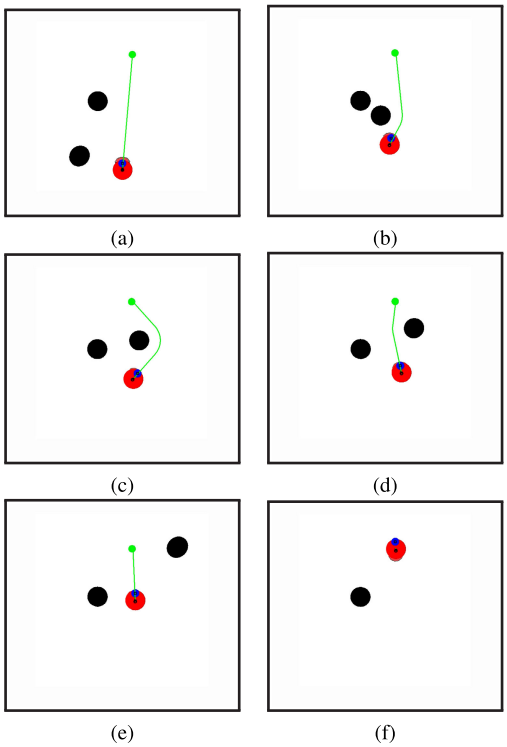



**FIGURE 12.** Moving obstacles environment using OGPR* algorithm.

It is clear that that the robot has unbalanced rotation until it reaches the target since it requires the robot to generate the first free-path more than 1.5 seconds.

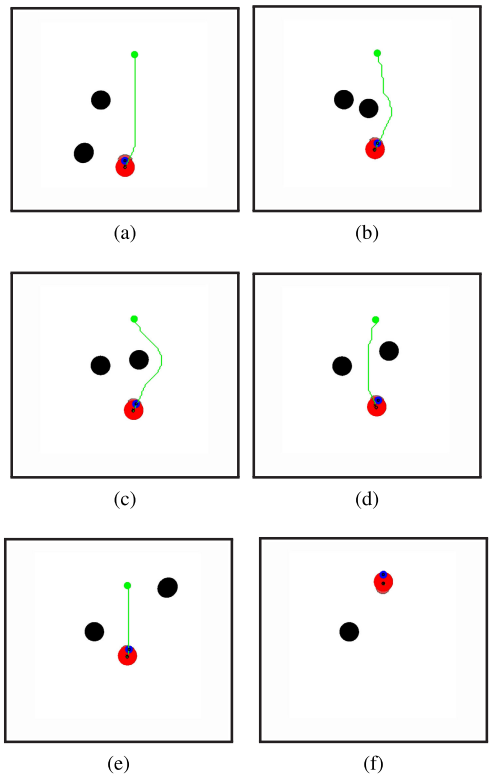



**FIGURE 13.** Moving environment using A* algorithm of 0.2 scale environment (102 × 102 pixels).

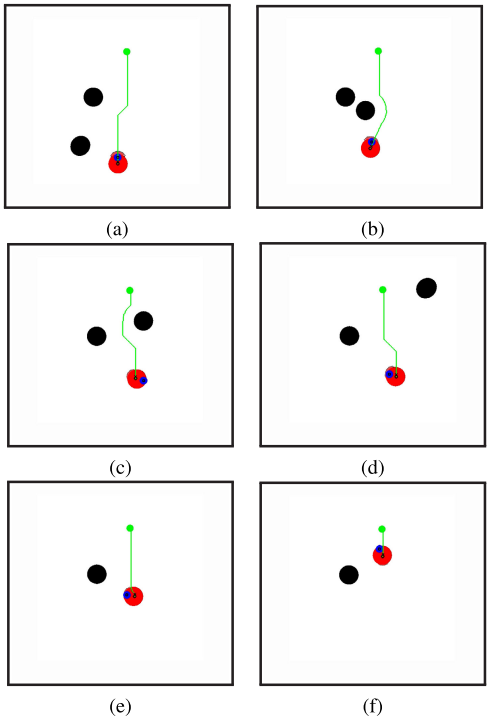



**FIGURE 14.** Moving environment using A* algorithm of 0.6 scale environment (307 × 307 pixels).

## VI. CONCLUSION

In this article, an OGPR* algorithm is proposed that would allow the OGPR algorithm previously proposed to collaboratively plan safe robot paths. To gain a deeper understanding of the OGPR* algorithm, a comparison between the state-of-art A* and the roadmap algorithms has been conducted with cluttered and corridor environments varying in their complexities and sizes. Furthermore, two real-time simulations with an aerial and a mobile robot with static and dynamic obstacles have been used to evaluate the performance of the OGPR* algorithm in critical time situations as compared to the A* algorithm. In the static obstacles simulation, the time reduction ratio of the OGPR* is about 98.94% compared to the A* algorithm while a reduction of the total path length to 4.94% is obtained using OGPR* compared to the A* algorithm. Additionally, in the dynamic obstacles simulation, the OGPR* has successfully guided the ground robot to its target point without collision. On the other hand, the A* algorithm has failed to plan safe paths in a 512 × 512 pixels environment unless its size is eventually reduced to 20% where jerky movements are obtained. However, the proposed OGPR* has been found to be more effective in guiding a ground robot toward a full-sized environment with no need for reduction.

Future research directions could be involving real robots in the experiments. Also, Monte-Carlo simulations could be conducted to verify the robustness of the proposed OGPR* algorithm.

## REFERENCES

[1] I. Arvanitakis and A. Tzes, "Collaborative mapping and navigation for a mobile robot swarm," in *Proc. 25th Medit. Conf. Control Autom. (MED)*, Jul. 2017, pp. 696–700.
[2] A. H. Memar and E. T. Esfahani, "Physiological measures for human performance analysis in human-robot teamwork: Case of tele-exploration," *IEEE Access*, vol. 6, pp. 3694–3705, 2018.
[3] E. A. Rodriguez Martinez, G. Caron, C. Pegard, and D. L. Alabazares, "Photometric path planning for vision-based navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 9007–9013.
[4] L. Yu, D. Kong, X. Shao, and X. Yan, "A path planning and navigation control system design for driverless electric bus," *IEEE Access*, vol. 6, pp. 53960–53975, 2018.
[5] T. Dang, S. Khattak, C. Papachristos, and K. Alexis, "Anomaly detection and cognizant path planning for surveillance operations using aerial robots," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2019, pp. 667–673.
[6] B. Fang, J. Ding, and Z. Wang, "Autonomous robotic exploration based on frontier point optimization and multistep path planning," *IEEE Access*, vol. 7, pp. 46104–46113, 2019.
[7] J. Kim, A. K. Mishra, R. Limosani, M. Scafuro, N. Cauli, J. Santos-Victor, B. Mazzolai, and F. Cavallo, "Control strategies for cleaning robots in domestic applications: A comprehensive review," *Int. J. Adv. Robotic Syst.*, vol. 16, no. 4, 2019, p. 1729881419857432.
[8] X. Miao, J. Lee, and B.-Y. Kang, "Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments," *IEEE Access*, vol. 6, pp. 38200–38215, 2018.
[9] H. El-Hussieny, S. F. M. Assal, and M. Abdellatif, "Robotic exploration: New heuristic backtracking algorithm, performance evaluation and complexity metric," *Int. J. Adv. Robotic Syst.*, vol. 12, no. 4, p. 33, Apr. 2015.
[10] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Jan. 2019.
[11] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 3260–3267.
[12] Y. Mu, B. Li, D. An, and Y. Wei, "Three-dimensional route planning based on the beetle swarm optimization algorithm," *IEEE Access*, vol. 7, pp. 117804–117813, 2019.

[13] D. Drake, S. Koziol, and E. Chabot, "Mobile robot path planning with a moving goal," *IEEE Access*, vol. 6, pp. 12800–12814, 2018.

[14] A. Ivanovic, M. Polic, O. Salah, M. Orsag, and S. Bogdan, "Compliant net for AUV retrieval using a UAV," *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 431–437, 2018.

[15] P. Victerpaul, D. Saravanan, S. Janakiraman, and J. Pradeep, "Path planning of autonomous mobile robots: A survey and comparison," *J. Adv. Res. Dyn. Control Syst.*, vol. 9, pp. 1535–1565, Mar. 2017.

[16] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3D path planning algorithms," *J. Control Sci. Eng.*, vol. 2016, Jul. 2016, Art. no. 7426913.

[17] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[19] C. Yin and H. Wang, "Developed Dijkstra shortest path search algorithm and simulation," in *Proc. Int. Conf. Comput. Design Appl.*, Jun. 2010, pp. V1–116.

[20] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *Proc. 2nd Int. Conf. Mechanic Autom. Control Eng.*, Jul. 2011, pp. 1067–1069.

[21] M. Luo, X. Hou, and J. Yang, "Surface optimal path planning using an extended dijkstra algorithm," *IEEE Access*, vol. 8, pp. 147827–147838, 2020.

[22] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Eng.*, vol. 96, pp. 59–69, 2014.

[23] L. H. O. Rios and L. Chaimowicz, "A survey and classification of A* based best-first heuristic search algorithms," in *Proc. Brazilian Symp. Artif. Intell.* Berlin, Germany: Springer, 2010, pp. 253–262.

[24] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[25] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, May 2002, pp. 968–975.

[26] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta∗: Any-angle path planning on grids," in *Proc. AAAI*, vol. 7, 2007, pp. 1177–1183.

[27] P. K. Y. Yap, N. Burch, R. C. Holte, and J. Schaeffer, "Any-angle path planning for computer games," in *Proc. 7th Artif. Intell. Interact. Digit. Entertainment Conf.*, 2011, pp. 1–7.

[28] A. Stentz, "The focussed D∗ algorithm for real-time replanning," in *Proc. IJCAI*, vol. 95, 1995, pp. 1652–1659.

[29] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1853

[30] L. Jaillet, J. Cortes, and T. Simeon, "Transition-based RRT for path planning in continuous cost spaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 2145–2150.

[31] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path planning of industrial robot based on improved RRT algorithm in complex environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018.

[32] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.

[33] I. Noreen, A. Khan, and Z. Habib, "A comparison of RRT, RRT* and RRT*-smart path planning algorithms," *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)*, vol. 16, no. 10, p. 20, 2016.

[34] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy, and I. Ali, "Informed RRT*-connect: An asymptotically optimal single-query path planning method," *IEEE Access*, vol. 8, pp. 19842–19852, 2020.

[35] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput.*, vol. 77, pp. 236–251, Apr. 2019.

[36] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[37] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 1, Apr./May 2004, pp. 446–452.

[38] P. O. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned helicopter," *J. Intell. Fuzzy Syst.*, vol. 17, no. 4, pp. 395–405, 2006.

[39] J. P. van den Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 885–897, Oct. 2005.

[40] B. Sun, W.-D. Chen, and Y.-G. Xi, "Particle swarm optimization based global path planning for mobile robots," *Control Decis.*, vol. 20, no. 9, p. 1052, 2005.

[41] Y. Zhang, D.-W. Gong, and J.-H. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomputing*, vol. 103, pp. 172–185, Mar. 2013.

[42] M. Wang, "Fuzzy logic based robot path planning in unknown environment," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 2, Aug. 2005, pp. 813–818.

[43] P. Sun and Z. Yu, "Tracking control for a cushion robot based on fuzzy path planning with safe angular velocity," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 610–619, Sep. 2017.

[44] M. R. Jabbarpour, H. Zarrabi, J. J. Jung, and P. Kim, "A green ant-based method for path planning of unmanned ground vehicles," *IEEE Access*, vol. 5, pp. 1820–1832, 2017.

[45] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 5, Apr./May 2004, pp. 4350–4355.

[46] J. Tu and S. X. Yang, "Genetic algorithm based path planning for a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Sep. 2003, pp. 1221–1226.

[47] Q. Yang and S.-J. Yoo, "Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms," *IEEE Access*, vol. 6, pp. 13671–13684, 2018.

[48] J. Ni, K. Wang, Q. Cao, Z. Khan, and X. Fan, "A memetic algorithm with variable length chromosome for robot path planning under dynamic environments," *Int. J. Robot. Autom.*, vol. 32, no. 4, pp. 414–424, 2017.

[49] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst., Man Cybern., B (Cybern.)*, vol. 34, no. 1, pp. 718–724, Feb. 2004.

[50] Y. Peng, S.-W. Li, and Z.-Z. Hu, "A self-learning dynamic path planning method for evacuation in large public buildings based on neural networks," *Neurocomputing*, vol. 365, pp. 71–85, Nov. 2019.

[51] R.-J. Wai and A. S. Prasetia, "Adaptive neural network control and optimal path planning of UAV surveillance system with energy consumption prediction," *IEEE Access*, vol. 7, pp. 126137–126153, 2019.

[52] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," in *Proc. Int. Conf. Comput. Design Appl.*, vol. 3, Jun. 2010, pp. V3–436.

[53] T. Guan-Zheng, H. Huan, and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta Automatica Sinica*, vol. 33, no. 3, pp. 279–285, 2007.

[54] M. G. B. Atia, O. Salah, and H. Ei-Hussieny, "OGPR: An obstacle-guided path refinement approach for mobile robot path planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2018, pp. 844–849.

[55] M. G. B. Atia, O. Salah, and A. Fouly, "Path planning for robot manipulator based on obstacle-guided path refinement," in *Proc. 58th Annu. Conf. Soc. Instrum. Control Eng. Jpn. (SICE)*, Sep. 2019, pp. 437–442.

[56] Z. Zhang, J. Wu, J. Dai, and C. He, "A novel real-time penetration path planning algorithm for stealth UAV in 3D complex dynamic environment," *IEEE Access*, vol. 8, pp. 122757–122771, 2020.

[57] F. H. Tseng, T. T. Liang, C. H. Lee, L. D. Chou, and H. C. Chao, "A star search algorithm for civil UAV path planning with 3G communication," in *Proc. 10th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Aug. 2014, pp. 942–945.

[58] E. Chicurel-Uziel, "Single equation without inequalities to represent a composite curve," *Comput. Aided Geometric Des.*, vol. 21, no. 1, pp. 23–42, Jan. 2004.

[59] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Math. Comput. Appl.*, vol. 10, no. 1, pp. 45–56, Apr. 2005.

[60] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018.

[61] B. Cao, J. Zhao, Z. Lv, X. Liu, S. Yang, X. Kang, and K. Kang, "Distributed parallel particle swarm optimization for multi-objective and many-objective large-scale optimization," *IEEE Access*, vol. 5, pp. 8214–8221, 2017.

[62] A. A. Nagra, F. Han, Q.-H. Ling, and S. Mehta, "An improved hybrid method combining gravitational search algorithm with dynamic multi swarm particle swarm optimization," *IEEE Access*, vol. 7, pp. 50388–50399, 2019.
[63] R. M. C. Santiago, A. L. De Ocampo, A. T. Ubando, A. A. Bandala, and E. P. Dadios, "Path planning for mobile robots using genetic algorithm and probabilistic roadmap," in *Proc. IEEE 9th Int. Conf. Humanoid, Nanotechnol., Inf. Technol., Commun. Control, Environ. Manage. (HNICEM)*, Dec. 2017, pp. 1–5.
[64] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef, "Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments," *Int. J. Adv. Robotic Syst.*, vol. 14, no. 2, 2017, Art. no. 1729881416663663.
[65] R. Uriol and A. Moran, "Mobile robot path planning in complex environments using ant colony optimization algorithm," in *Proc. 3rd Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2017, pp. 15–21.
[66] K. K. Reddy and K. Praveen, "Path planning using VREP," *Int. J. Res. Eng. Technol.*, vol. 02, no. 09, pp. 94–97, Sep. 2013.
[67] M. Egerstedt, "Motion planning and control of mobile robots," Ph.D. dissertation, Dept. Math., Matematik, KTH Roy. Inst. Technol., Stockholm, Sweden, 2000.

**MOHAMED G. B. ATIA** (Graduate Student Member, IEEE) received the B.Sc. degree (Hons.) from the Department of Mechanical Engineering, Faculty of Engineering, Mechatronics Section, Assiut University, Assiut, Egypt, in 2016. He has been a Teaching Assistant with the Department of Mechatronics and Robotics Engineering, Egypt-Japan University of Science and Technology (E-JUST), Alexandria, Egypt. His research interests include exoskeletons, path planning, soft robotics, and mechatronics. He has been awarded the Ph.D. Scholarship in mechanical, materials and manufacturing engineering from the University of Nottingham, U.K.

**HAITHAM EL-HUSSIENY** received the B.Sc. degree in electronics and communication engineering from the Faculty of Engineering (Shoubra), Benha University, Egypt, in 2007, and the M.Sc. and Ph.D. degrees in mechatronics and robotics engineering from the Egypt-Japan University of Science and Technology (E-JUST), Alexandria, Egypt, in 2013 and 2016, respectively. He is currently working as an Assistant Professor of robotics engineering with the Department of Electrical Engineering, Faculty of Engineering (Shoubra), Benha University. Since August 2019, he has been working as a Senior Research Fellow in soft robotics with the University of Salford, Manchester, U.K. His research interests include soft robots, soft haptics, teleoperation, model predictive control, and applied intelligence.

**OMAR SALAH** received the B.Sc. degree (Hons.) from the Department of Mechanical Engineering, Mechatronics Section, Assiut University, in 2007, and the M.Sc. and Ph.D. degrees in mechatronics and robotics engineering from the Innovation Design Engineering School, Egypt-Japan University of Science and Technology (E-JUST), in 2012 and 2015, respectively. He has worked as an Exchange Researcher with Waseda University, Japan. He also worked as a Postdoctoral Research Fellow with LARICS, Zagreb University, Croatia. He is currently a Lecturer with the Department of Mechanical Engineering, Faculty of Engineering, Assiut University. His current research interests include mechatronics, robotics, path planning, and assistive devices.