Robotica: (2020) volume 38, pp. 235–255. © Cambridge University Press 2019
doi:10.1017/S0263574719000572

Investigating Reduced Path Planning Strategy for Differential Wheeled Mobile Robot

Raouf Fareh†*, Mohammed Baziyad‡, Mohammad H. Rahman¶, Tamer Rabie† and Maamar Bettayeb†,§

† *Electrical and Computer Engineering Department, University of Sharjah, UAE*
E-mails: trabie@sharjah.ac.ae, maamar@sharjah.ac.ae
‡ *Research Institute of Sciences and Engineering (RISE), University of Sharjah, UAE*
E-mail: mbaziyad@sharjah.ac.ae
¶ *Mechanical/Biomedical Engineering Department, University of Wisconsin-Milwaukee, Milwaukee, USA*
E-mail: rahmanmh@uwm.edu
§ *MCEIES, King Abdulaziz University, Jeddah, KSA*

(Accepted March 30, 2019. First published online: May 14, 2019)

SUMMARY
This paper presents a vision-based path planning strategy that aims to reduce the computational time required by a robot to find a feasible path from a starting point to the goal point. The proposed algorithm presents a novel strategy that can be implemented on any well-known path planning algorithm such as A*, D* and probabilistic roadmap (PRM), to improve the swiftness of these algorithms. This path planning algorithm is suitable for real-time scenarios since it reduces the computational time compared to the basis and traditional algorithms. To test the proposed path planning strategy, a tracking control strategy is implemented on a mobile platform. This control strategy consists of three major stages. The first stage deals with gathering information about the surrounding environment using vision techniques. In the second stage, a free-obstacle path is generated using the proposed reduced scheme. In the final stage, a Lyapunov kinematic tracking controller and two Artificial Neural Network (ANN) based-controllers are implemented to track the proposed path by adjusting the rotational and linear velocity of the robot. The proposed path planning strategy is tested on a Pioneer P3-DX differential wheeled mobile robot and an Xtion PRO depth camera. Experimental results prove the efficiency of the proposed path planning scheme, which was able to reduce the computational time by a large percentage which reached up to 88% of the time needed by the basis and traditional scheme, without significant adverse effect on the workability of the basis algorithm. Moreover, the proposed path planning algorithm has improved the path efficiency, in terms of the path length and trackability, challenging the traditional trade-off between swiftness and path efficiency.

KEYWORDS: Path planning; Image processing; Depth sensor; Mobile robot; Kinematic control; Probabilistic roadmap; Obstacle avoidance; Stability; Trajectory tracking.

1. Introduction

The path planning problem of an autonomous robotic system and its requirements have attracted researchers to this research area in the recent years.[1–10] An automated robot has the ability to acclimate to the surrounding environment and interact with it without direct human intervention. In

* Corresponding author. E-mail: rfareh@sharjah.ac.ae

order to build an effective autonomous system, fast and robust techniques are required to deal with the real-time nature of such applications. An autonomous robotics system tries to imitate human behaviors with an acceptable level of intelligence and precision in a real-time fashion. Avoiding moving obstacles, generating a free-obstacle path to the destination, face detection and recognition are all examples of such behaviors. However, real-time algorithms suffer from the enormous computational complexity and the massive resources required. As a result, a traditional trade-off between simplicity and efficiency have formed a challenging problem for researchers in this field, as enhancing the efficiency increases the complexity of a system, which may end up with a system that fails with real-time scenarios. One of the critical objectives for an autonomous robot is to compute a feasible path to its destination without colliding into an obstacle. To find a free-obstacle path, the robot must sense the environment instantaneously to find obstacles, calculate distances, and compute the shortest path all in a real-time fashion. In robotics, various tools are utilized for sensing the environment such as sonar, laser, and cameras (vision). In recent years, with the massive improvements in the processing power of modern computers, cameras (vision) became a widely used tool since it is the most human-like sensor available.[11–14]

In ref. [13], a non-holonomic mobile robot tracks a pre-defined path generated with the help of a vision system. Vision was also used as a sensor to estimate the location of the robot instantaneously without the need of the traditional location measurements. Wallack et al. [11] proposed a system that calibrates a vision system and a robot. The proposed system is equipped with multiple cameras in a calibration mode to successfully detect and locate objects of interest with reference to the robot's initial position and orientation.

In ref. [14], a framework for implementing computer vision functions on cognitive robotics is proposed. The proposed framework targets to combine visual perception and cognition processes in one platform. The framework provides capabilities for locating 3D objects in the real world shown in a 2D image plane. The framework was tested on an iCub robot and showed outstanding performance.

The path planning algorithm starts right after gathering the required information about the environment such as the location of obstacles. The objective of a path planning algorithm is to find a feasible path to the destination without colliding into an obstacle. Path planning algorithms can be classified into two main categories: the grid-based algorithms and the sampling-based algorithms.

First, grid-based algorithms[15,16] install a grid on a configuration space and represent each configuration with a point. The robot can freely navigate from a grid point to another if the region between these points is free and does not include obstacles. There is a substantial parameter when dealing with grid-based algorithms which is the grid resolution. The resolution parameter is simply the size of the grid in terms of points or pixels. The importance of choosing this parameter becomes more critical when dealing with real-time applications. Increasing this value may result in a slow system that may fail dealing with the real-time nature of an autonomous robotic system. However, having a low-resolution value can affect the robustness and the efficiency of the system, and the algorithm may fail finding a feasible path.

A* is an example of a grid-based algorithm.[17,18] The algorithm's strategy in finding the shortest path is by first finding a number of possible paths to the destination and then electing the path with the lowest traveling cost. This creates a tree of paths starting from the starting point, until one of the paths reaches the goal point.

Then, A* associates each path with a cost value that reflects the path length from the starting point to the destination using a special heuristic function. Finally, A* elects the path with the lowest cost value. **D*** is another example of a grid-based path planning algorithm.[19] Unlike A*, D* starts searching for the feasible path inversely from the goal point to the start point. In D*, each point has a pointer that points to the adjacent point that leads to the destination. The cost value is calculated to the goal for each point. The algorithm ends when the pointer is pointing to the starting point, and the path is obtained by tracking the pointer.

In the second category, sampling-based algorithms, a number of samples are randomly distributed in the free space. These points are then connected together, which results in having a pack of paths from the starting point to the goal point. Finally, the shortest path is selected by using Dijkstra's shortest path query. The probabilistic roadmap (PRM) algorithm is an example of a sampling-based algorithm.[7] In PRM, there are two critical parameters. The first parameter is the number of samples (N), and the second is the distance (D). N is simply the number of samples to be distributed in the

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

free space. PRM connects all points separated by this distance or less in the map. In ref. [8], a new path planning technique based on PRM is presented. The idea is to use a two-level system, or a PRM of PRMs. A manipulation graph, whose nodes represent stable positions of the manipulated objects while the edges represent transfer and transit actions. In the second level, PRM planners plan the actual motion for the transfer and transit paths. Another modification of PRM named Lazy-PRM is presented in ref. [9]. The algorithm tries to reduce the number of collision checks performed during planning.

Another sampling-based technique is the Rapidly-exploring Random Trees (RRTs) techniques.[2] These algorithms operate by incrementally building two RRTs rooted at the start and the goal configurations. The tree increases randomly in the search space and tends to grow towards large unsearched areas of the map. RRT techniques are popular in nonconvex, high-dimensional spaces, and are used widely in autonomous robotic motion planning.[20,21] In ref. [21], a modified RRT path planning technique is used in a 3D path planning system. The proposed system uses a Kinect V2 sensor to obtain a point cloud to emphasize the interest regions and the obstacles of the environment.

Once the path is generated, it is fed into a controller. The kinematic controller is a type of controllers that adjusts the robot's wheels' velocity to track a certain trajectory. The authors presented in ref. [12] a sliding mode control design to stabilize the mobile robot to a trajectory. A vision-based control system in ref. [11] is developed to solve real-time experiences. In ref. [22], a kinematic controller is used to track a PRM-based path obtained using a single RGBD sensor mounted on the ceiling. The stability of the system was proved using Lyapunov theory.

Similar to the resolution attribute in A* and D*, raising the values of the PRM parameters (N and D) increases the possibility of finding a path from starting point to goal point (if it exists), however, a very high value will increase the complexity and computation time of the system, which is not suitable for real-time applications. On the other hand, low values can decrease the computational time, but the planner might fail to find a path, even if it exists.

To overcome the time-consuming problem, this paper proposes a path planning strategy that can be implemented on any traditional path planning algorithm, which aims to reduce the computational time required to find a feasible path to the desired destination. The proposed path planning strategy consists of four main phases. First, a linear line is calculated from the starting point to the goal. In the second phase, the planner detects obstacles that intersect with the linear line. Next, the basis algorithm is performed around the obstacle to obtain a feasible path around each obstacle. Finally, the planner connects these paths together to establish a free-obstacle path from the starting point to destination.

The proposed path planning strategy is named "the reduced path planning strategy". Moreover, a vision-based control system is implemented to test the effectiveness of the proposed path planning strategy. First, information about the environment is collected using vision algorithms. Next, a free-obstacle path is calculated using the proposed path planning reduced scheme. Finally, a Lyapunov kinematic controller and two ANN-based controllers are developed to ensure a good tracking of the desired path. Experimental comparisons prove the efficiency and the effectiveness of the proposed path planning strategy. The proposed path planning strategy is able to reduce 88% of the computational time, without any effect on the workability of the basis algorithm.

The rest of this paper is organized as follows. In Section 2, the system description is illustrated. Section 3 explains the vision and the image processing algorithms. The proposed reduced path planning scheme is defined in Section 4. Section 5 presents the kinematic and dynamic modeling of the mobile robot. The control law for the mobile robot is calculated in Section 6. The experimental results are discussed in Section 7, and Section 8 presents the conclusion.

2. System Description

The system used is composed of a depth-RGB camera fixed on the ceiling with a height of **H** and a differential wheeled mobile robot that has two primary wheels fixed in the same axis in front. A small free castor rotational wheel is placed at the back of the robot for stability purposes. Figures 1 and 2 illustrate the general description of the system.

The position and orientation of the robot is represented by the generalized coordinate $q = \left[x, y, \emptyset\right]^T$.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

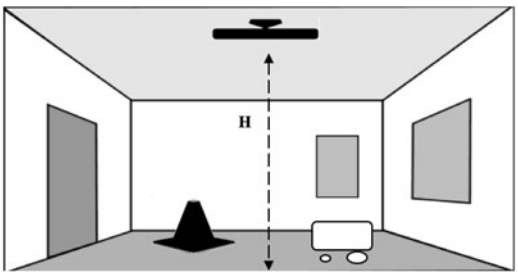

Fig. 1. General description of the system.

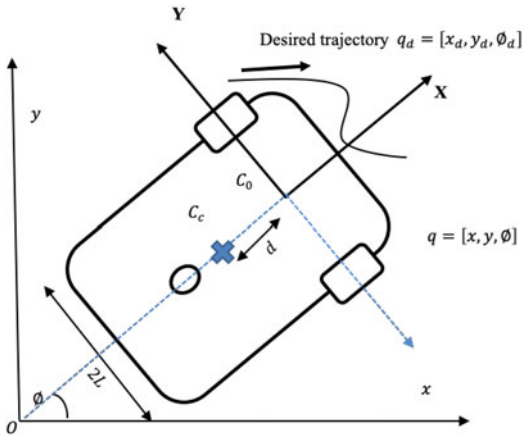

Fig. 2. The differential wheeled mobile robot.

$q_d = [x_d, y_d, \emptyset_d]^T$ denotes the desired trajectory calculated from the path planning algorithm.

The main objective of the control system is to track the path generated by the proposed path planning scheme using a kinematic control. Figure 3 presents the block diagram of the control design. It shows the main parts of the system; the vision part, the proposed reduced scheme planner, and the kinematic controller. Vision techniques assist the robot to understand the surrounding environment. The locations of obstacles, the starting point, and the goal point are all captured in this phase. The path planner takes this information along with the election of the basis algorithm and finds a feasible path from the starting point to the goal point. Finally, the kinematic controller helps the robot to track the desired path, by adjusting the linear and rotational velocities (v and w), based on the current measurements of the location and orientation of the robot with respect to the origin frame.

3. Vision and Image Processing Algorithms

Vision and image processing algorithms are essential tools that are used as a sensor in the system to detect obstacles and to find the coordinates of the staring and goal points. Since the proposed system targets real-time scenarios, these algorithms must meet some characteristics such as simplicity, robustness, and fast execution. To achieve that, a depth-RGB camera is mounted on the ceiling to detect and track obstacles. A depth camera is a device that can return an image where each pixel value represents the distance from the camera. First, the distance matrix $\boldsymbol{D}$ is calculated mathematically for all pixels where each index in the $\boldsymbol{D}$ matrix corresponds to the trigonometric real-world distance of the corresponding location from the camera (the trigonometric distance from the floor to ceiling (camera) for each point in the map). After that, the actual distance depth matrix $\boldsymbol{P}$ is obtained from the camera, and each pixel in the new matrix $\boldsymbol{P}$ is compared to each pixel in the matrix $\boldsymbol{D}$. If a certain index (x, y) in the $\boldsymbol{P_{xy}}$ matrix has a value less than the corresponding index value in the $\boldsymbol{D_{xy}}$ matrix, then the location (x, y) can be considered as an obstacle index. Figure 4 shows the obstacle tracking method used in the proposed system.

To detect the starting point and the goal point, an RGB image is required since these points are represented by two distinct colored labels placed in the corresponding points.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

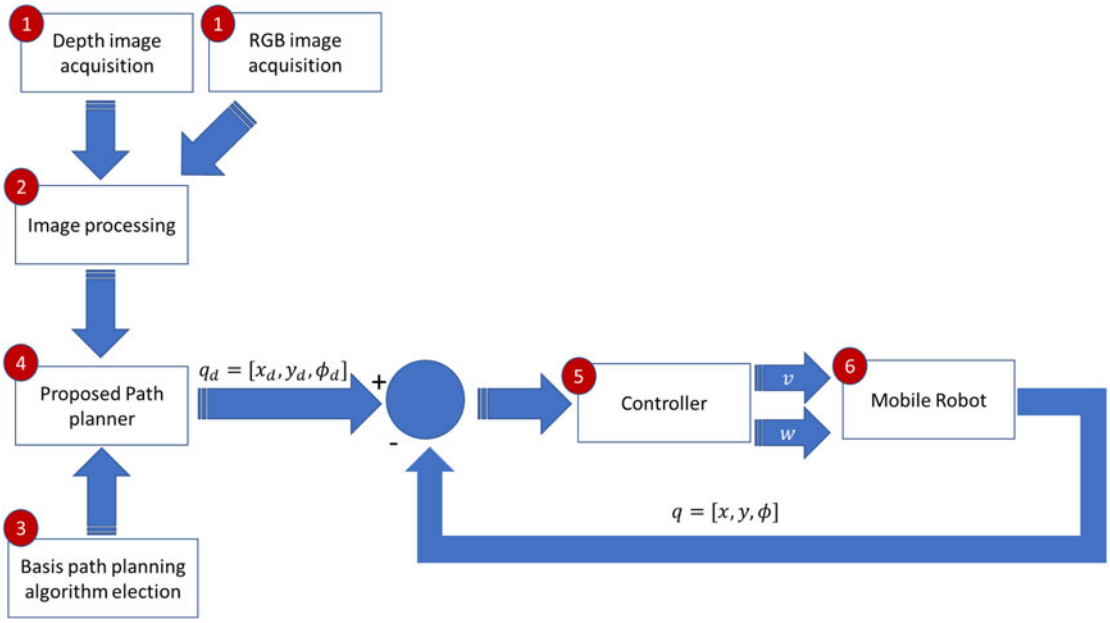

Fig. 3. Block diagram of the proposed control strategy.

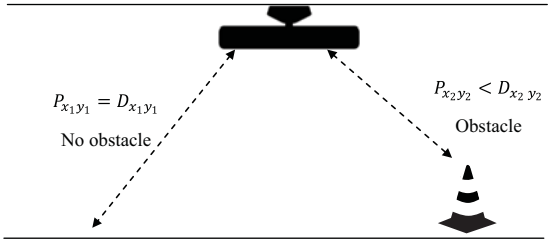

Fig. 4. Obstacle detection.

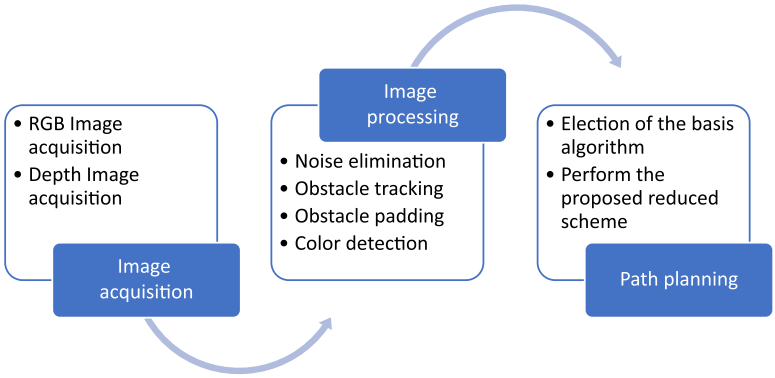

Fig. 5. Overview demonstration of the vision phase.

Before the path planning phase begins, these raw images must first run into a processing phase. The objective of this phase is to equip the planner with some essential parameters such as obstacle locations, starting point and goal point. This phase is composed of four subparts, namely the noise elimination, obstacle tracking, obstacle padding, and color detection. Figure 5 shows an overview of the vision and image processing phase.

Image acquisition: Two images are captured from the RGB-depth sensor, the RGB, and the depth image. The RGB image is retrieved in the three-channel format namely the red, green, and

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

4.1	3.9	3.9	4.2	4.1
4.2	3.8	3.9	4.1	4.0
4.2	4.1	0 3.8	4.0	3.9
3.9	3.8	4.2	4.2	4.0
3.8	3.9	4.1	4.0	3.9

Fig. 6. Noise elimination.

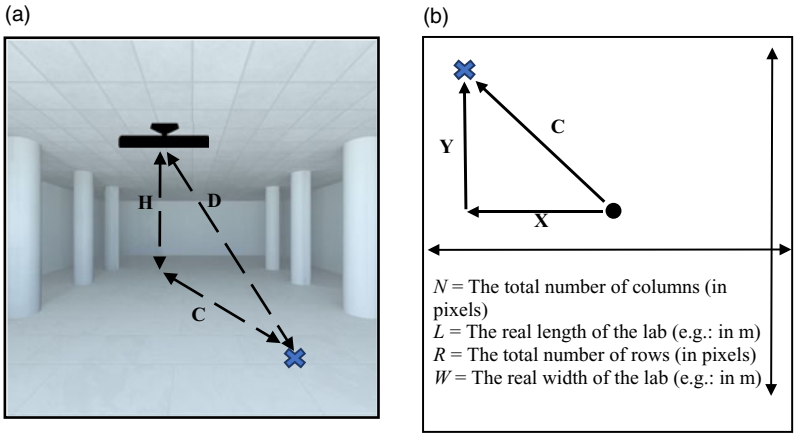

Fig. 7. Obstacle tracking technique. (a) Calculating the D distance (b) Top view sketch illustrates the calculations of the distance C.

blue channels. On the other hand, the depth image has pixels with values that represent the distances of objects from the camera.

Image processing: The raw images are processed to extract some fundamental information such as the coordinates of the starting point, the goal point and obstacles. A pre-processing step, which is noise elimination, is performed first since the depth camera is not an ideal sensor as it sometimes fails to estimate the distance. In this case, the corresponding pixel will have a value of zero which indicates the failure. To deal with this noise issue, the average value of the surrounding pixels is calculated and replaces the zero pixel. This operation is similar to filtering an image with an averaging filter, but replacing only zero pixels with the average value. Figure 6 presents an example of the noise elimination process. In this example, the zero pixel was replaced by the average value of 3.8.

Next, the obstacle tracking sub-part begins. The first step in the obstacle tracking phase is to mathematically calculate the distance of each point from the mounted camera. Figure 7 shows the procedure used to calculate the D distance, which is the real distance from the camera to each (x, y) location in the map. The D distance can be computed easily by forming a right triangle composed of the three sides D, H and C as shown in Fig. 7. The side D (the hypotenuse) is the distance from the camera to the location (x, y). The side H is the distance from the camera to the floor, while the side C is the distance from the camera projection on the floor to the location (x, y). Figure 7b shows a top view sketch to illustrate the procedure used to find the distance C. As shown in Fig. 7b, another right triangle arises in the floor plane, with sides of C, X and Y. The distances X and Y are simply the distance from the location T to the center of the camera projection on the floor in the X-direction and Y-direction, respectively. Since the total number of rows and columns of the depth

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

image and the real length and width dimensions of the laboratory are all known quantities, the X and Y distances can be easily computed using the following ratio relationship:

$$X = \frac{m \times L}{N},\ Y = \frac{r \times W}{R} \tag{1}$$

where m and r are the number of columns and rows from the projection of the center of the camera on the floor to the location T, respectively (number of pixels); L is the real length of the map (in meters); N is the total number of columns (in pixels); R is the total number of rows (in pixels) and W is the real width of the map (in meters).

Now, the side C can be computed using a simple Pythagoras equation:

$$C = \sqrt{X^2 + Y^2} \tag{2}$$

Finally, the distance D can be calculated using:

$$D = \sqrt{C^2 + H^2} \tag{3}$$

Combining Eqs. (2) and (3), the distance D can be computed as:

$$D = \sqrt{X^2 + Y^2 + H^2} \tag{4}$$

After finding the distance D_{xy} for all (x, y) locations in the map, the actual depth data P_{xy} is obtained from the camera and compared to the D_{xy} matrix. If a certain index (x, y) in the P_{xy} matrix has a value less than the corresponding index value in the D_{xy} matrix**,** then the location (x, y) can be considered as an obstacle index. The output of the obstacle tracking phase is a binary image where zero pixels (black pixels) represent obstacles, while white pixels represent the free space in the map. Thus:

$$grid\,(x, y) \ = \ \left\{ \begin{array}{c} 0\ if\ P\ (x, y) + h < D\ (x, y) \\ 255\ otherwise \end{array} \right\} \tag{5}$$

where h is the threshold value, $D(x, y)$ is the calculated distance of each (x, y) location from the camera. $P(x, y)$ is actual location returned from the depth camera, $grid$ is the new binary image. The threshold value h is introduced to neglect very small obstacles and to not consider them as obstacles, as the robot can easily drive over them. Empirically, the recommended h value is half of the height of the robot's wheel.

After detecting the obstacles, a safety margin is added around each obstacle. The purpose of this process is that the path planning algorithm treats the robot as a point instead of its actual size. Thus, the size of the added margin should be equal to the size of the robot. The last sub-part in the image processing phase is detecting the starting and the goal points. These points are represented by two labels with two different colors placed in the desired places. To find these points, a range search of these colors is performed instead of a single value search since the value might change due to compression and other impairments.

4. Path Planning

The proposed path planning scheme is described in this section. The proposed path planning technique aims to reduce the computational time and complexity. The idea is that instead of using some well-known algorithms such as A*, D* and PRM on the whole map, the new path planning scheme suggests performing these algorithms only around obstacles. The proposed reduced path planning strategy is divided into four sub-parts. First, a linear line is calculated from the starting point to the goal. In the second phase, the planner detects obstacles that intersect with the linear path. Next, the basis algorithm is performed around each obstacle to find a feasible path around each obstacle. Finally, the planner connects these paths together to establish a free-obstacle path from the starting point to the destination. Figure 8 shows the final path produced by the proposed reduced path planning scheme.

Figure 9 illustrates and simplifies the idea of the proposed reduced scheme. For simplicity and illustration purposes, the map in Fig. 9 is assumed to be 12×12 pixels only. First (Fig. 9a), the coordinates of obstacles are obtained by searching for black pixels (zeros). Next, a linear path is

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

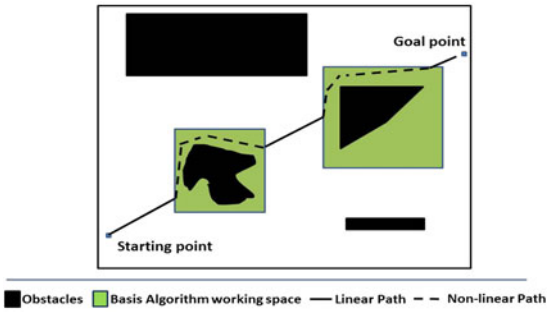

Fig. 8. The reduced path planning scheme.

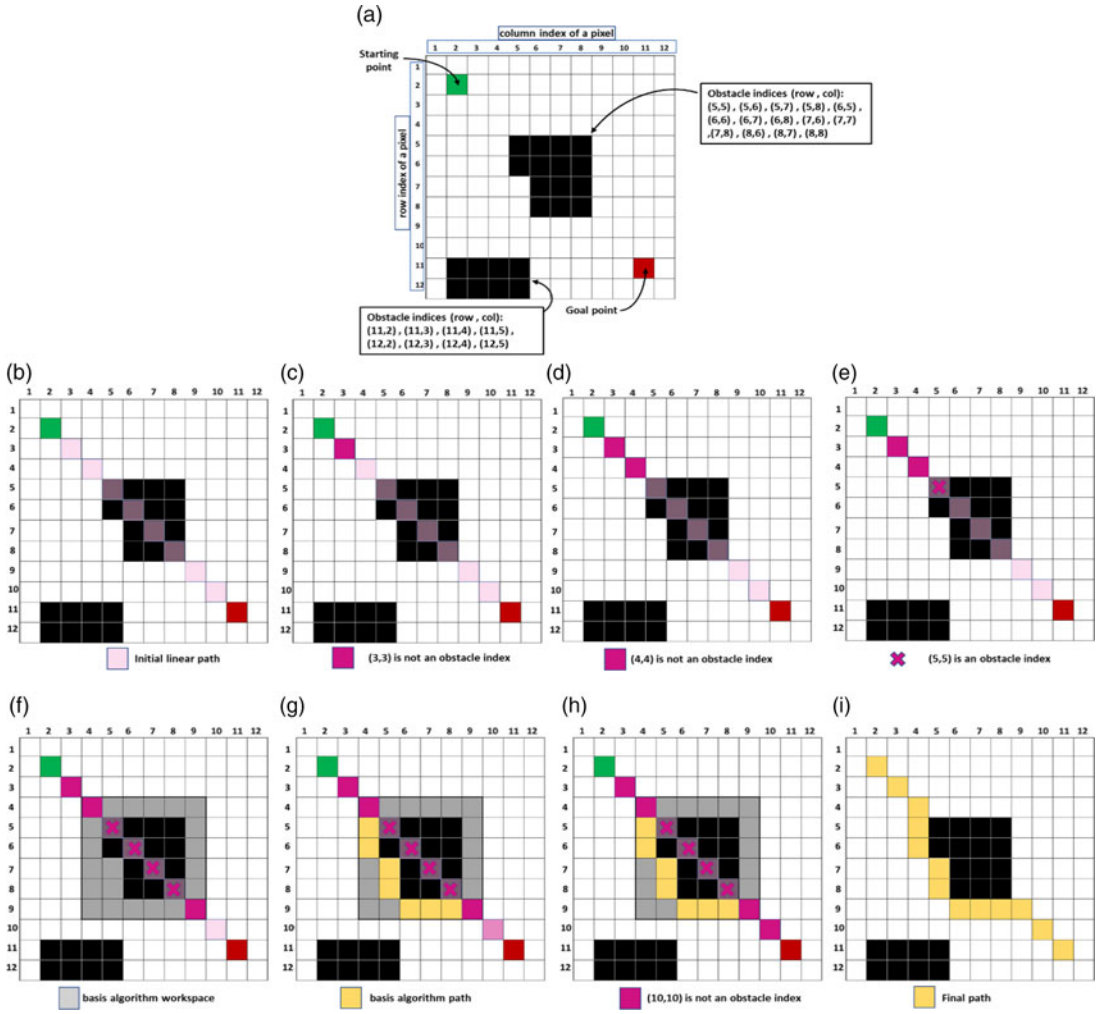

Fig. 9. The proposed reduced path planning strategy step-by-step illustration.

calculated from the starting point to the goal point (Fig. 9b). Then, each pixel is checked whether it is located inside an obstacle or not, by comparing its index with all obstacle indices (Figs. 9c, 9d and 9e). Once a path pixel is detected to be located inside an obstacle, a margin is added around this obstacle. This margin will be the basis path planning algorithm (A*, D*, PRM) working area (Fig. 9f). Next, the basis algorithm will generate a path around the obstacle (Fig. 9g). The basis algorithm path will lead the robot back to the linear path. The planner continues checking whether the linear path pixel is located inside an obstacle (Fig. 9h). Finally, all path parts are combined together

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

to establish a free-obstacle path from the starting point to the goal point (Fig. 9i). More details about different parts of the proposed path planning strategy can be found in the next discussion.

4.1. Linear planner

In Fig. 9b, the proposed reduced scheme starts by calculating a linear path from the starting point to the goal using the following equation:

$$y = m\,(x - x_1) + y_1 \tag{6}$$

where:
x_1, y_1 are the coordinates of the starting point or the goal point.
x is the vector: $\left[x_s,\ x_{s+1},\ x_{s+2},\ \ldots,\ x_g\right]$.
x_s is the x coordinate of the starting point.
x_g is the x coordinate of the goal point.
m is the slope of the linear line as follows:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{7}$$

where x_1 and y_1 are any two points in the slope. The only known points are the starting point and the goal point. The points (x_1, y_1) and (x_2, y_2) can be either the starting point or the goal point. The election is based on the y value of both points, y_2 must be greater than y_1. If the y value of the starting point is greater than the y value of the goal point, then (x_2, y_2) is the starting point and (x_1, y_1) is the goal point. On the other hand, if the y value of the goal point is greater than the y value of the starting point, then (x_2, y_2) is the goal point and (x_1, y_1) is the starting point.

4.2. Obstacle detector

The obstacle detector is responsible to find obstacles that intersect with the linear path. Since the map is a binary image with obstacles with black pixels (zeros) and free-space in white pixels (white), the locations of obstacles can be easily obtained by performing a simple search for the indices of all zero pixels. The obstacle detector checks each point in the linear path, starting from the starting point towards the goal point if it is located inside an obstacle. This is done by comparing the index of the linear path point with every obstacle index. If there is any match between two indices, then the linear path intersects with an obstacle in that index.

Figures (9c, 9d and 9e) illustrate the operation of obstacle detector. The obstacle detector starts the checking process with the pixel after the starting point as in Fig. 9c which has an index of (3,3). The index (3,3) is not an obstacle index. Thus, it is approved to be part of the final path. The obstacle detector moves to the next linear path index (4,4), which is also not an obstacle index. The index (4,4) is also accepted to be in the final path list.

The next index (5,5) is found to be an obstacle index. Once the obstacle detects that an obstacle lies in the linear path, it creates a small margin around the obstacle, and sends the obstacle with the margin to the basis path planner. The margin is added to allow the basis planner to find a path around the obstacle as in Fig. 9g. The first pixel of the linear path located within the margin (index 4,4) is now the starting point for the basis function. Similarly, the last point of the linear path located within the margin (index 9,9) is the goal for the basis function.

The width of the margin is a tuning parameter, increasing the width increases the computational time, but increases the possibility of having a shorter path length. On the other hand, decreasing the width will definitely decrease the computational time, but will decrease the possibility of having a shorter path length. The reason is that by expanding the working area of the basis algorithm, the number of samples N (PRM) or the grid resolution (A*, D*) will increase. This increment will for sure affect negatively on the computational time, but positively on the path efficiency.

These parameters, such as the map (obstacle and margin), new starting point and the new goal point are sent to the basis planner (A*,D*, PRM, etc.) to find the shortest feasible path from the new staring point to the goal point. The obstacle detector does that for each obstacle intersecting with the linear path.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

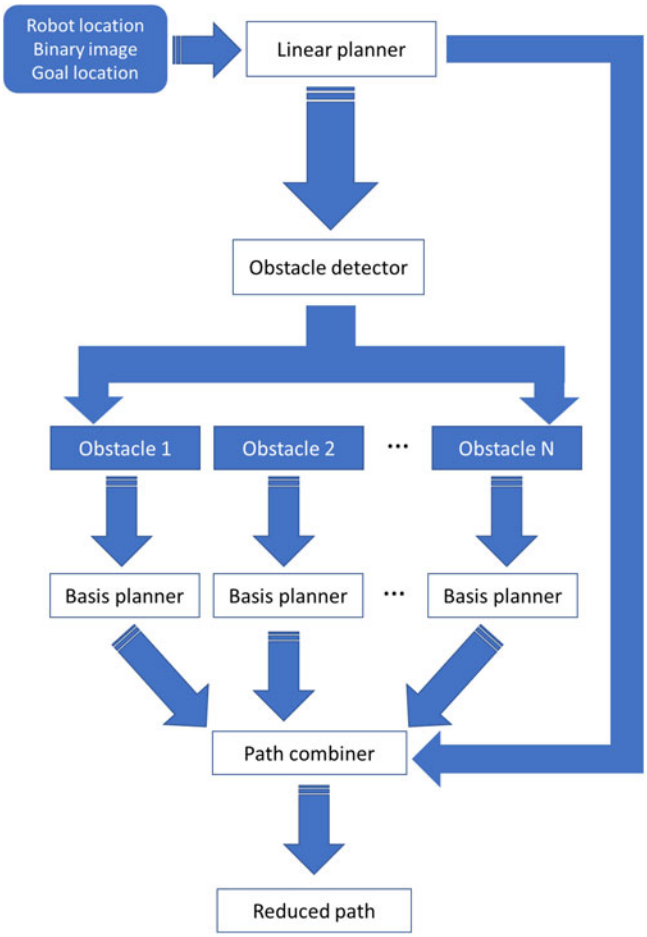

Fig. 10. Reduced-PRM path planner general diagram.

4.3. Basis planner

The basis planner receives the required information, and finds the shortest path around each obstacle. As discussed before, there are some parameters that are critical, and must be chosen carefully. These parameters are the grid size (for A*, D*) and the number of samples N (for PRM). Minimizing these parameters can help reducing the computational time, but that may affect the performance of the system. However, in the proposed scheme, these values will be automatically small since the working area is a much smaller area than the whole map. Moreover, these small values suit the working space, which means that there is no significant adverse effect on the performance.

Figure 9g shows the operation of the basis planner. It is to be noted that the path generated by the basis path planning algorithm will vary based on the basis path planner used. Fig. 9g shows just the illustration, and not the actual path to be generated by the basis algorithm.

4.4. Path combiner

Once all of the paths are generated around each obstacle, the path combiner combines the different linear segments with the non-linear paths generated by the basis planner and produces the final free-obstacle path from the starting point to the goal point. Figure 10 shows a general diagram of the proposed system.

5. Dynamic and Kinematic Modeling

The mobile robot considered in this paper consists of a wheeled mobile robot shown in Fig. 2. The general dynamic equation is described as follows:[23]

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B(q)\tau - A^T(q)\lambda \tag{8}$$

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

where $\tau = [\tau_r, \tau_l]^T \in R^r$ is the input vector and consists of motors' torques τ_r and τ_l, which act on the right and left wheels, $\lambda \in R^m$ is the vector of constraint forces, $M(q) \in R^{n\times n}$ is a symmetric and positive-definite inertia matrix, $C(q, \dot{q}) \in R^{n\times n}$ is the centripetal and Coriolis vector, $G(q) \in R^n$ is the gravitational vector, $B(q) \in R^{n\times r}$ is the input transformation matrix and $A(q) \in R^{m\times n}$ is the matrix associated with the constraints. The kinematic constraints can be denoted as:

$$A(q)\,\dot{q} = 0 \tag{9}$$

$$\dot{x}sin\emptyset - \dot{y}cos\emptyset = 0 \tag{10}$$

When selecting a full rank matrix $S_1(q)$ to be a basis of null space A(q), the constraint equation will be:

$$A(q)\,S_1(q) = 0. \tag{11}$$

where

$$S_1 = \begin{bmatrix} \frac{r}{2L}(Lcos\emptyset + dsin\emptyset) & \frac{r}{2L}(Lcos\emptyset - dsin\emptyset) \\ \frac{r}{2L}(Lsin\emptyset - dcos\emptyset) & \frac{r}{2L}(Lsin\emptyset + dcos\emptyset) \\ \frac{r}{2L} & -\frac{r}{2L} \end{bmatrix}.$$

Therefore, we have:

$$\dot{q} = S_1(q)\,W \tag{12}$$

where $W = \begin{bmatrix} \omega_l & \omega_r \end{bmatrix}^T$ ω_r; ω_l represents the angular velocities of the right and left wheels and $\dot{q} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\emptyset} \end{bmatrix}^T$. If we consider v; ω as the linear angular velocities of the mobile robot, the relation between v; ω and ω_r; ω_l can be explained as:

$$\begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} = \begin{bmatrix} 1/r & L/r \\ 1/r & -L/r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \rightarrow W = HV \tag{13}$$

where $V = \begin{bmatrix} v & \omega \end{bmatrix}^T$. Note that:

$$S_1 H = S = \begin{bmatrix} cos\emptyset & dsin\emptyset \\ sin\emptyset & -dcos\emptyset \\ 0 & 1 \end{bmatrix} \tag{14}$$

The derivative of Eq. (12) gives:

$$\dot{q} = S_1 HW \Rightarrow \dot{q} = SV \Rightarrow \ddot{q} = \dot{S}V + S\dot{V} \tag{15}$$

6. Control Law Design

Note that the objective is to track a reference trajectory by the mobile platform. In this section, the velocity control based on kinematic model is designed to develop the desired forward and rotational velocities. The desired position is $q_d = \begin{bmatrix} x_d & y_d & \phi_d \end{bmatrix}$ and the desired velocity is denoted by $V_d = \begin{bmatrix} v_d & \omega_d \end{bmatrix}^T$. Using the Kanayama transformation,[24] the tracking errors are obtained as follows:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{\phi} \end{bmatrix} = \begin{bmatrix} cos\phi & sin\phi & 0 \\ -sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \phi_d - \phi \end{bmatrix} \tag{16}$$

and the related error dynamics are expressed as follows:

$$\begin{aligned} \dot{\tilde{x}} &= \omega\tilde{y} - v + v_d cos\emptyset \\ \dot{\tilde{y}} &= -\omega\tilde{x} + v_d sin\emptyset \\ \dot{\tilde{\phi}} &= \omega_d - \omega \end{aligned} \tag{17}$$

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

Proposition: when selecting the velocity control law given in (18), the error dynamics (17) are asymptotically stable.

$$V(t) = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} k_1\tilde{x} + v_d cos\tilde{\phi} \\ \omega_d + k_2 v_d \tilde{y} + k_3 v_d sin\tilde{\phi} \end{bmatrix} \tag{18}$$

where $k_1 > 0; k_2 > 0; k_3 > 0$ are the controller gains.

To prove Proposition 2, let us define the following positive Lyapunov function:

$$W = \frac{1}{2}\tilde{x}^2 + \frac{1}{2}\tilde{y}^2 + \frac{1 - cos\tilde{\phi}}{k_1} \tag{19}$$

The time derivative of W is given as follows

$$\dot{W}(t) = \tilde{x}\dot{\tilde{x}} + \tilde{y}\dot{\tilde{y}} + \dot{\tilde{\phi}}\frac{sin\tilde{\phi}}{k_1}$$

$$= \tilde{x}\,(\omega\tilde{y} - v + v_d cos\emptyset) + \tilde{y}\,(-\omega\tilde{x} + v_d sin\emptyset) + (\omega_d - \omega)\,\frac{sin\tilde{\phi}}{k_1}$$

$$= -v\tilde{x} - \omega\frac{sin\tilde{\phi}}{k_1} + \tilde{x}v_d cos\emptyset + \tilde{y}v_d sin\emptyset + \omega_d\frac{sin\tilde{\phi}}{k_1}$$

Using the control law (18), the time derivative of the Lyapunov function becomes:

$$\dot{W}(t) = -k_1\tilde{x}^2 \tag{20}$$

The time derivative of $W(t)$ is negative because k_1 is positive gain. By (20), W is monotonously decreasing. Therefore, $\tilde{x} \in L_\infty$, $\tilde{y} \in L_\infty$ and $\tilde{\phi} \in L_\infty$, *and* $\tilde{x} \in L_2$. According to (17), we know that $\dot{\tilde{x}}$, $\dot{\tilde{y}}$ *and* $\dot{\tilde{\phi}}$ are bounded and therefore $\tilde{x}$, $\tilde{y}$ and $\tilde{\phi}$ are uniformly continuous. Then, using Barbalat's theorem,[22] we can conclude that $\tilde{x} \to 0$, $\tilde{y} \to 0$ and $\tilde{\phi} \to 0\, as\, t \to \infty$ and the error dynamics are asymptotically stable.

7. Experimental Comparative Results

The experiments were done in the laboratory where an Asus Xion pro depth camera was mounted on the ceiling. The robotic system used in these experiments was the Pioneer P3-DX shown in Fig. 11. The experimental setup is shown in Fig. 12.

The parameters used in the following experiments are shown in Table I. To test the efficiency of the proposed path planning strategy in terms of the computational time, the proposed scheme was tested with three different scenarios. In the first scenario, there are no obstacles "intersecting" with the linear path from the starting point to the goal point (even though obstacles are presented in the scene). In the second scenario, one obstacle is intersecting with the linear path. Finally, in the third scenario, two obstacles are intersecting with the linear path. Each scenario has two different configurations by changing the starting and end points. Figure 13 shows the RGB and depth images captured by the ASUS Xtion Pro camera, while Fig. 14 shows the three different scenarios with different configurations.

In the following experiments, A*, D* and PRM are chosen to be the basis algorithm to work on. For A* and D*, a grid of 31×41 (1271 total points) was chosen for a 480×640 image. PRM was configured to have 1000 samples (N), and a value of 50 is chosen to be the connection distance (D). Since the workspace in the reduced area is smaller than the whole map, the proposed reduced strategy allows theses parameters to be minimized. As explained earlier, the basis algorithm workspace is composed of the obstacle and the surrounding margin. Thus, this area varies based on the obstacle size and the margin width. To ensure having a fair comparison, N, D and the grid values were scaled down linearly so that they are related to their workspace size.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

Table I. Experiment parameters.

Parameters	**values**
Image Processing:	
H [Threshold] (m)	3.8
Ls (R,G,B)	255,255,0 yellow)
Lg (R,G,B)	0,128,0 (green)
Robot:	
m_c (kg)	7.4
m_ω (kg)	0.8
L (m)	0.19
r (m)	0.0975
d(m)	0.1
I_c (kg m^2)	0.020
I_m (kg m^2)	0.00475

Fig. 11. Pioneer P3-DX.

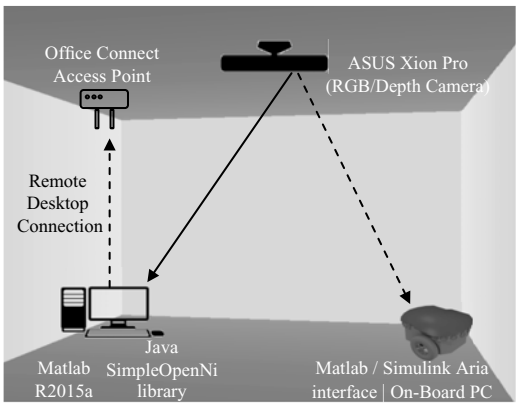

Fig. 12. Experimental setup.

Figure 15 shows the path calculated from both methods for the third scenario, including the traditional algorithm and the reduced and proposed strategies.

7.1. Computational time comparison

A comparison of the time execution between the reduced and original versions of A*, D* and PRM is shown in Table II. The experiment was done under different lighting conditions.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

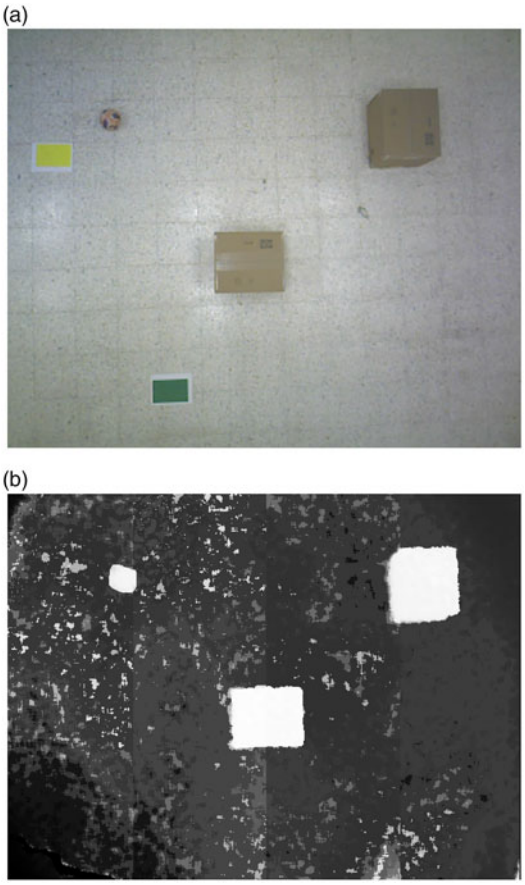

Fig. 13. (a) RGB image captured by the ASUS Xtion Pro. (b) Raw depth image captured by the ASUS Xtion Pro.

Table II clearly proves the efficiency of the proposed path planning scheme in terms of the computational time required to calculate the free-obstacle path to the goal point. It is obvious that scenario 1, where no obstacles are intersecting with the linear path, will have the greatest result, since all that is required is computing a simple linear path using Eq. (6). Thus, there is no need to define grids, distribute points or establish lines between points, as all these processes consume time and effort. Figure 16 illustrates how it is inefficient to use PRM in scenario 1. Another important note, when comparing scenario 2 with scenario 3, is that the computational time increases by only 0.029 seconds for reduced A*, 0.062 for reduced D* and PRM. This indicates that even if we increase the number of obstacles that intersect with the linear path, the reduced scheme will still be much faster than the original path planning algorithms. Another noticeable fact is that the proposed scheme was not affected under different environmental (lighting) conditions. The average execution time results were independent of the lighting condition. That is because the proposed system relies on the depth information (infrared signals) in locating obstacles, which is robust against different lighting conditions.

The improvement percentage was calculated using the following equation:

$$\mathrm{Ip} = \left(1 - \frac{t_r}{t_c}\right) \times 100 \tag{21}$$

where:
t_r is the computational time by the reduced scheme.
t_c is the computational time by the classical scheme.

7.2. Path length comparison

As stated in the Introduction section, there is a trade-off between computational time and the path efficiency. Path efficiency is studied in this paper from two aspects, path length and trackability. Since

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

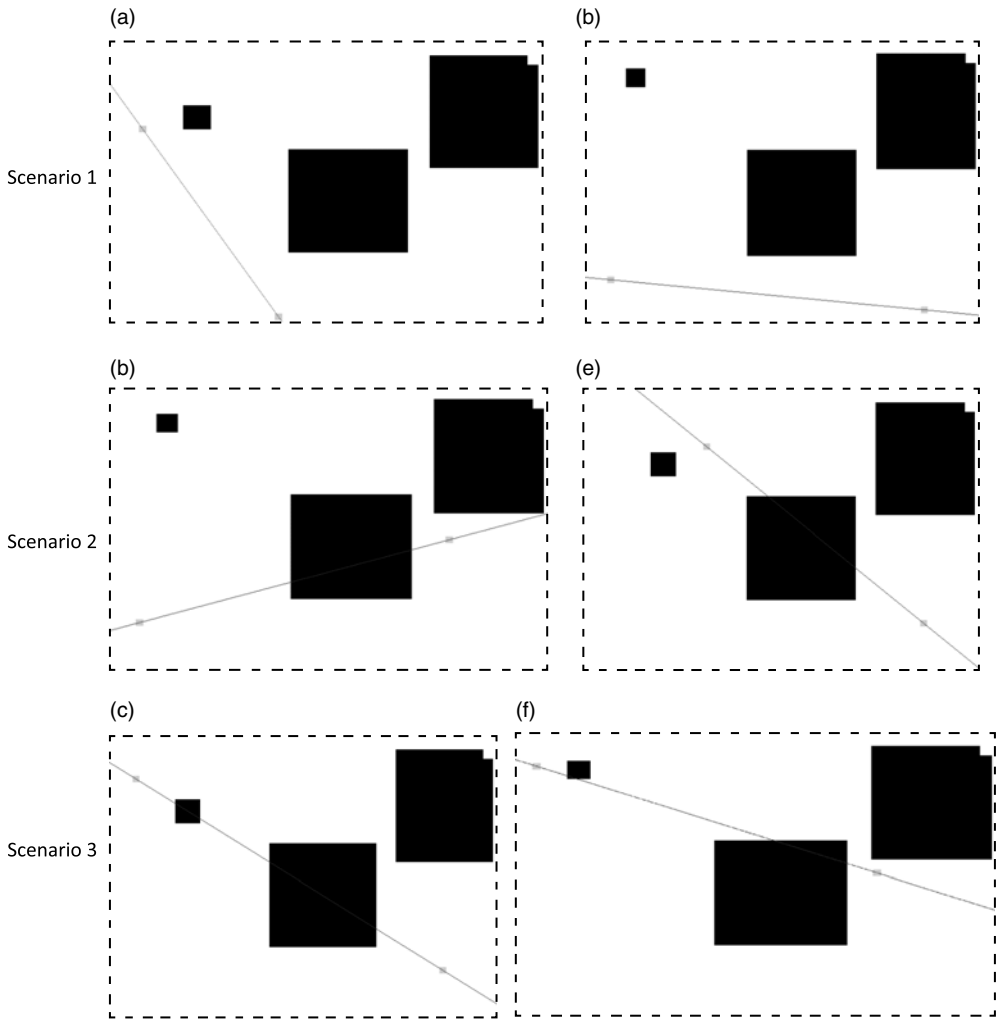

Fig. 14. Binary map after processing – configurations (a) and (b) represent scenario 1 (no obstacles intersecting with the linear path). (c) and (d) configurations represent scenario 2 (1 obstacle intersecting with the linear path). Configurations (e) and (f) represent scenario 3 (2 obstacles intersecting with the linear path).

the proposed reduced scheme has shown a huge improvement in the time execution calculations, it is expected that the proposed reduced path will be less efficient in terms of length and trackability. Table III shows a comparison between classical and reduced scheme in terms of path length calculations for scenario 2.

Table III shows that the proposed reduced paths perform better than classical methods for scenario 2. Even though the proposed reduced scheme focused on improving the computational time, it could in some cases break the trade-off between path efficiency and computational time. Another advantage of the reduced paths is that the differences between the lengths of experimental paths and the lengths of the simulated paths of the reduced scheme are less than the equivalent differences in the original scheme. For example, the path length of the reduced PRM is 4.3397 m in the real-life experiment while it is 4.2010 m in simulations, with a difference percentage of 3.30%. On the other hand, the path length of the original PRM is 4.2772 m in the real-life experiment while it is 3.4011 m in simulations, with a difference percentage of 25.76%. That is an indication that the reduced scheme can be controlled easier with minimal errors between the real path and the desired simulated path. To be noted here that the kinematic controller described in Section 6 was used to obtain the results in Table III. More details about the path trackability tests will be presented in the next sub-section.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

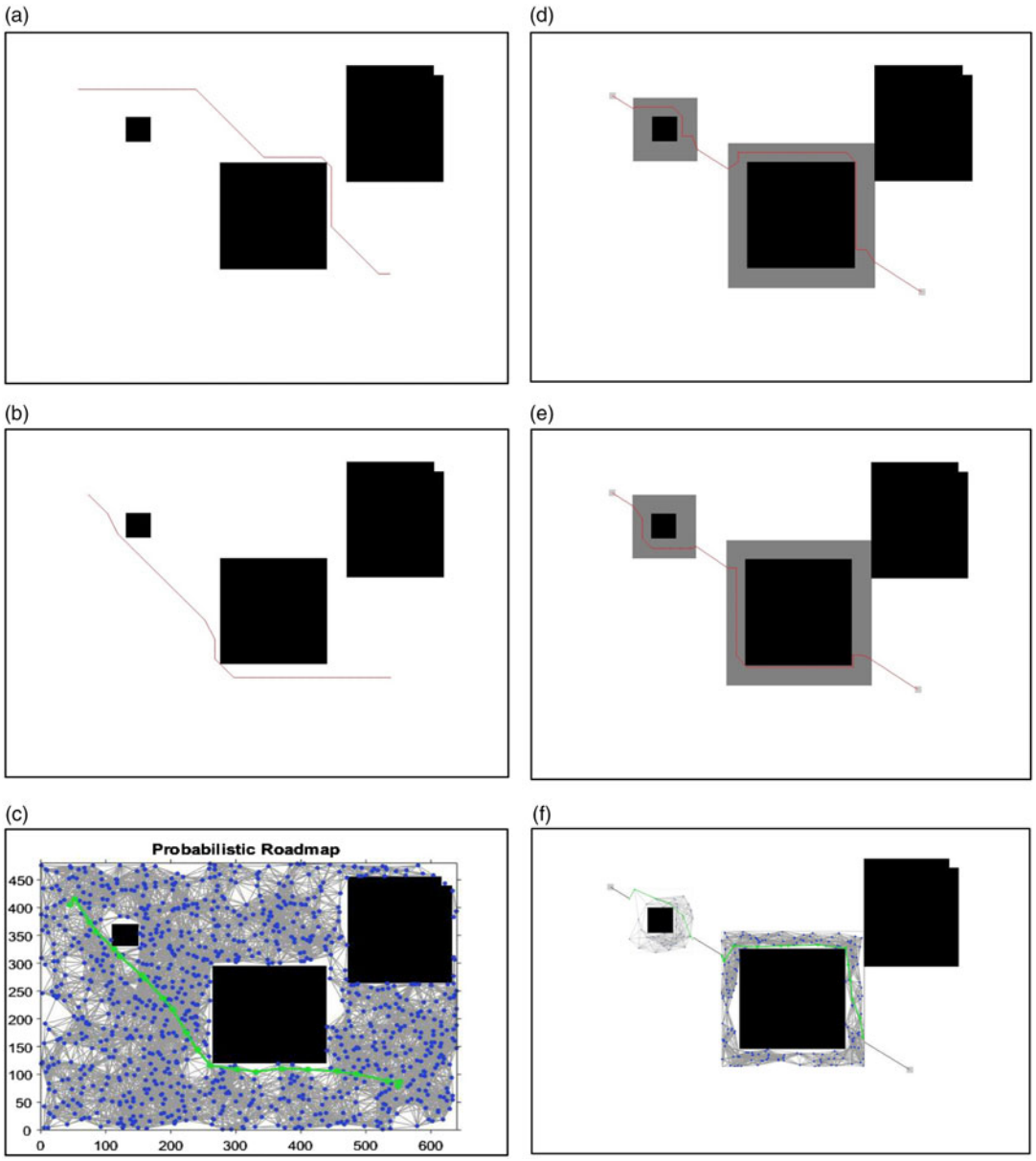

Fig. 15. Path planning with the basis and proposed algorithms. (a) Path calculated by original A*. (b) Path calculated by reduced A*. (c) Path calculated by original D*. (d) Path calculated by reduced D*. (e) Path calculated by original PRM. (f) Path calculated by reduced PRM.

7.3. Path trackability test

Next, the reduced PRM path – scenario 2 – is chosen to be tested with our kinematic controller and two intelligent controllers based on Artificial Neural Networks (ANN), which are the Multi-Layer Perceptron (MLP) and the Non-linear Auto-Regressive with Exogenous inputs (NARX). A popular approach for function approximations is by using ANN. An ANN is trained to represent the forward dynamics of a system. Training is done by minimizing the prediction error between the system output and the neural network output. Using a two-layered network with a sigmoid activation function in the hidden layer, and a linear activation function in the output layer is a good universal approximator.[25] Thus, this MLP structure is going to be used as the first NN controller.

Another approach being popular in the system identification field is the usage of Recurrent Neural Networks (RNN). An RNN is a category of ANN where a neuron can receive a previous prediction and include it along with the input in the calculations. This is done by having some feedback connections between neurons. Thus, RNN can capture temporal plant behavior for a time sequence.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

Table II. Comparison between Reduced-PRM and the standard PRM.

Lighting condition		Average Execution time (sec)						Improvement percentage		
		Normal		Dark		Bright				
Scenario	Method	Reduced	Original	Reduced	Original	Reduced	Original	Normal	Dark	Bright
Scenario 1	A*	0.243	0.589	0.223	0.569	0.233	0.609	58.7	60.8	61.7
(No obstacles intersecting with the linear path).	D*	0.243	1.296	0.233	1.286	0.233	1.286	58.7	81.9	81.9
Configuration a Figure 14-a	PRM	0.243	2.074	0.223	2.054	0.233	2.054	81.3	89.1	88.7
Scenario 1	A*	0.280	0.556	0.270	0.546	0.280	0.556	88.3	50.5	49.6
(No obstacles intersecting with the linear path).	D*	0.280	1.37	0.270	1.360	0.280	1.370	49.6	80.1	79.6
Configuration b Figure 14-b	PRM	0.280	3.75	0.280	3.750	0.270	3.740	79.6	92.5	92.8
Scenario 2	A*	0.304	0.649	0.314	0.659	0.284	0.629	92.5	52.4	54.8
(1 obstacle intersecting with the linear path).	D*	0.458	1.262	0.448	1.252	0.448	1.252	53.2	64.2	64.2
Configuration c Figure 14-c	PRM	0.558	2.149	0.558	2.149	0.578	2.169	63.7	74.0	73.4
Scenario 2	A*	0.331	0.672	0.321	0.662	0.331	0.672	74.0	51.5	50.7
(1 obstacle intersecting with the linear path).	D*	0.480	1.30	0.460	1.280	0.490	1.310	50.7	64.1	62.6
Configuration d Figure 14-d	PRM	0.80	3.77	0.820	3.790	0.810	3.780	63.1	78.4	78.6
Scenario 3	A*	0.333	0.601	0.333	0.601	0.353	0.621	78.8	44.6	43.2
(2 obstacles intersecting with the linear path).	D*	0.52	1.26	0.540	1.280	0.530	1.270	44.6	57.8	58.3
Configuration e Figure 14-e	PRM	0.62	2.27	0.620	2.270	0.610	2.260	58.7	72.7	73.0
Scenario 3	A*	0.354	0.685	0.344	0.675	0.354	0.685	72.7	49.0	48.3
(2 obstacles intersecting with the linear path).	D*	0.54	1.316	0.520	1.296	0.550	1.326	48.3	59.9	58.5
Configuration f Figure 14-f	PRM	0.90	3.71	0.920	3.730	0.890	3.700	59.0	75.3	75.9

Table III. Comparison between proposed reduced scheme and the classical schemes in the path length.

Scenario	Method		Path length – experiment (meters)	Improvement percentage	Path length – simulation (meters)	Simulation-Experiment error (percentage)
2	A*	Reduced	4.1611	20.07%	4.4343	6.16
(1 obstacle		Classical	5.2061		4.5473	14.49
intersecting	D*	Reduced	4.1811	3.725%	3.9951	4.66
with the linear		Classical	4.3429		4.2937	1.15
path).	PRM	Reduced	4.3397	−1.461%	4.2010	3.30
Figure 14		Classical	4.2772		3.4011	25.76

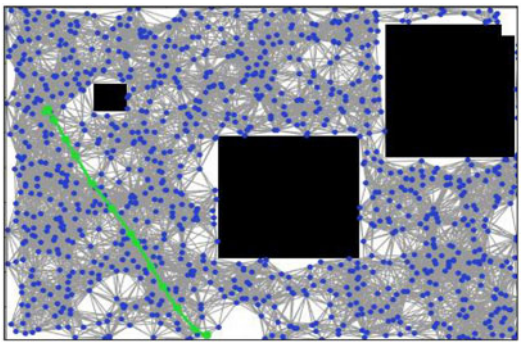

Fig. 16. Using PRM with scenario 1.

The NARX network is an RNN. The architecture contains some feedback connections that surround several layers of the network. The NARX model is being widely used in time series modeling.[26] The NARX network can be defined as:

$$y(t) = f\left(\left(y(t-1), y(t-2), \ldots, y\left(t-n_y\right), u(t-1), u(t-2), \ldots, u\left(t-n_u\right)\right)\right) \tag{22}$$

It is clear from Eq. (22) that the prediction is highly dependent on the previous predictions using the feedback connections. The fact that distinguishes NARX networks in the RNN family is that the final output is fed to be the input of the network.

The reference trajectory is chosen as $q_{rd} = \left[x_r \; y_r \; \emptyset_r\right]^T$ then $\dot{x}_r = V_r cos\emptyset_r$; $\dot{y}_r = V_r sin\emptyset_r$, $\dot{\emptyset}_r = \omega_r$. where the linear velocity and the reference angular velocity are chosen as $V_r = 0.2\, m/s$ and $\omega_r = 0.1\, rad/s$, respectively.

The tracking experimental results obtained are shown in Figs. 17, 18 and 19 for the kinematic controller, the MLP controller and the NARX controller, respectively.

Regarding the tracking results, the three controllers were able to perfectly track the path with minimal errors. Figures 17, 18 and 19 (a, b, and c) show the tracking in the x, y, and phi, respectively. This good tracking is confirmed by Figs. 17, 18 and 19 (d, e, and f), which illustrate the tracking error. The kinematic controller has achieved the best tracking performance. The errors have small values that do not exceed 0.14 m for x-position, 0.05 for y-position and phi. Moreover, Fig. 17 shows that in each sudden change in the path, the kinematic controller successfully deals with that sudden change and makes the error approach zero. For the NN-based controllers, the NARX controller has achieved slightly a better tracking performance than MLP. That is because NARX networks are more suitable for time series signals. The feedback connections between neurons have allowed this NN model to mimic the temporal dynamic behavior of a kinematic controller.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

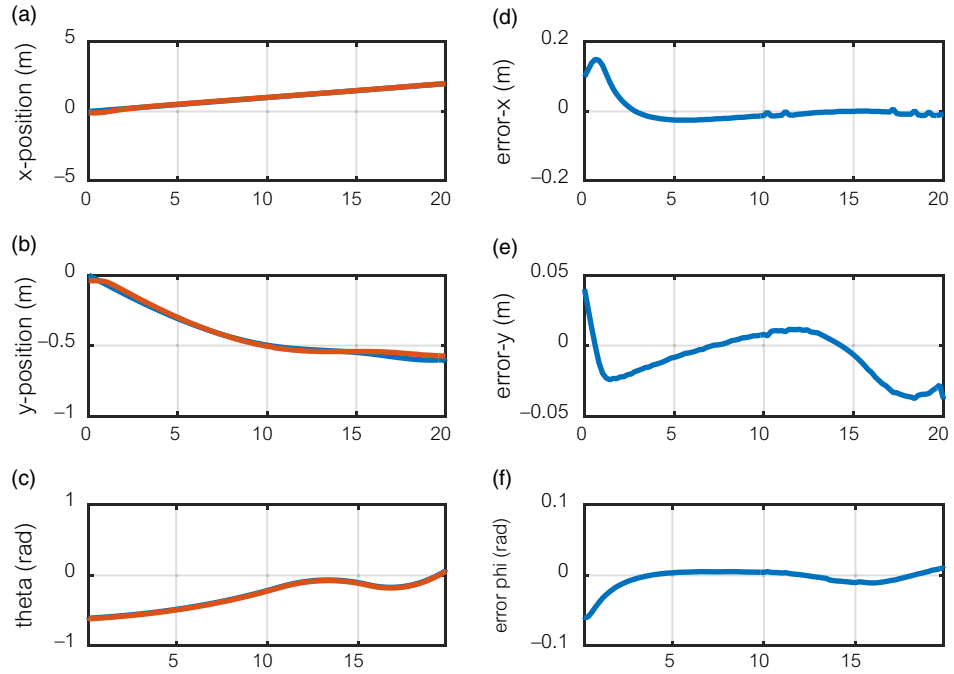

Fig. 17. Kinematic control of the platform subsystem: (a) Tracking trajectory of x-position. (b) Tracking trajectory of y-position. (c) Tracking trajectory of φ-direction. (d) Tracking error of x-position. (e) Tracking error of y-position. (f) Tracking error of φ-direction.

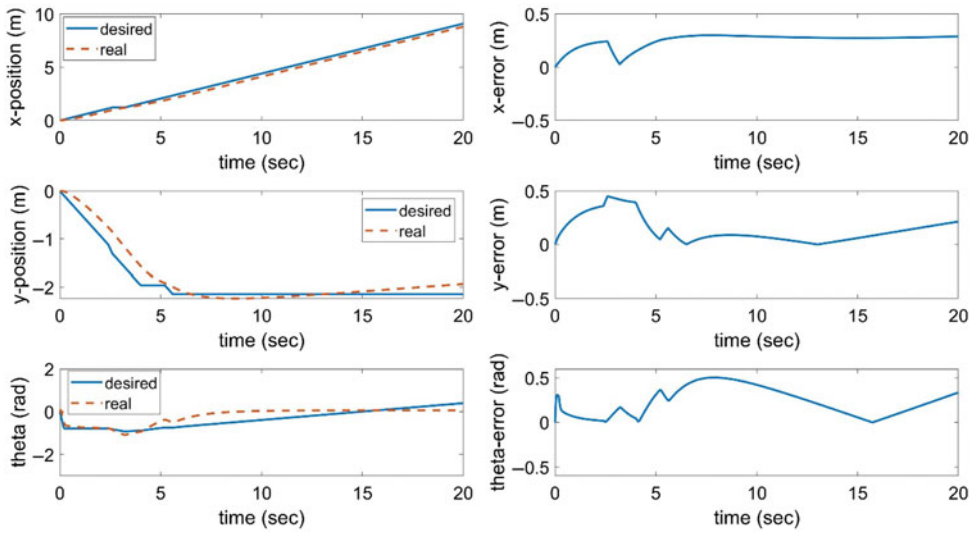

Fig. 18. Intelligent control of the platform subsystem using MLP: (a) Tracking trajectory of x-position. (b) Tracking trajectory of y-position. (c) Tracking trajectory of φ-direction. (d) Tracking error of x-position. (e) Tracking error of y-position. (f) Tracking error of φ-direction.

8. Conclusion

This paper introduced a path planning strategy that is efficient for real-time scenarios. The novel strategy can be applied on any path planning algorithm to reduce the computational time required to calculate the path. The path planning strategy was tested on three control systems for differential wheeled mobile robot, namely a kinematic controller, which was proved by the Lyapunov approach, an MLP-based controller and an NARX-based controller. First, vision and image processing algorithms were used to collect and analyze information. Then, a free-obstacle path was calculated by using the proposed reduced-scheme path planner. This planner was the key feature of the proposed system due to its simplicity and fast execution. Finally, three different controllers were developed to

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

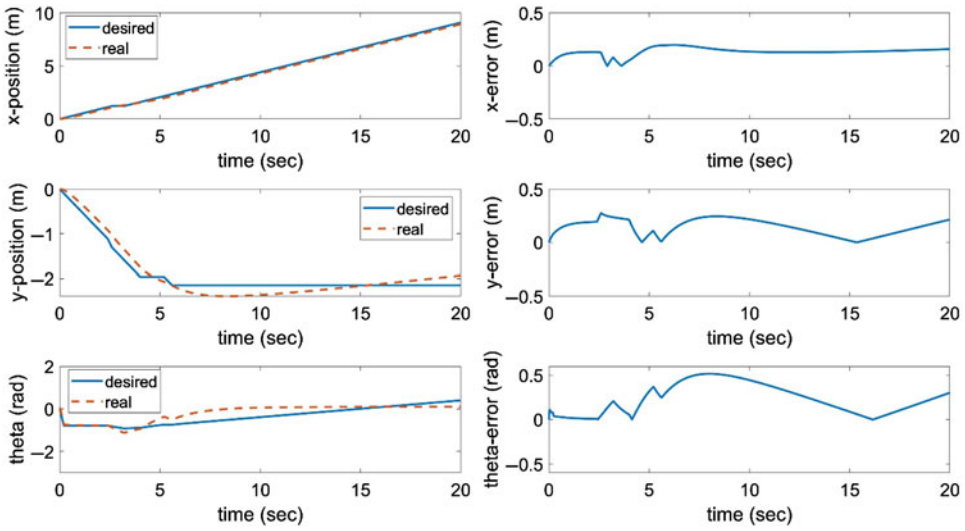

Fig. 19. Intelligent control of the platform subsystem using NARX: (a) Tracking trajectory of x-position. (b) Tracking trajectory of y-position. (c) Tracking trajectory of φ-direction. (d) Tracking error of x-position. (e) Tracking error of y-position. (f) Tracking error of φ-direction.

follow the path. Experimental comparisons have proved that the path planning technique was able to deduct the computational time needed by the corresponding classical path planning algorithm without significant effects on the path quality. In some cases, it actually improved the quality as well represented by the path length and the trackability. As a future work, the kinematic controller will be upgraded to a dynamic intelligent controller that can directly send torque commands to the robot's motors.

References

1. D. Held, J. Levinson, S. Thrun and S. Savarese, "Robust real-time tracking combining 3D shape, color, and motion," *Int. J. Rob. Res.* **35**(1–3), 30–49 (2016).
2. S. Haddadin, R. Weitschat, F. Huber, M. C. Özparpucu, N. Mansfeld and A. Albu-Schäffer, "Optimal control for viscoelastic robots and its generalization in real-time," **In**: *Robotics Research* (Springer, 2016) pp. 131–148.
3. H. Kim, S. Leutenegger and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," **In**: *European Conference on Computer Vision (ECCV)*, no. 6 (Springer, Cham, 2016) pp. 349–364.
4. V. Vaquero Gómez, E. Repiso Polo, A. Sanfeliu Cortés, J. Vissers and M. Kwakkernaat, "Low Cost, Robust and Real Time System for Detecting and Tracking Moving Objects to Automate Cargo Handling in Port Terminals," *Robot 2015: Second Iberian Robotics Conference*, Lisbon, Portugal (Springer, 2015) pp. 491–502.
5. B. Patle, D. Parhi, A. Jagadeesh and O. Sahu, "Real time navigation approach for mobile robot," *JCP* **12**(2), 135–142 (2017).
6. K. Wang, Y. Liu and L. Li, "Vision-based Tracking Control of Nonholonomic Mobile Robots without Position Measurement," *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany (IEEE, 2013) pp. 5265–5270.
7. M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access* **2**, 56–77 (2014).
8. C. L. Nielsen and L. E. Kavraki, "A Two Level Fuzzy PRM for Manipulation Planning," **In**: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2000)*, vol. **3**, (IEEE, 2000).
9. J. Denny, K. Shi and N. M. Amato, "Lazy Toggle PRM: A Single-query Approach to Motion Planning," *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany (IEEE, 2013).
10. B. K. Patle, D. R. Parhi, A. Jagadeesh and S. K. Kashyap, "Matrix-binary codes based genetic algorithm for path planning of mobile robot," *Comput. Electr. Eng.* **67**, 708–728 (2018).
11. A. S. Wallack, L. Lifeng and X. Ye, "System and method for robust calibration between a machine vision system and a robot," ed: Google Patents (2016).
12. X. C. Chen, Z. Q. Cao, Y. Q. Yang and C. Zhou, "Vision-based fuzzy controller for quadrotor tracking a ground target," **In**: *Advances in Science and Technology*, vol. **101**, (Trans Tech Publications, 2017) pp. 121–126.

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press

13. Z. Li, C. Yang, C.-Y. Su, J. Deng and W. Zhang, "Vision-based model predictive control for steering of a nonholonomic mobile robot," *IEEE Trans. Control Syst. Technol.* **24**(2), 553–564 (2016).
14. J. Leitner, S. Harding, M. Frank, A. Förster and J. Schmidhuber, "An integrated, modular framework for computer vision and cognitive robotics research (icvision)," **In**: *BICA* (Springer, Berlin, Heidelberg, 2012) pp. 205–210.
15. H. Choset, "Coverage for robotics-A survey of recent results," *Ann. Math. Artif. Intell.* **31**(1), 113–126 (2001).
16. Y. Singh, S. Sharma, D. Hatton and R. Sutton, "Optimal path planning of unmanned surface vehicles," *Indian J. Geo. Mar. Sci*. **47**(7), 1325–1334 (2018).
17. 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*. **96**, 59–69 (2014).
18. A. Le, V. Prabakaran, V. Sivanantham and R. Mohan, "Modified a-star algorithm for efficient coverage path planning in Tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors* **18**(8), 2585 (2018).
19. X. Liang, G. Meng, Y. Xu and H. Luo, "A geometrical path planning method for unmanned aerial vehicle in 2D/3D complex environment," *Intell. Serv. Rob*. **11**(3), 301–312 (2018).
20. 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," arXiv preprint arXiv:1404.2334 (2014).
21. W. G. Aguilar and S. G. Morales, "3D environment mapping using the Kinect V2 and path planning based on RRT algorithms," *Electronics* **5**(4), 70 (2016).
22. R. Fareh, T. Rabie and M. Baziyad, "Tracking Control for Wheeled Mobile Robot Using RGBD Sensor," *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, St. Paul's Bay, Malta (IEEE, 2017).
23. Y. Kanayama, Y. Kimura, F. Miyazaki and T. Noguchi, "A Stable Tracking Control Method for an Autonomous Mobile Robot," *1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, USA (IEEE, 1990) pp. 384–389.
24. J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, vol. 1 (Prentice Hall, Englewood Cliffs, NJ, 1991).
25. K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks* **2**(5), 359–366 (1989).
26. W. J. Lee, J. Na, K. Kim, C.-J. Lee, Y. Lee and J. M. Lee, "NARX modeling for real-time optimization of air and gas compression systems in chemical processes," *Comput. Chem. Eng.* **115**, 262–274 (2018).

https://doi.org/10.1017/S0263574719000572 Published online by Cambridge University Press