Chapter 12
Robot Path-Planning Research Applications in Static and Dynamic Environments

Firas A. Raheem, Safanah M. Raafat, and Shaymaa M. Mahdi

Abstract The purpose of path planning is primarily to find a set of continuous motions for moving robots from an initial point and ending at the goal point in Cartesian or configuration space. Solving the robot arm path-planning problem can be considered as one of the most significant aspects of robot navigation to guarantee the best path free of obstacle collisions before tracking. The complexity of the robot environment is one of the most important characteristics of robots, especially in environments that contain static and/or dynamic obstacles such as robots, human operators, and other moving objects. The problem of path planning is subjected to the constraints of finding the possible path. The constructed path satisfies specific optimization criteria if the environment is known or has a property of intelligence if the environment is unknown. Many recent path-planning approaches have optimization and intelligent properties when the robot environment is partially unknown. In addition to intelligent methods that are used for planning a successful path, recent advances in path-planning methodologies involve using both heuristic and meta-heuristic with artificial intelligence concepts with relevance across various field types with great application in calculated optimization of sustainability and resource use.

Keywords Robot arm path planning · Robot environment · Optimization · Intelligent methods · Metaheuristic · Dynamic · Static

F. A. Raheem (✉) · S. M. Raafat · S. M. Mahdi
Automation and Robotics Research Unit, Control and Systems Engineering Department, University of Technology-Iraq, Baghdad, Iraq
e-mail: firas.a.raheem@uotechnology.edu.iq; safanah.m.raafat@uotechnology.edu.iq; shaymaa.m.mahdi@uotechnology.edu.iq

J. N. Furze et al. (eds.), *Earth Systems Protection and Sustainability*,
https://doi.org/10.1007/978-3-030-85829-2_12

12.1 Introduction to Path Planning

In any robot workspace, the solution of the motion control problem includes firstly finding a suitable path which can be indicated as computing the movement of any robot joint continuously, which enables a collision-free robot motion with all located obstacles in its environment from the start point or configuration to the goal point or goal configuration at any chosen point of its motion path.

Planning a path requires a map depicting the robot and its surrounding environment, so that the robot knows its position on the map and can determine its location using the map to avoid transitory obstacles on its path. How to find the robot location and how to take into consideration the undefined location information will be the core of the rest of this chapter.

In a cluttered and crowded environment with obstacles, path-planning solutions for manipulators and autonomous robots and the determination of optimum path are considered to be attractive research topics. Most research focuses on the working conditions of the robot and the application of path-finding steps in fully known static or dynamic environments. In fractionally known or totally undiscovered, uncharted, unexplored environments, the steps of planning a path are difficult and cannot be applied directly. The robot encounters uncertainties, reflected in how it makes decisions to continue moving through unexplored and explored areas of the robot workspace. These types of planning methods are related to known areas and operate in a work zone where the target point is placed. Moreover, we will discuss the type of data structure required to support outstanding algorithms for path planning.

Before considering path-planning research, it is useful to resolve the problem of solving path-planning issues from a human perspective – how people plan paths.

From the perspective of investigating path-planning issues, it should be noted that the robot path planner must obtain knowledge of the strategy used and the related features that must be considered when adjusting the route-planning algorithm (Hameed 2019).

12.2 Path-Planning Concept with Obstacle Avoidance

Planning a path is a subtask of general motion planning tasks. Finding a collision-free path from the initial configuration to the target configuration is a purely geometric challenge. Obstacle avoidance means that the robot moves in its workspace with its diverse mobility and controllability without colliding with environmental obstacles. The task of planning paths and avoiding obstacles is more difficult for manipulators than for mobile robots; the problem is not only to find the path of the end effector but also to avoid collisions with manipulator links. Path planning requires information of the size of the robot, first and final robot configurations, work environment, and static stationary obstacles.

Planning a trajectory can be considered independently from path planning. Trajectory planning is defined as the problem of determining the path as a function of time, which means determining the velocity and the acceleration of the robot at each point (Hussain 2017).

12.3 Path-Planning Classifications

The solution of robot path-planning task can be divided into five categories: (i) planning type, (ii) planning time, (iii) environment type, (iv) obstacle motion behavior, and (v) according to the robot space used for trajectory computations. Category (i) includes global path planning and local path planning. Through global path planning, the robot knows all the information in the work area prior to planning and can directly determine its path without collision. Global path planning is sometimes referred to as a deliberative method. Planning based on robot sensors' information measured at each sample of motion is referred to as local path planning where the environment is incomplete or unknown and the robot uses feedback from sensors to find the path. Category (ii) includes planning a path depending on the time to execute the robot motion and can be divided into online planning and offline planning. Online real-time planning affects robot movement in systematic environments. Offline planning provides a complete path before the robot's motion. Category (iii) includes path planning subject to the environment type and can be classified into known, totally unknown, and partially known environments. Category (iv) depends on the obstacle motion behavior and can be divided into static stationary obstacles and dynamic or movable obstacles (Hussain 2017). Category (v) divides the robot path and trajectory computations according to Cartesian space, joint space, and configuration space.

In robotics, trajectory planning strategies are computed and combined together according to the different situations of the robot and its environment that match the above classifications (Hameed 2019); these classifications are described in detail in the rest of this section.

12.3.1 Path Planning According to Obstacles

- Static obstacles
- Moving or dynamic obstacles

12.3.2 Path Planning Depends on Environment and Obstacle Types

- Known static environment
- Known dynamic environment
- Partially known static and dynamic environment
- Unknown static environment
- Unknown dynamic environment

12.3.3 Types of Planning Methodologies

- Global planning: This methodology assumes that the total information that represents the robot environment is known and available.
- Local planning: This methodology uses only a part of the global model to find the robotic path and control the motion, which is a disadvantage due to the loss of path optimization cases. The local plan is effectively suffering from local minima.

A preferred key point of the local and global methods is that local planning calculations are not as complex as global calculations. This is especially important when the world model is updated with each sample due to sensor collision data measurements which pose a known problem to autonomous robots.

12.3.4 Path Planning According to Robot Space

- Cartesian space
- Joint space
- Configuration space (C-space)

12.3.5 Path According to Planning Time

- Online planning
- Offline planning

Path-planning schema of classifications and approaches is shown in Fig. 12.1.

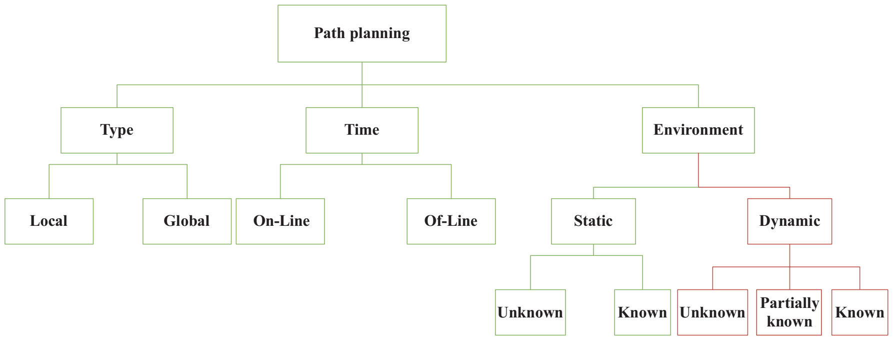

Fig. 12.1 Classifications and approaches of path planning

12.4 Cartesian Space Path Planning

Robot workspace or Cartesian space is the space in which the robot operates. Generally, the work area is the part of the environment in which the robot executes a motion plan. The planning problem in Cartesian workspace is the movement of an end effector of the manipulator on a specific path and specific trajectory. The advantage of path planning in Cartesian workspace is that it is conceptually straightforward, and the required configuration of the end effector is easy to sense and understand. Moreover, it is easy to imagine, define, and determine the motion of the manipulator end effector. Inverse kinematics is applied multiple times at each point along the trajectory to obtain values of robot joints. Commonly, when using Cartesian planning algorithms, the planned path passes through singular configurations of the robot manipulator, or it may override the manipulator's reachable workspace. In practice, in order to kinematically avoid the singularity problem of the manipulator, it is necessary to provide a sufficient number of via points inside the workable area. Cartesian trajectory planning is ordinarily used for precise local motion (Hussain 2017).

12.5 Path Planning Using Configuration Space

The main problem of planning a robotic path is finding a continuous, collision-free path for the robot from start to target configuration. Configuration space (C-space) can be considered an important tool for finding and formulating the path solution. C-space points are computed mathematically, disregarding all singular points so that the planned path represents a unique inverse kinematic solution, without redundancy or undefined states. Another benefit of using C-space for path planning is that a final path can be summarized easily in a unified method of finding the appropriate map that converts the robot's complex geometry to a single C-space point. The main difficulty of using C-space planning is that the intricacy of the path-planning difficulties raises exponentially with increasing number of degrees of freedom (DOF), which leads to increasing the C-space dimensions and the required parameters to specify the robot configuration.

The first contribution to precise C-space path planning was made by Lozano-Perez. Lozano-Perez and Wesley proposed simplification of complex robot geometry shape to a defined point in "configuration space" (C-space). In his path-planning method, two sub-problems were specified: (i) finding the C-space map and (ii) finding the appropriate collision-free path. Configuration space requires intensive computations and is often overlooked by researchers (Hussain 2017), though is a useful tool for finding path solutions.

12.6 Heuristic Methods

12.6.1 A Algorithm*

The A* algorithm is the most commonly used search algorithm for finding the shortest path. It represents an extension of Dijkstra's algorithm. The usage of the heuristic process is what makes the A* different from other graph search algorithms. The heuristic provides an estimated distance from a current node to the goal node, denoted by $h(n)$. The A* takes into account the cost from the start node to the current node, denoted as $g(n)$. The cost function is determined as follows (Raheem and Hussain 2017a, b):

$$f(n) = g(n) + h(n) \tag{12.1}$$

Each C-space node in the total map is represented by a joint angle pair, $n : (\theta_1,\ \theta_2)$. The A* algorithm includes two lists: open list (O_List) and closed list (C_List). It chooses the nodes that have minimum cost functions to create a path from the start node to the goal node.

12.6.2 D Algorithm*

Algorithm D* or "dynamic algorithm A*" behaves similarly to algorithm A*. The cost of the arc changes dynamically during the execution of the algorithm. The D* algorithm uses a Cartesian grid made up of eight-connected nodes to attain the robot position. The D* algorithm can be used to solve the path-planning problem based on the assumption that the robot must move in free space from the start point and continue exploring the area in order to find the minimum-cost path until reaching the destination point. The D* algorithm analyzes the robot's environment and expresses the problem space as a series of cells that indicate the position of the robot with the corresponding arcs or connections (Choset 2007; Nosrati et al. 2012).

Each cell in the D* algorithm contains an estimated cost of the path to the destination point and a back-pointer to one of its neighbor cells as an indicator of the destination direction. The arc's direction (connections) between a pair of neighboring cells is assigned with a positive scalar value of the motion cost (Choset 2007; Nosrati et al. 2012).

Algorithm D* uses a reverse direction search strategy, in which the D* algorithm forces the robot to start searching from the target cell and moves backward to other cells along arcs until it reaches the starting cell by repeatedly selecting a cell from the open list to evaluate and calculate the path cost to the goal. The D* algorithm moves from cell to cell along arcs and repeatedly selects a cell from the open list to evaluate and calculate the cost to the target cell. Finally, eight minimum cost neighboring cells are put on the open list (Raheem et al. 2019).

The D* algorithm sets conditions that specify when all alterations are made, either to find a new path or to continue using an old optimal path that has been chosen previously. D* is computationally efficacious and memory active and can be applied in unrestrained areas. Formulating the conditions for repeating the optimal search path problem in a directed graph includes marking the arcs by the transition cost values to prepare a cost range on the continuum. The arc cost corrections that are normally measured by sensors can be saved for a little time, while the well-known, approximately evaluated and calculated arc costs are the elements that make the environment map. The D* algorithm ensures keeping a closed list to save the path nodes and obstacles nodes and opens an expansion list for further path computations. In the D* algorithm, the arc cost parameter changes dynamically at each iteration. The D* algorithm sense is dynamic, and the path traverse cost can be minimized or maximized dynamically. This strategy can be used for any plan computation, including visibility graphs and grid-cell structures (Hameed 2019; Stentz 1994). The objective function (*F*) D* can be illustrated by A*:

$$F = g + h \tag{12.2}$$

where the cost estimated from the initial point is *g* while *h* is the motion cost to the target point.

Cost changes to neighbors are propagated as shown in Fig. 12.2a, b.

$$\mathrm{N}(\mathrm{x1}, \mathrm{x2}) = 1$$

$$\mathrm{N}(\mathrm{x1}, \mathrm{x3}) = 1.414$$

$$\mathrm{N}(\mathrm{x1}, \mathrm{x4}) = 100000, \text{in case of x4 has an obstacle and x1 is a free cell.}$$

$$\mathrm{N}(\mathrm{x1}, \mathrm{x5}) = 100000.4, \text{in case of x5 has an obstacle and x1is a free cell}$$

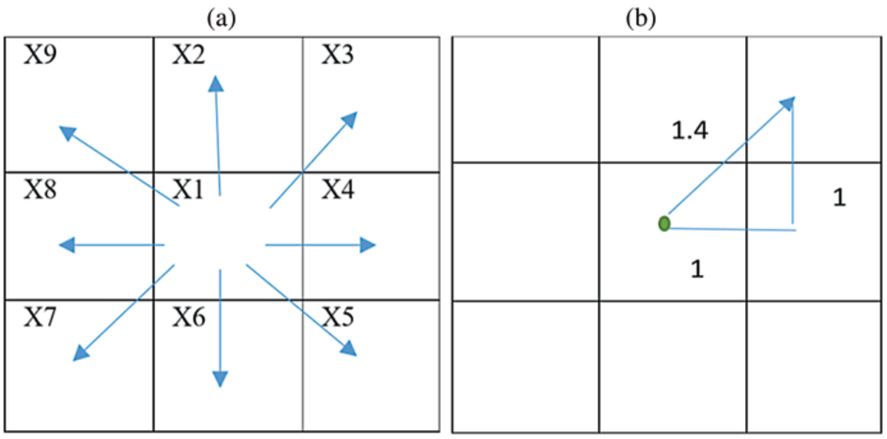

Fig. 12.2 Node expansion and calculation. (**a**) Node expansion, (**b**) node expansion calculation

The initial node has an arc cost equal to (0.0), and the arc cost of the vertical and horizontal nodes can be computed as:

$$\text{From x1 to x2} = \sqrt{(0-0)^2 + (0-1)^2} = 1$$

$$\text{From x1 to x4} = \sqrt{(0-1)^2 + (0-0)^2} = 1$$

In case of a diagonal nodes calculation, the arc cost is:

$$\text{From x1 to x3} = \sqrt{(0-1)^2 + (0-1)^2} = \sqrt{2} = 1.414 \text{ and so on.}$$

12.7 Theory of Particle Swarm Optimization (PSO)

PSO is an optimization method that is considered a stochastic onerous technique akin to the intelligent flying motion of birds. PSO was inspired by natural social and dynamic motion behavior, as well as the communication of fish, insects, or birds. Convergence speed and quality of solutions are advantages of the technique (Raheem and Hameed 2018; Boonyaritdachochai et al. 2010).

The principle of PSO relies on generating a fixed number of particles at random positions in a certain workspace, with velocity of particles specified randomly. Each particle has a memory that stores all the best cell locations that have been visited prior to the current cell location; and stores the improved fitness over time (Poli et al. 2007). In each PSO method iteration, the pos_{ij} and $vel_{i,j}^{t}$ vectors of particle i

are adjusted for each j dimension to make the particle i toward either the previous best vector $pbest_{ij}$ or toward the best vector that represent the swarm's ($gbest_{ij}$).

$$vel_{i,j}^{t+1} = w \times vel_{i,j}^{t} + c_1 \times r_{1,j} \times \left(pbest_{ij} - pos_{ij}\right) + c_2 \times r_{2,j} \times \left(gbest_{ij} - pos_{ij}\right) \tag{12.3}$$

Each particle (the point that is chosen from D* path) has an updated position according to the equation below, using the new velocity for that particle:

$$pos_{ij}^{t+1} = pos_{ij}^{t} + vel_{ij}^{t+1} \tag{12.4}$$

where the cognitive coefficients $c_1$and c_2 are applicable under the condition ($c_1 + c_2 \leq 4$) and $r_{1,j}$ and $r_{2,j}$ are real numbers that can take a value between [0, 1] randomly, while controlling the momentum of the particle can be done by the weight of inertia w. The velocity vel_{ij} can take a value inside the range [$-vel_{max}$, vel_{max}] to decrease leaving chance of being out of the search space by the particle. Given a space of search that can be identified within the bounds [$-pos_{max}$, pos_{max}], then the vel_{max} value setting is usually $vel_{max} = \mathrm{k} * pos_{max}$, where $0.1 \leq \mathrm{k} \leq 1.0$.

A weight (w) that represents large inertia has two cases when its value is small, making the global search easier. A large w value simplifies localization (Poli et al. 2007).

12.8 Research Case Studies

12.8.1 Heuristic Path-Planning Enhancement Based on Free Cartesian Space Analysis

In heuristic path planning, when free Cartesian space analysis is known, the shortest path and trajectory planning of the two-link robot arm with 2 DOF in the 2D static known environment can be analyzed. The analysis deals with three main problems. The first concern regards the construction of free Cartesian space by analyzing the inverse kinematic solutions, which guarantees collision-free path planning. The second problem focuses on generating the shortest path that satisfies the aims of motion and applies the D* algorithm. The third problem is the selection of the specified number of intermediate via points and attaining the corresponding smooth trajectory through using fifth-order polynomial equations. Results illustrate that free Cartesian space ensures a collision-free path and trajectory planning (Raheem et al. 2019).

12.8.2 Implementation of D Algorithm for Path Planning*

The modification of the D* algorithm is proposed for solving the problem of generating the shortest collision-free path for a 2-DOF planar robotic arm in a known stationary (static) environment. The primary principle of applying the D* algorithm in proposed robot arm path planning makes use of navigation and environment and analyzes criteria to construct the short path of the arm end effector subject to certain constraints regarding safety and smoothness.

The proposed method is composed of three stages: free Cartesian space analysis followed by path construction and smooth trajectory generation. In the first stage, the proposed system is initialized by selecting a robot arm environment where the D* algorithm plans the shortest path. Subsequently, the analysis of the corresponding free Cartesian space has been made according to the free Cartesian space analysis procedure in order to reveal free space, ensuring a collision-free path planning (Raheem et al. 2019).

In the second stage of path construction, the D* algorithm is applied to generate the shortest path from start to goal points by avoiding all the obstacles. The principle of operation of the proposed D* algorithm is local planning for the shortest path within the free Cartesian space until it reaches the goal point. Moreover, the D* algorithm is initialized by placing the start "current" node on the open list, which is inserted into the currently planned path. Subsequently, the current node is expanded to eight connected neighborhood nodes for determining the next arm movement or the "candidate next node." However, only nodes that belong to the free Cartesian space are used. Ultimately, this iterative process of node expansion and selection is terminated as soon as the goal point is appended to the closed list (Raheem et al. 2019).

In the third stage of the method, fifth-order polynomial equations are applied for smooth trajectory generation based on the generated path of the D* algorithm. Initially, several intermediate points are randomly selected from the shortest path generated by the D* algorithm to be added to the initial and goal points. Then quintic fifth-order polynomial equations are used to add the required smoothness action and guarantee generating a smooth trajectory at a specific time according to equations (Raheem et al. 2019; Sreenivasulu 2012), after transforming the intermediate points to a joint angle using an inverse kinematics function. Consequently, a checking function is applied for checking and testing the generated segments within the free Cartesian space. In the case of being outside the free Cartesian space, the process iteratively selects and generates another point and corresponding trajectory until achieving a possible trajectory whose segments belong to the free Cartesian space. Subsequently, the cost value is calculated for the trajectory according to the following equation:

$$\text{Trajectory} - \text{Cost} = \sum_{i=1}^{k} \sqrt{\left(x_i - x_{i+1}\right)^2 + \left(y_i - y_{i+1}\right)^2} \tag{12.5}$$

where k is the number of the configuration that is equal to the amount of specific time along the trajectory, (x_i, y_i) representing the current coordinate of ith points of the trajectory.

12.8.3 Simulation Result

Environment maps with various difficulties were used to test and verify the proposed method. The dimension limits of the environments from −50 to 50 cm for both x and y contain various shapes of static obstacles. Moreover, both robot arm links have an equal length of 25 cm. In addition, the suggested joint variable limitations are specified as $0 \le \theta_1 \le 360$ and $0 \le \theta_2 \le 360$. Initially, the procedure of the free Cartesian space analysis has been applied to verify the required analysis for the environments (as shown in Fig. 12.3), in order to discover the free space and ensure finding a collision-free planning path. Accordingly, inverse kinematics functions are applied to construct offline planning both of the free elbow-up solution area and free elbow-down solution area in the Cartesian space (Raheem et al. 2019).

The final smooth trajectory of the robot is shown in Fig. 12.3 which is the result of applying the equations of the quintic polynomial trajectory. This clarifies the difference between the planned path of the D* algorithm and the smoothed generated final trajectory; green points indicate the intermediate via points that have been selected from the planned D* path (Raheem et al. 2019).

Figure 12.4 demonstrates the movements of the arm that follow the shortest path, where the red line indicates the first link of the robot arm and the blue line indicates the second link of the robot arm.

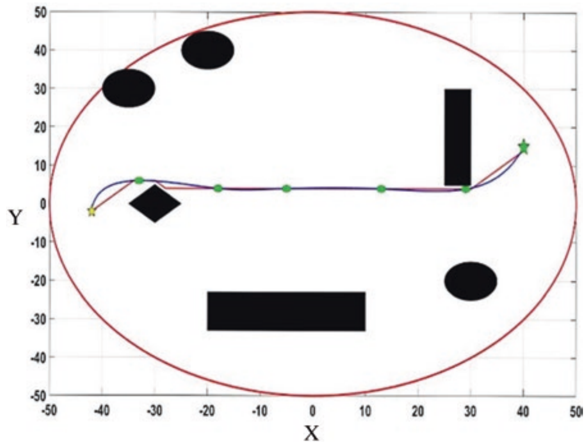

Fig. 12.3 The planned D* path and generated trajectory. (Raheem et al. 2019).

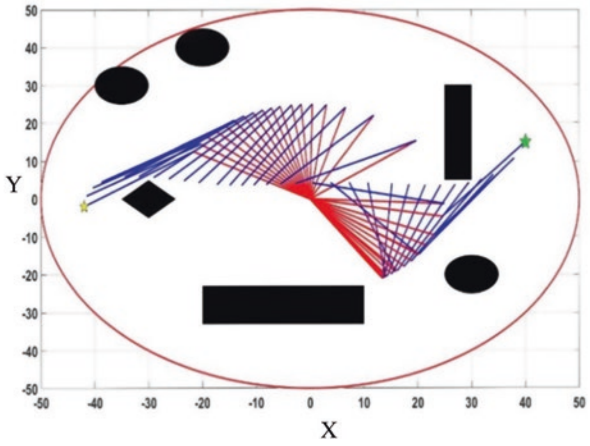

Fig. 12.4 Two-link robot arm movements. (Raheem et al. 2019).

12.9 Artificial Potential Field (APF) Based on PSO for Factor Optimization

An improved and comprehensive hybrid method for robot path planning is proposed, which firstly uses PSO for determining the best value for each factor of the artificial potential field (APF) and makes a progressive improvement in the resultant path in an iterative process until the shortest path is found. A spline equation was used for smoothing the path and producing a final smoothed trajectory. The results clearly show the strength and the efficiency of the hybrid PSO and APF method (Raheem and Badr 2017).

12.9.1 Artificial Potential Field Theory

In a mass point path-planning case, q refers to the position coordinate of the robot moving in a 2D environment. The current position coordinate of the robot is referred to as $q = [x\, y]$, while the obstacle position coordinate is referred to as $q_{obs} = (x_{obs},\ y_{obs})$; similarly the coordinate of the goal position is denoted by $q_{goal} = (x_{goal},\ y_{goal})$. The parabolic shape is the general style of the artificial potential field function, where Fig. 12.5a shows the attractive potential, which increases quadratically with the distance to the target point (Raheem and Badr 2017).

$$U_{att}\left(q\right) = \frac{1}{2} k_a\, d^2\left(q, q_{goal}\right) \tag{12.6}$$

where k_a refers to the relative factor that constructs the attractive potential surface, $d(q,\ q_{goal})$, which is the Euclidean distance from the current position of the robot to

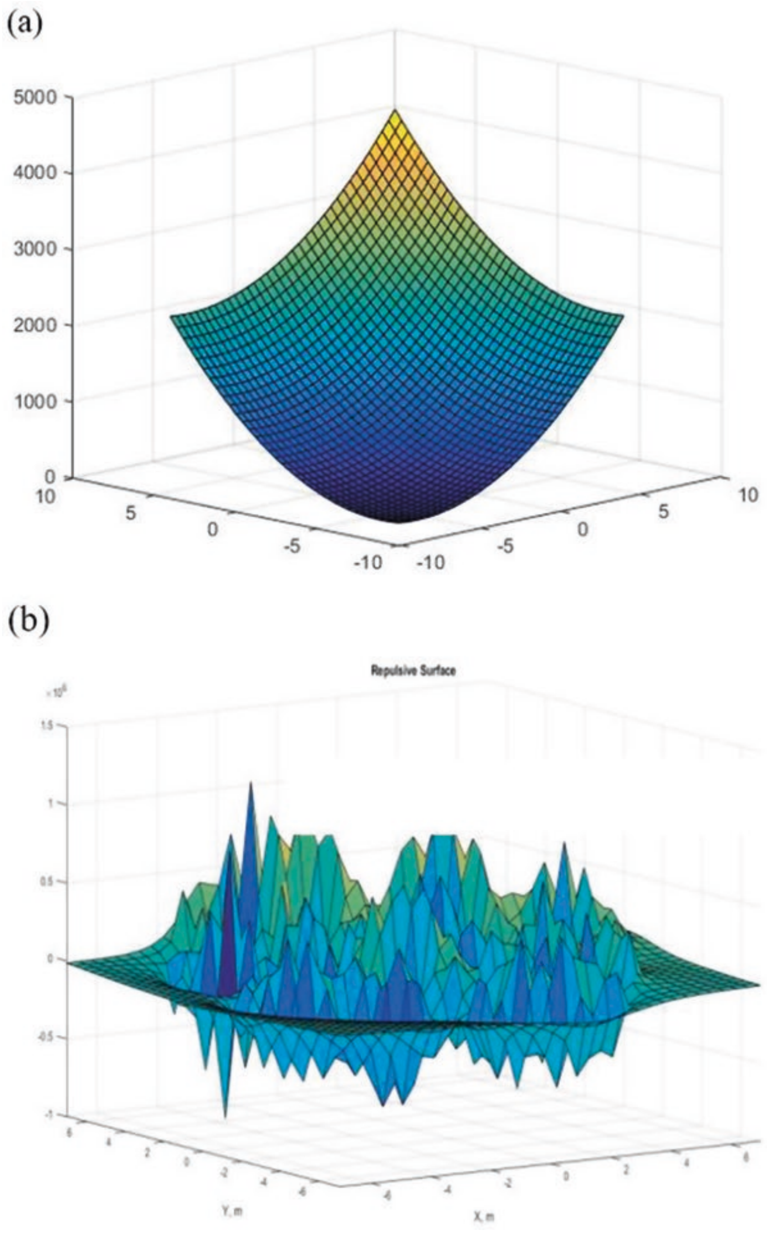

Fig. 12.5 Potential artificial fields, (**a**) attractive surface, (**b**) repulsive surface

the goal point (target) q_{goal}. The attractive force can be calculated as the attractive potential field negative gradient (Raheem and Badr 2017):

$$F_{att}\left(q\right) = -\nabla U_{att}\left(q\right) = -k_a\, d\left(q, q_{goal}\right) \tag{12.7}$$

There is a relative relationship between the repulsive force and the obstacle distance to the robot position. The surface of the potential repulsive was constructed using the repulsive forces produced from all obstacles located within the environment. The repulsive potential function can be represented by Eqs. (12.3) and (12.4), while Fig. 12.5b represents the corresponding repulsive surface (Raheem and Badr 2017).

$$U_{rep}\left(q\right) = \sum_i U_{repi}\left(q\right) \tag{12.8}$$

where $U_{repi}(q)$ denotes the repulsive potential field which is constructed by the obstacles and i represents obstacle number.

$$U_{rep}\left(q\right) = \begin{cases} \frac{1}{2} K_{rep}\left(\frac{1}{d\left(q, q_{obs}\right)} - \frac{1}{d_0}\right)^2 d\left(q, q_{goal}\right)^n & if\;\; d\left(q, q_{obs}\right) < d_0 \\ 0 & if\;\; d\left(q, q_{obs}\right) > d_0 \end{cases} \tag{12.9}$$

The variable q represents the current position of the robot, n refers to a real integer number, the obstacle position is q_{obs}, and d_0 refers to a positive number, which represents the distance to the efficacious obstacle; a distance that measured between all obstacles and the robot can be denoted as d (q, q_{obs}), while the repulsive potential surface factor which is an adaptable constant is represented in Eq. (12.9) as K_{rep}. The overall repulsive force has a negative slope due to its nature as explained in Eq. (12.10) (Gue et al. 2013):

$$F_{rep}\left(q\right) = -\nabla U_{rep}\left(q\right) \begin{cases} K_{rep}\left(\frac{1}{d\left(q, q_{obs}\right)} - \frac{1}{d_0}\right)\frac{\left(q, q_{obs}\right)}{d^{3\left(q, q_{obs}\right)}} & if\;\; d\left(q, q_{obs}\right) < d_0 \\ 0 & if\;\; d\left(q, q_{obs}\right) > d_0 \end{cases} \tag{12.10}$$

The two surfaces, the attractive potential surface U_{att} and the repulsive potential surface U_{rep} , can be combined into the total potential field as shown in Eq. (12.11):

$$U\left(q\right) = U_{att}\left(q\right) + U_{rep}\left(q\right) \tag{12.11}$$

The applied forces toward the robot are produced from the negatively gradient that utilizes a well-known method called a steepest descent to drive the direction of the robot to its desired target value.

$$F(q) = -\nabla U(q) = -\nabla U_{att}(q) - \nabla U_{rep} \tag{12.12}$$

where ∇U represents the gradient vector of U, while the total impact force that affects the robot work can be calculated as the two components' summation, the attractive vector force and the repulsive vector force, F_{att} and F_{rep}, respectively (Li et al. 2000):

$$F(q) = F_{att}(q) + F_{rep}(q) \tag{12.13}$$

12.9.1.1 Proposed Method

The optimization of the artificial potential field factor (APF) uses the PSO technique. Two modifications have been carried out. The force and its direction in the ordinary artificial potential field will be initially modified. The environment can be represented and converted to a grid of points. Each point on the grid has a force that is computed from two sources: the attractive force due to the goal point and the repulsive force due to the obstacle (if the range of influence covers these points). Each point can be affected by two forces, firstly toward the x-axis direction and secondly toward the y-axis direction. Some researchers change these forces to impose the path of the robot according to Eqs. (12.14), (12.15), (12.16), and (12.17) (Raheem and Badr 2017):

$$F_{x\_total} = F_{x\_att} + F_{x\_rep} + F_{y\_rep} \tag{12.14}$$

$$F_{y\_total} = F_{y\_att} + F_{y\_rep} - F_{x\_rep} \tag{12.15}$$

$$F_{x\_total} = F_{x\_att} + F_{x\_rep} - F_{y\_rep} \tag{12.16}$$

$$F_{y\_total} = F_{y\_att} + F_{y\_rep} + F_{x\_rep} \tag{12.17}$$

The results of Eqs. (12.14) and (12.15) (Fig. 12.6) and Eqs. (12.16) and (12.17) (Fig. 12.7) clarify that all forces have one direction usually surrounding the obstacle.

In this work, the third term in Eqs. (12.14) through (12.17) has been removed; another term has been added, which was found by a trial and error tuning process to be appropriate to specific cases; and this was added to each term in the equation making the overall force in the coordinates of *x*-axis and *y*-axis going toward the target point and simultaneously ensuring a collision-free path as presented in Eqs. (12.18) and (12.19) and Fig. 12.8:

$$F_{x\_total} = F_{x\_att} + F_{x\_rep} - 0.75 \tag{12.18}$$

$$F_{y\_total} = F_{y\_att} + F_{y\_rep} + 0.5 \tag{12.19}$$

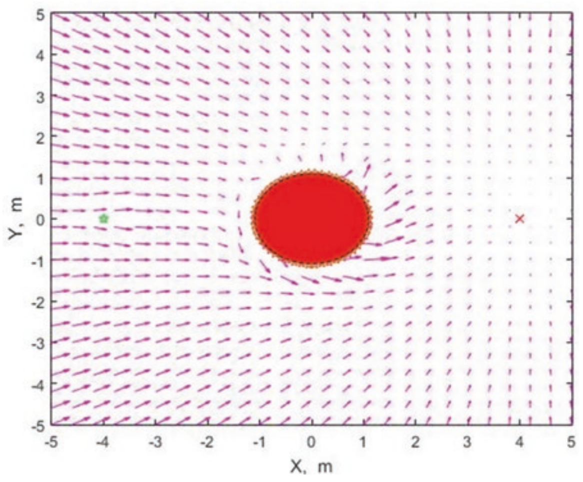

Fig. 12.6 The forces' direction according to Eqs. (12.14) and (12.15). (Raheem and Badr 2017)

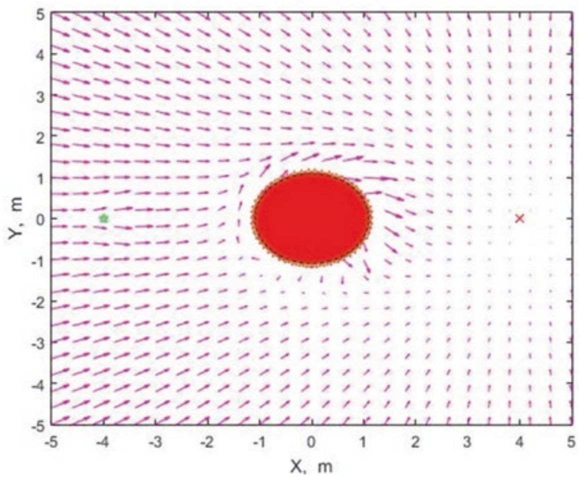

Fig. 12.7 The forces' direction according to Eqs. (12.16) and (12.17). (Raheem and Badr 2017)

The second modification includes focusing on the APF attractive force factor (k_a) and repulsive force factor (k_r). This modification plays a significant function in this proposed work. In case the value of the attractive force is very high, only its effectiveness will be considered. Furthermore, if it has a small value, then its effect can be disregarded. PSO is applied to determine the forces; the repulsive force factor will affect a certain particular area, which indicates the APF influence around obstacles that need to be optimized. The mathematical application of the PSO algorithm is according to Raheem and Badr (2017).

The details of the proposed method include the computation of the factors of both attractive and repulsive forces and can be listed in accordance with the following ten steps:

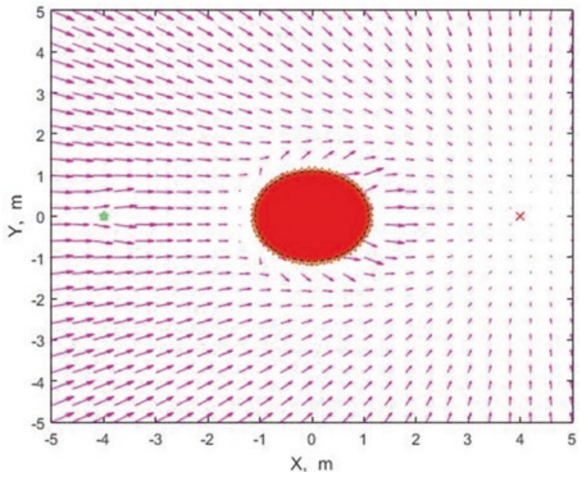

Fig. 12.8 The forces' direction according to Eqs. (12.18) and (12.19). (Raheem and Badr 2017)

1. Initialize the PSO parameters and specify the APF factors (K_a, K_{rep}) and their suitable ranges.
2. Initiate the generation APF factors randomly according to the suggested extents.
3. Perform the algorithm APF method instantly by using the factors that are randomly generated after the construction of the APF, and then find a possible path with the calculations of the cost function.
4. Update the PSO algorithm factors which include the local and the best global factors in accordance with the acceptable path.
5. Repeat the previous three steps, (step 2 to step 4) for the chosen particle number.
6. Update the values of position and velocity according to the mathematical equations below for the following APF factor calculation:

$$\begin{aligned} v_i(k+1) &= w \times v_i(k) + c_1 \times rand \times \left(P_{iatt}(k) - x_{iatt}(k)\right) \\ &+ c_2 \times rand \times \left(P_{gatt}(k) - x_{iattL}(k)\right) \end{aligned} \tag{12.20}$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{12.21}$$

$$\begin{aligned} v_i(k+1) &= w \times v_i(k) + c_1 \times rand \times \left(P_{irep}(k) - x_{irep}(k)\right) \\ &+ c_2 \times rand \times \left(P_{grep}(k) - x_{irepL}(k)\right) \end{aligned} \tag{12.22}$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{12.23}$$

where *rand* is a random real number between [0, 1], P_{iatt} is the best position for the *iatt* particle for finding the attractive factors, x_{ia} is the current existing position, and $x_{iaL}$is the best local position where the letters (*att*) indicate the attractive factors, a similar definition to the Eqs. (12.20) through (12.23) where the letters (*rep*) indicate the repulsive factor.

7. Apply the APF method using the updated values of the repulsive and the attractive factors, then compute the cost objective function.
8. Update the APF values locally and globally to determine the factors' best global values.
9. Repeat the previous three steps (step 6 to step 8) using the updated factors according to the selected population number and the number of iterations.
10. After finding the best value globally for each factor, the related route generated will be the best computed path.

12.9.2 Simulation Result

12.9.2.1 The First Proposed Environment

Figure 12.9 shows the first robot navigation environment using the proposed method of APF-based PSO as a planning algorithm to find an optimal path, while the smoothed path after using the spline method is shown in Fig. 12.10. In this suggested test environment, the robot moves from the start point (2, 4) and reaches the target point (−1, −6), in which seven static obstacles are included in a test environment of dimensions [−7 to 7] for both *x* and *y* coordinates.

12.9.2.2 The Second Environment

Figure 12.11 represents the first resultant path of APF-based PSO, but Fig. 12.12 is the final smoothed path after applying the spline equations for smoothness. In this environment, seven stationary (static) obstacles are included, and the starting point

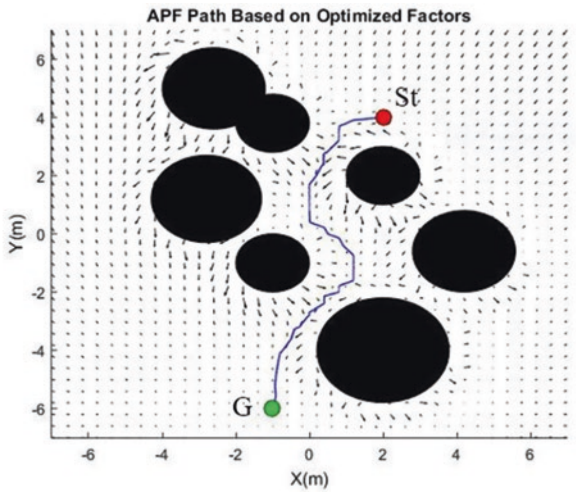

Fig. 12.9 Robot path planning for the first environment after employing the APF method. (Raheem and Badr 2017)

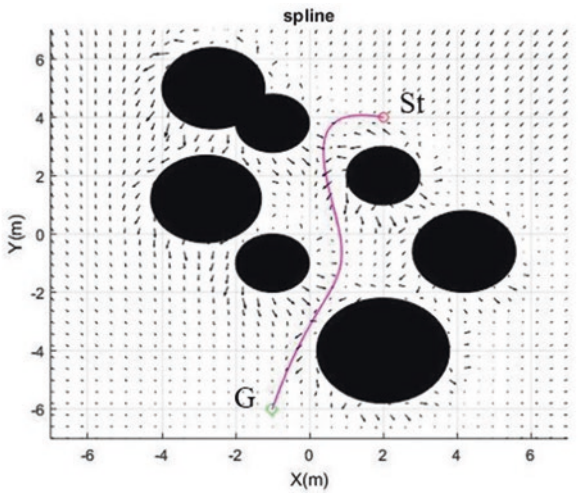

Fig. 12.10 Robot path planning for the first environment after employing the spline method. (Raheem and Badr 2017).

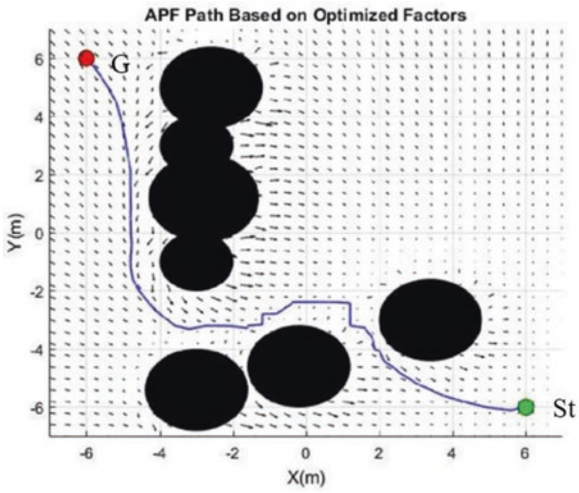

Fig. 12.11 Robot path planning for the second environment after employing the APF method. (Raheem and Badr 2017)

is at (−6,6) while the target point is at (6,−6); the environment has the same dimensions.

12.10 Dynamic Environment Path Planning Using D* Heuristic Method Based on PSO

This method is used for robots to find a safe and short route of planning in a dynamic moving obstacle environment. These moving obstacles can be various objects, people, animals, or other moving robots. Hybrid robotic path-planning methods use the

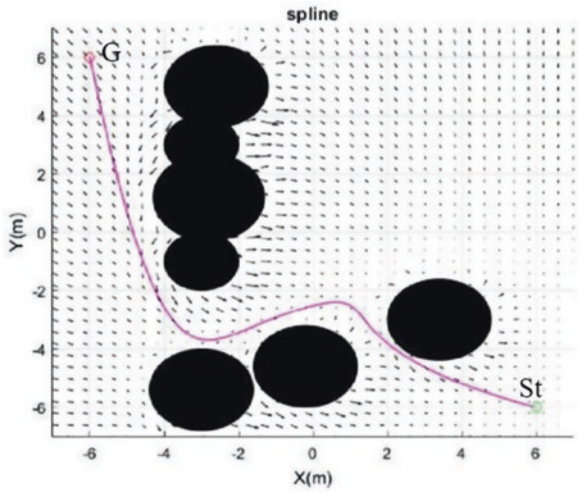

Fig. 12.12 Robot path planning for the second environment after employing the spline method. (Raheem and Badr 2017)

combination of heuristic calculations and an optimization algorithm. The D* heuristic method was used again to determine the shortest path. In addition, the method of particle swarm optimization (PSO) can be applied after the D* algorithm for further improvement of the path and to provide the final optimal path. Computations of this hybrid method in a dynamic environment consider complete changes at the domain of each sample of motion. The effectiveness of this hybrid proposed method was verified through simulation results (Raheem and Hameed 2018).

12.10.1 Proposed Method: Hybridization of D* Heuristic Method Based on PSO

The heuristic D* algorithm is used as an algorithm for path planning, which relies on node expansion in a test environment with dynamic moving objects. The creation of the test environment comprises defining the limitation of the map and starting and ending points in addition to the obstacles' location and size (Raheem and Hameed 2018).

After the creation of an environment, the execution of the algorithm includes standing the robot at the goal node; it starts moving virtually with an initial expansion to its eight connected neighbors. Each node has an initial cost function, and the robot selects the node with the lowest cost function for moving and places it in the D* closed list; further nodes will be placed into the open list. The huge cost-function dynamic obstacles that were sorted from the beginning of the procedure in the closed list and until reaching back to the starting node are taken into account (the D* method is called a reverse searching method). Upon completion of the heuristic D* search, the robot moves from the previously saved starting node in the closed list to

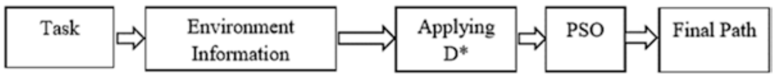

Fig. 12.13 Mass point proposed path-planning stages in a dynamic environment. (Raheem and Hameed 2018)

the goal node averting all the obstacle nodes, current node, and the node of the next step of motion.

The moving obstacles have a constant velocity, while the mass point (a virtual point that represents a robotic object and can be considered as either a mobile robot center or after some modifications as a robot manipulator end effector) velocity is changing with time ($MS \times Ts$) during navigation and can be determined by (Raheem and Hameed 2018):

$$v_x(t) = \frac{\Delta x(t)}{MS \times T_s} \tag{12.24}$$

$$v_y(t) = \frac{\Delta y(t)}{MS \times T_s} \tag{12.25}$$

where Δx is the change in x-axis distance, Δy is the change in y-axis distance, MS refers to the sample of motion, and Ts is the time of the sample at 0.01 s intervals.

After determining the path using the heuristic D* method, a path improvement stage requires applying the PSO optimization method for further path enhancement by eliminating the sharp edges and shortening the path. The path produced from the D* algorithm generally takes a stair shape. The stages of the proposed mass point path planning in a known dynamic environment are presented in Fig. 12.13.

12.10.2 Simulation Result

12.10.2.1 Test Environment Number One

A non-interactive path solution dynamic environment has been tested to verify the proposed method of path planning by finding a collision-free and shortest path solution based on the combination of the D* heuristic method and PSO optimization technique among the moving obstacles and the robot. In this environment, many obstacles have different sizes and different behaviors of motion. The environment contains eight obstacles, six moving obstacles with different styles of movement, and two static obstacles. In this critical case, only a few possible paths exist that can maintain the shortest route length (Raheem and Hameed 2018).

The use of the PSO and D* algorithm ensures finding the shortest path avoiding collisions with dynamic and static obstacles, as shown in Fig. 12.14a. The hybrid

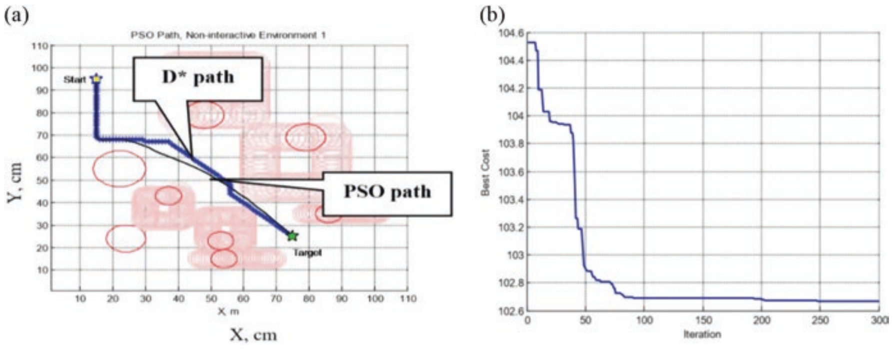

Fig. 12.14 Non-interactive path planning using D* algorithm based on PSO: (**a**) shows D* and D*-based PSO paths, (**b**) shows the curve of the path cost function using PSO

approach of the D*-based PSO enhanced the path to 102.6672 cm, as presented in Fig. 12.14b. The cost function reaches a 3.9013 cm minimum value after 300 motion samples (Raheem and Hameed 2018).

12.10.2.2 Test Environment Number Two

Non-interactive path solutions in a known dynamic environment that has been tested to verify the proposed method of path planning by finding a collision-free and shortest path solution based on the combination of the D* heuristic method and PSO optimization technique are provided here. In this environment, many dynamic obstacles have different sizes and different and difficult behaviors of motion. The environment contains six moving obstacles with different sizes and difficult styles of movement.

The use of the PSO and D* algorithm ensures finding the shortest path avoiding collisions with dynamic obstacles, as shown in Fig. 12.15a. The hybrid approach of the D*-based PSO enhanced the path to 106.6039 cm, as presented in Fig. 12.15b. The cost function reaches a 4.0651 cm minimum value after 300 motion samples (Raheem and Hameed 2018).

12.11 Interactive Path Solution Using Heuristic D* Method and PSO in Known Dynamic Environment

A new methodology is presented here for finding an interactive robot path-planning solution in a fully known dynamic environment. This methodology includes the use of the PSO technique together with an improved and modified D* heuristic method which is applied to the total analysis of the free Cartesian space at each sample of motion. The essential D* method has been modified to meet the requirements of the

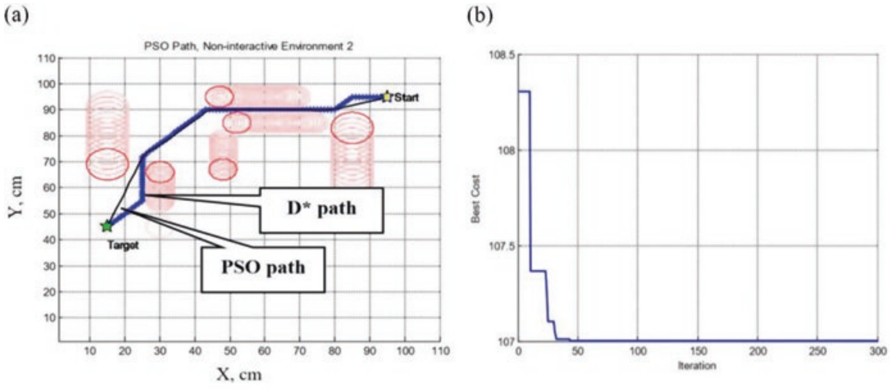

Fig. 12.15 Non-interactive path planning using D* algorithm based on PSO: (**a**) shows D* and D*-based PSO paths, (**b**) shows the curve of the path cost function using PSO

known dynamic environment by considering the addition of two cases, a motion stop and a return backward case. These cases are not covered in the original theory of the D* heuristic method (Raheem and Hameed 2019).

12.11.1 Proposed Method: Hybridization of Modified D Heuristic Method and PSO*

After applying the D* heuristic method and studying the final path solution which was a non-interactive path in a well-known environment that contains movable obstacles, an enhancement by finding an interactive path solution method has been proposed for finding an enhanced path by taking into consideration the full dynamic behavior of the obstacles, including spatial and temporal information. Finding the robot interactive path solution by applying the heuristic D* method in case of known obstacle time and position dynamic environment requires a map with all task information and limitations, start and goal positions, and the position, shape, and size of the obstacles. This method was used with a 2-DOF planar robotic arm as presented in Fig. 12.16. The interactive path solution includes the free Cartesian space calculations (FCS) in each motion sample of the robot and from node to node. In this case, the motion range of the robot can be described accurately. It is more intelligent to analyze and calculate multiple and changeable FCS maps for each motion sample. A new calculated FCS map will indicate the feasible area for the robot to move. Although this is a time-consuming process, it produces an accurate decision for the right motion step with high robot motion accuracy. Since this calculation process is an offline search method that starts after constructing the environment, the method will start when the robot is located at the goal node first and then expanded initially to its eight connected neighbors. Each node has an initial value of the cost objective

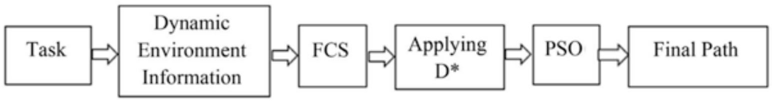

Fig. 12.16 Proposed interactive path-planning method in a dynamic environment for 2-DOF planar robotic arm

function, where the robot selects the node that has the minimum cost to continue moving on and then save it in the closed list, at the same time further remaining nodes will be saved in the open list. In the same way, keep in mind the obstacles that have big or huge values of the cost function, sorting these values and storing the initial starting positions in the closed list. At each sample of motion, the D* closed list is updated, to know the new locations of the obstacles, and the closed list will be dynamically changing at each motion sample and storing the nodes representing the obstacle at the current motion sample. The process will complete an offline iterative process only in case of reaching the start node. In case of search completed, the robot moves interactively from the initial node considering the closed list immediately to have required information about the free nodes prior to move on toward the goal node. The heuristic D* function f = g + h, where f is the objective function and g is the estimated cost function value from the start point while h is the cost to the goal (Raheem and Hameed 2019).

12.11.2 Simulation Results

12.11.2.1 First Environment

As a robot example, a 2-DOF planar robotic arm is used here. The arm length is assumed to be equal to half of the environment length, and each robot-link length is equal to 50 cm. The arm joints can theoretically rotate 360 degrees. Figure 12.17 presents the path of the D* algorithm for the 2-DOF planar robotic arm, where part (a) shows the end effector located at the start point at the beginning of the planning process. Part (b) shows the arm configuration after eight samples of motion. The arm at motion sample 15 was returned backward to avoid collision as shown in part (c). Parts (d) and (e) show the motion planning of the arm after 21 and 36 samples. At the sample of motion number 38, the arm was returned backward to avoid collision with a moving obstacle, while two samples later, the arm safely reached the target point as shown in parts (g) and (h) (Raheem and Hameed 2019).

Since this is an offline interactive path solution approach, optimizing the D* by applying the PSO algorithm has been done offline to remove the sharp edges and to shorten the D* path. Figure 12.18a shows the blue line of the D* path, and the black line is the final D*-based PSO path solution. Moreover, Fig. 12.18b explains that the length of the D*-based PSO path solution was reduced by 3.2846 cm compared with the original D* path (Raheem and Hameed 2019).

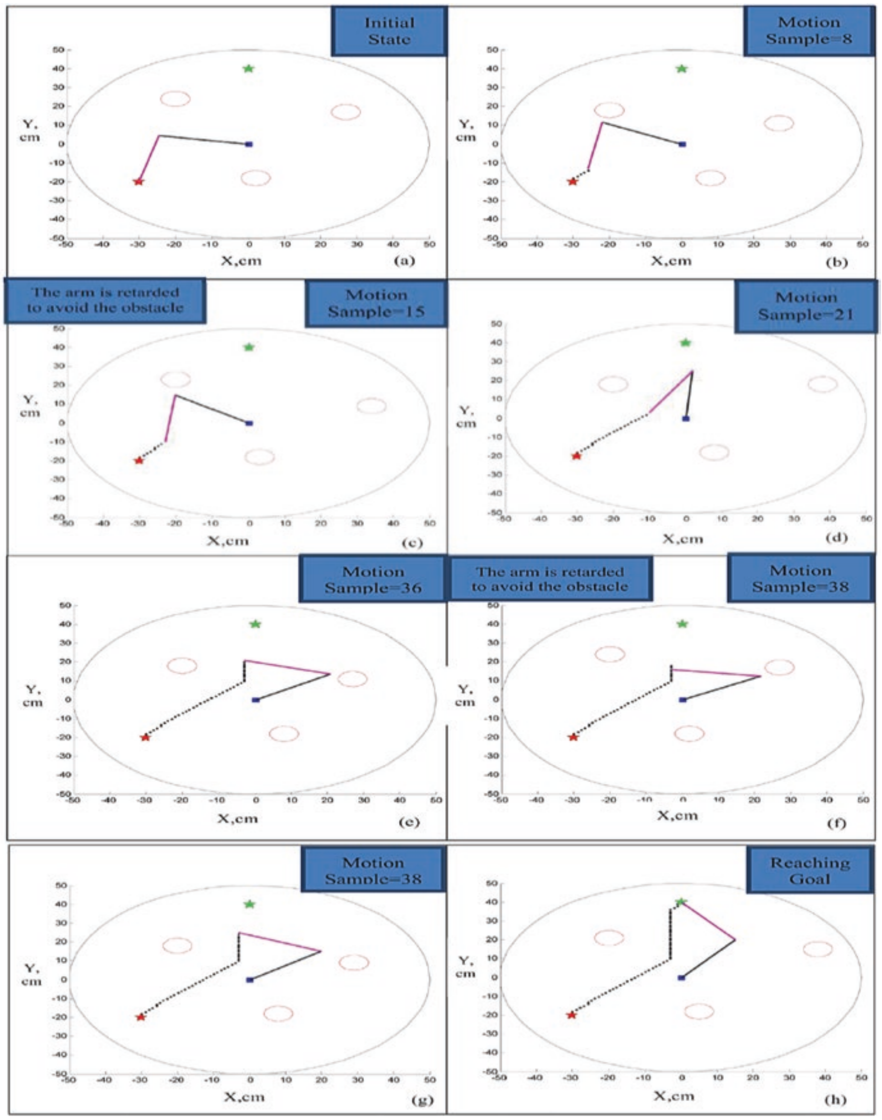

Fig. 12.17 2-DOF planar robot path results after applying the D* algorithm. (Raheem and Hameed 2019)

The motion style of this path solution is clearly featured as an interactive path movement, where the robot must make a decision under certain circumstances and may be stopped or returned backward in such a way that the stopping state and backward movement reduce the time and ensure getting a shorter collision-free path.

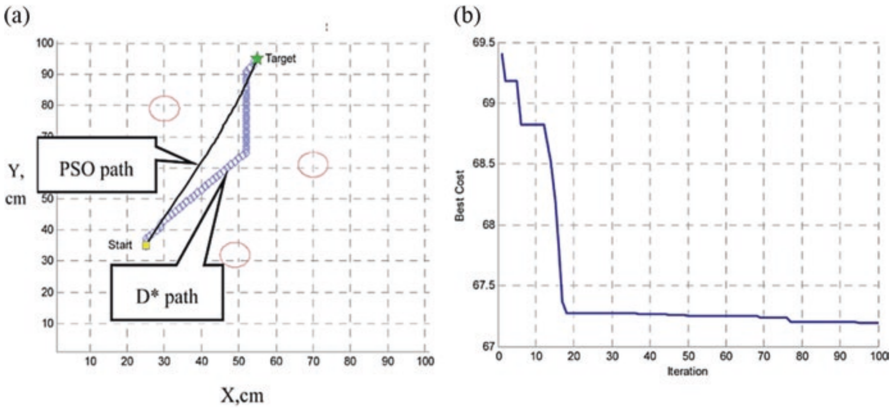

Fig. 12.18 Interactive path planning using D* algorithm based on PSO: (**a**) final D* and D*-based PSO paths and (**b**) cost function of the path

12.11.2.2 Second Environment

For further testing, this method was proposed and verified by a second more complicated environment setup where four dynamic obstacles move closer to the links of the robot and two of these obstacles have a square path behavior of motion. The robot can make an appropriate decision for choosing the next points freely and intelligently within an allowable area. Figure 12.19 presents the D* path of the 2-DOF planar robotic arm, where part (a) indicates the start of planning (the robotic end effector located on the initial starting point), part (b) presents the robot motion at the end of 11 samples, part (c) presents the robot in a return backward state at sample 21 of motion, parts (d) and (e) present the robot motion configuration after 33 and 42 samples of motion, respectively, and parts (g) and (h) clarify the robot motion after 46 samples of motion; the robot was returned backward to avoid collision with the closest moving obstacle before reaching the target point of destination safely (Raheem and Hameed 2019).

Since the location and time behavior of the moving obstacles are known, finding a path solution as a process can be performed offline, so PSO plays an important role in completing the process and finding the final applicable best, shortest, and optimal path with removing the sharp edges of the heuristic D* original path. Figure 12.20a shows both of these paths, D* path and D* based on PSO path, where the blue line indicates the path of D*, while the black line indicates the D* based on PSO path. Figure 12.20b plots the cost progress of the objective function optimization and verifies that the last definitive optimized path utilizes heuristic D* based on optimization technique (PSO); the length of the path was shorter than the original D* path by 4.3724 cm (Raheem and Hameed 2019).

In comparison, Cubero (2007) tried to use the heuristic D* method to solve the same problem for a mobile robot path-planning case. The proposed method introduced here differs because of its new characteristic of the modification added to the

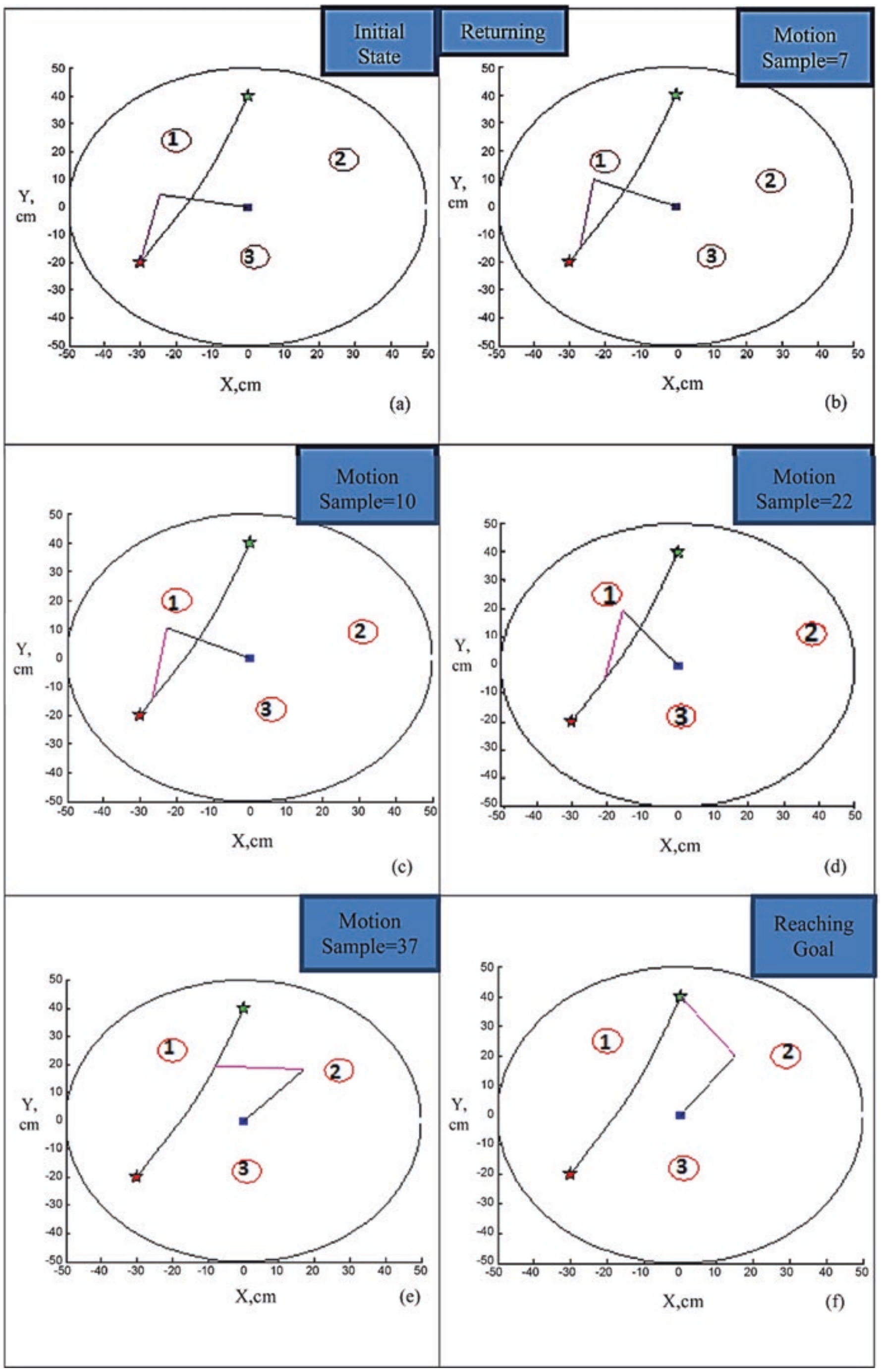

Fig. 12.19 The two-link robot arm result for the D* path for the second environment. (Raheem and Hameed 2019)

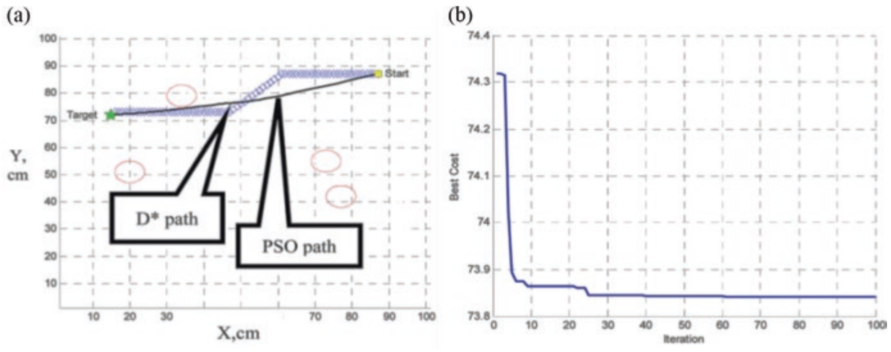

Fig. 12.20 (**a**) Final D* and PSO path and (**b**) the path cost function for the second environment

D* heuristic method which is suggested in our work. This modification ensures the safety, interactivity, and finesses of the final PSO-optimized path.

12.12 Heuristic A* Path Solution Based on C-Space Analysis

In this method, a configuration space (C-space)-based path-planning approach for a 2-DOF planar robotic arm was presented. The analysis of the constructed C-space map makes the process of finding a feasible, safe, and successful path solution much easier. After constructing and analyzing the C-space map for the planar robot which comprises all the possible collision situations between the obstacles and the robot links and joints from the base till the end effector, the heuristic A* method has been applied to find an optimal path in the C-space map that represents a successful heuristic path in the Cartesian workspace. A modification on the original heuristic A* method equations and calculations has been carried out to make this method applicable in the C-space. Moreover, further modifications are required to apply the A*-based C-space methodology for robotic manipulators with high degrees of freedom (more than 2 DOF). The method was verified by simulation results and proved the overall C-space map construction accuracy. These results were efficient and successful and guarantee a collision-free path from the start point to the target point by priorly eliminating all the probabilities of collisions with any robot point that appeared clearly in the C-space map (Raheem and Hussain 2017a, b).

12.12.1 C-Space Derivation

For a robotic arm, C-space is the space that can be expressed by the joint's variable parameters, whether it is a linear positioning joint or a rotational positioning joint. C-space map can be divided into two main areas, C-obstacle area which indicates the robot configurations in which collisions between the robot links and obstacles definitely occur and C-free area which represents the free allowable area for planning the path (Raheem and Hussain 2017a, b; Pan and Mancha 2015). The obstacle(s) can be represented geometrically and indicated as B in the equations below, so the C-space of the obstacles (C_{obs}) is as follows (Althoefer 1996):

$$C_{obs} = \left\{ q : A\left(q\right) \cap B \neq \varnothing \right\} \tag{12.26}$$

while the C-space free (C-free) area can be referred to as below (Pan and Mancha 2015):

$$C_{free} = \left\{ q : A\left(q\right) \cap B = \varnothing \right\} \tag{12.27}$$

where C_{obs} is C-space obstacles, q is robot configuration, and $A(q)$ are the set of points that are included in an area confined by configuration q of the robot. The overall C-space map can be represented as (Pan and Mancha 2015):

$$C_{space} = C_{obs} \cup C_{free} \tag{12.28}$$

The mathematical study of the surface properties, which do not change if this surface has deformations, such as bending or stretching, is called the topology of space. The topology for robot arms with two revolute joints can be described as (Lynch and Park 2017):

$$S^1 * S^1 = T^2 \tag{12.29}$$

The above equation means that the C-space for two revolute joint robot arms is a torus (with no joint limits for the joints), where S^1 is the circle description and $T^2$is the two-dimensional torus surface (Lynch and Park 2017). Each joint angle pair corresponds with a unique point on the torus, as shown in Fig. 12.21.

To find the mapping of C-space, basic shapes of obstacle(s) will be detailed.

12.12.1.1 Point Obstacle C-Space Construction

Mathematically, the simplest and essential for the beginning of C-space calculations which can be located in a robot workspace is the point obstacle. Herein, a modified derivation for C-space analysis will be discussed, in which collision checking is based on geometrical analysis. In the case of a point obstacle and according to the calculations of all probabilities of collisions, it can be noted that the collision area with the first link contains straight lines, while the collision between the point obstacle and the second link appears as a curved shape. As the point obstacle approaches the end effector of the planar robotic arm, the points of the C-space obstacle will be reduced, as presented in Fig. 12.22 (Raheem and Hussain 2017a, b).

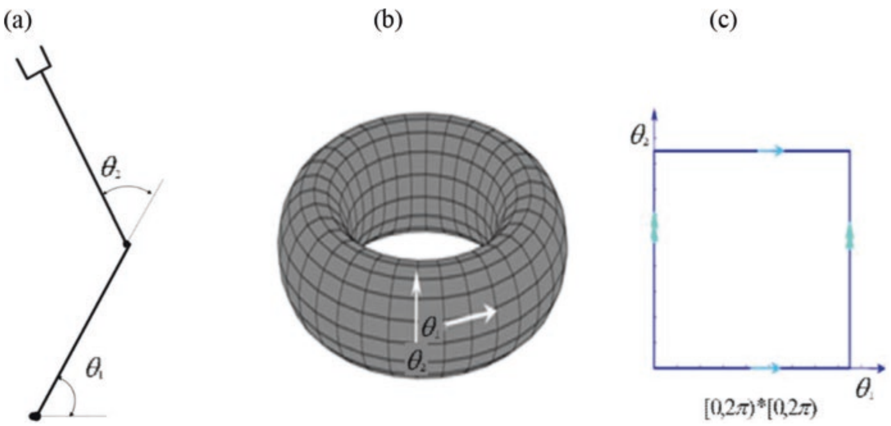

Fig. 12.21 Topological representation of the two-dimensional C-space. (**a**) 2R robot arm; (**b**) 2Torus; (**c**) sample representation. (Lynch and Park 2017)

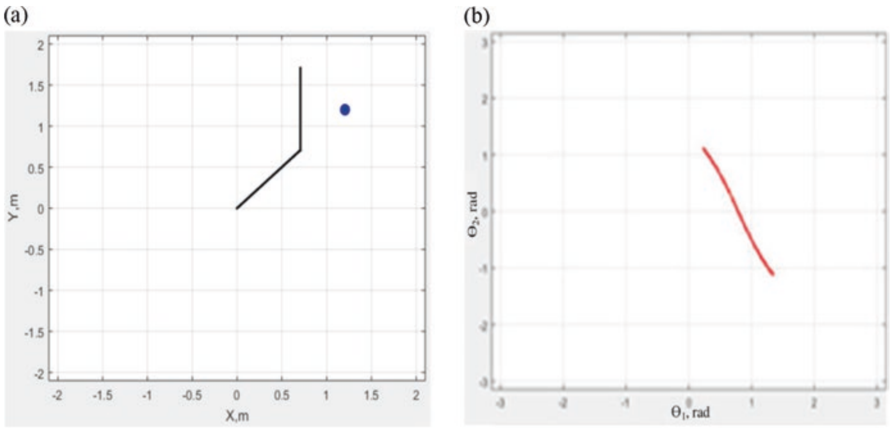

Fig. 12.22 C-space mass point analysis for (**a**) point obstacle collides with the second link only; (**b**) C-space curve where the curve points decrease as the point is nearest to the end effector

12.12.1.2 Line Obstacle C-Space Construction

Robot links can be handled as two lines, regardless of their width. Here, the derivation of C-space analysis comprises two analyses for collision testing; the same C-space curved map of the point obstacle can be applicable for each line(s) point. Thus, these compact form curves will represent the C-space of the line or polygonal obstacles, as illustrated in Fig. 12.23 (Raheem and Hussain 2017a, b).

A C-space map for any 2D shape formed by three or more straight lines will be considered a C-space map for a polygon. In case of a triangle (three lines), square, rectangle, rhombic (four lines), or more than four lines, the same method of

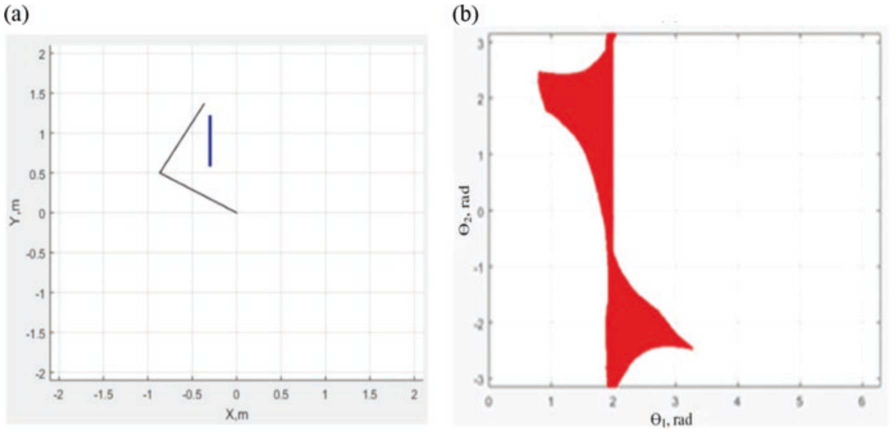

Fig. 12.23 C-space line obstacle analysis for (**a**) robot arm with line obstacle in the workspace, (**b**) corresponding C-space map

calculations and analysis for constructing the C-space map will be applied here as the method used for line obstacle (Bloch 2015).

In case the obstacle in the Cartesian workspace approaches closer to the first link, then the number of points on the C-space map increases. Besides that, the constructed C_{free} space will be on both sides of the constructed C_{obs} space. For that reason, finding a feasible path solution from one side of the C-space map to the other side requires choosing a suitable range of changing θ_1 for better unification of both regions which leads to finding a C_{free} path planning much easier, as presented in Fig. 12.24 (Raheem and Hussain 2017a, b).

12.12.1.3 Circle Obstacle C-Space Construction

A circle obstacle can be defined by a radius (rad) and center (x_c, y_c); the collision can be analyzed and tested by the derivation of C-space analysis.

Two main parts of edges of the C-space map constructed for a circular obstacle are presented in Fig. 12.25. Part (a) includes the collision between the circle obstacle circumference and the first link, which denote the vertical straight lines with all the probabilities of θ_2 values, while part (b) comprises the outer lower and outer upper curved shapes denoting the collision between the circle obstacle circumference and the second link (Raheem and Hussain 2017a, b).

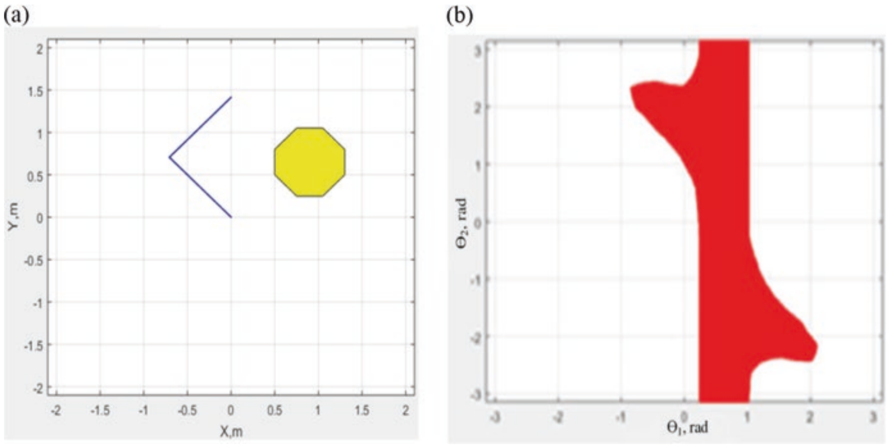

Fig. 12.24 C-space polygon obstacle analysis for (**a**) 2-DOF planar robotic arm Cartesian workspace contains octagonal obstacle. (**b**) Equivalent constructed C-space map

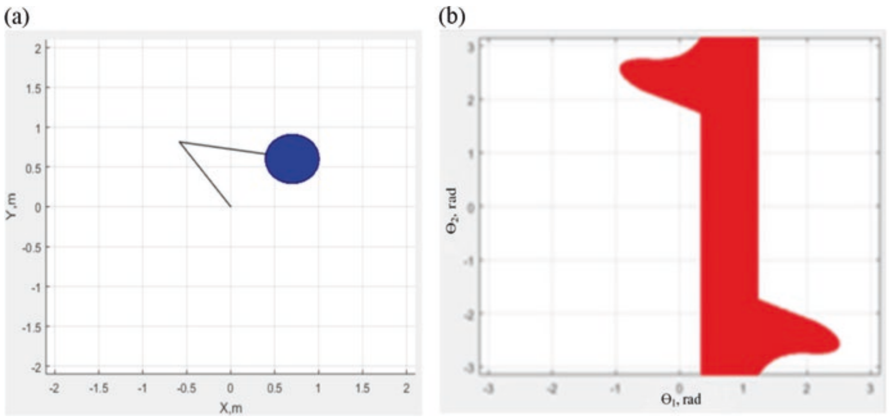

Fig. 12.25 C-space circle obstacle analysis for (**a**) 2-DOF planar robotic arm Cartesian workspace contains circle obstacle. (**b**) Equivalent constructed C-space map

12.12.2 Results of Applying A Algorithm on Modified C-Space*

Two environments have different obstacle shapes and tasks. First, we apply all the steps of C-space analysis and then perform the construction to get the exact map, which includes the collision and free areas. Second, the start and goal configuration, which are represented by the dot and the star shapes, respectively, have been computed by inverse kinematic equations. For the given test environment, the start and goal point are represented by the dot and the star shapes, respectively, as shown in Figs. 12.26 and 12.27 (Raheem and Hussain 2017a, b).

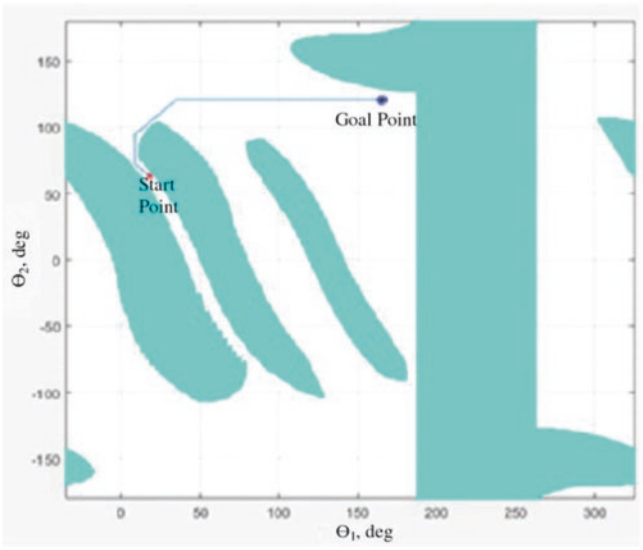

Fig. 12.26 A* path solution in the constructed C-space map according to the given environment task

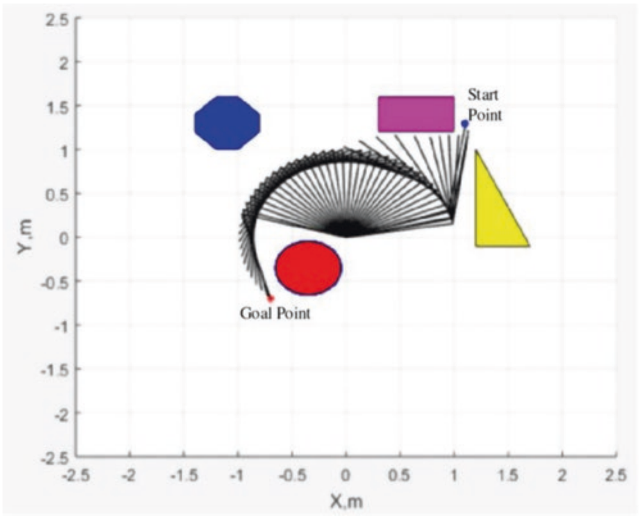

Fig. 12.27 2-DOF planar robotic motion from start to goal point for given environment task

12.13 Summary and Conclusion

Usually, finding the shortest feasible path is considered an optimization problem to determine the shortest path length from the start to the target points without colliding with obstacles. We included different robot path-planning applications and methodologies of their solutions depending on environment type, robot space, and the required calculations; we also showed the theory used to find a suitable path solution. The path-planning problem has been solved in both Cartesian and configuration spaces.

The analysis of the free Cartesian space for a 2-DOF planar robotic arm represents a significant task when complete information about the static robot arm environment is available and the path is globally offline, planned prior to the robot motion. While in the configuration space, a specific configuration of the robot can be represented by the values of joint angles as a point in the C-space map. Moreover, heuristic, A*, and D* methods are used to compute the best path solution in both known static and known dynamic environment.

The particle swarm optimization (PSO) technique has been used to make the smoothest, shortest path. The artificial potential field (APF) was coupled with PSO to tune the factors of the field construction to find the best mass point path solution. The results of these path-planning applications are simulated, analyzed, and compared to prove the effectiveness of the solutions. The techniques and solutions presented in this chapter have numerous "real-life" applications as the use of robotics increases to ensure unnecessary resource loss is not incurred in both static and dynamic environments in chemical, biological, geological, and temporal fields.

References

Althoefer K (1996) Neuro-fuzzy motion planning for robotic manipulators. Department of Electronic and Electrical Engineering, King's College London, University of London

Bloch ED (2015) Polygons, polyhedra, patterns & beyond. http://faculty.bard.edu/bloch/math107_notes.pdf. Accessed 22 May 2021

Boonyaritdachochai P, Boonchuray C, Ongsakul W (2010) Optimal congestion management in an electricity market using particle swarm optimization with time-varying acceleration coefficients. Comput Math Appl 60:1068–1077. https://doi.org/10.1016/j.camwa.2010.03.064

Choset H (2007) Robotic motion planning: A* and D* search. Carnegie Mellon University School of Computer Science: The Robotics Institute. Available via https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf. Accessed 22 May 2021

Cubero S (ed) (2007) Industrial robotics theory, modelling, and control. Verlag Robert Mayer-Scholz

Gue J, Gao Y, Cui G (2013) Path planning of mobile robot base on improved potential field. Inf Technol J 12(11):2188–2194. https://doi.org/10.3923/itj.2013.2188.2194

Hameed UI (2019) Development of path planning algorithm for robotic manipulator in dynamic environment. MSc thesis. University of Technology-Iraq, Baghdad, Iraq

Hussain AA (2017) New approaches for c-space map construction for heuristic path solutions based on PSO technique. MSc thesis. University of Technology-Iraq, Baghdad, Iraq

Li G, Yamashita A, Tamura HAY (2000) An efficient improved artificial potential field based regression search method for robot path planning. In: Proceedings of the 2000 IEEE international conference on robotics & automation, San Francisco, CA

Lynch KM, Park FC (2017) Introduction to robotics: mechanics, planning and control. Cambridge University Press

Nosrati M, Karimi R, Hasanvand H (2012) Investigation of the * (Star) search algorithms: characteristics, methods, and approaches. World Appl Program 2(4):251–256

Pan J, Mancha D (2015) Efficient configuration space construction and optimization for motion planning. Engineering Sciences Press

Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1:33–57. https://doi.org/10.1007/s11721-007-0002-0

Raheem FA, Badr MM (2017) Development of modified path planning algorithm using artificial potential field (APF) based on PSO for factors optimization. Am Sci Res J Eng Technol Sci 37(1):316–328

Raheem FA, Hameed UI (2018) Path planning algorithm using D* Heuristic method based on PSO in dynamic environment. Am Sci Res J Eng Technol Sci 49(1):257–271

Raheem FA, Hameed UI (2019) Interactive heuristic D* path planning solution based on PSO for two-link robotic arm in dynamic environment. World J Eng Technol 7:80–99

Raheem FA, Hussain AA (2017a) Applying A* path planning algorithm based on modified C-space analysis. Al-Khwarizmi Eng J 13(4):124–136

Raheem FA, Hussain AA (2017b) Applying A* path planning algorithm based on modified C-Space analysis. Al-Khwarizmi Eng J 13(4):124–136

Raheem FA, Sadiq AT, Abbas NF (2019) Robot arm free cartesian space analysis for heuristic path planning enhancement. Int J Mech Mech Eng IJMME-IJENS 19(01):29–42

Sreenivasulu R (2012) Simulation of Desired end point trajectory for A 2-Dof planar manipulator. Int J Adv Sci Tech Res 5:688–696

Stentz A (1994) The D* algorithm for real-time planning of optimal traverses. CML-RI-TR-94-37. The Robotics Institute, Carnegie Mellon University, Pittsburgh