An Intelligent Hybrid Control to Enhance Applicability of Mobile Robots in Cluttered Environments

TAIMOOR ZAHID1, ZAREENA KAUSAR2, MUHAMMAD FAIZAN SHAH3, MUHAMMAD TALLAL SAEED4,5, AND JIANFEI PAN6, (Member, IEEE)

1College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen 518060, China
2Department of Mechatronics and Biomedical Engineering, Air University, Islamabad 44000, Pakistan
3Department of Mechanical Engineering, Khawaja Fareed University of Engineering and Information Technology, Rahim Yar Khan 64200, Pakistan
4Department of Intelligent Systems and Control Engineering, School of Automation Sciences and Electrical Engineering, Beihang University, Beijing 100083, China

Corresponding author: Jianfei Pan (pan_jian_fei@163.com)

This work was supported in part by the Natural Science Foundation of China under Grant U1913214, and in part by the China Postdoctoral Foundation under Grant 2019M653038.

ABSTRACT Trajectory planning and tracking are the most important aspects of mobile robot research for industrial application. In this research, an intelligent hybrid control is presented to enhance the usability of mobile robots in environments cluttered with static obstacles. The control is hybrid in two ways. On one hand, the algorithm combines obstacles avoidance and trajectory generation. The generated trajectory acts as a reference path to be followed by the mobile robot. On the other hand, the algorithm not only generates trajectory but also tracks the generated trajectory. An optimization-based intelligent algorithm, simulated annealing, is designed to plan and generate a trajectory for collision-free robot motion in an environment with stationary obstacles. The PID (Proportional-Integral-Derivative) control algorithm is designed to track the generated trajectory with minimum error. Both algorithms are hybridized such that the generated trajectory becomes a reference trajectory for the tracking control algorithm. The simulated annealing generates a realistic trajectory that consists of a series of the best points with obstacles on the path. The best point is selected with the shortest distance from the robot’s destination. These points are selected one from each of the grids, generated between the start and destination point of the robot motion. The dimensions of the robot are considered and included in the obstacles dimension to reduce the dimension complexity of the robot. The effectiveness of the proposed hybrid technique is tested in simulation and in real-time experiments for seven types of cluttered environments. The environments vary with size, shape, placement and arrangement of the obstacles. The results show that the simulated annealing generates a collision-free trajectory intelligently without trapping in local minima in these cluttered environments while the PID control tracks the reference trajectory with a maximum absolute error of 0.2 mm. Hence, the proposed method enhances the usability of mobile robots in environments cluttered with static objects.

INDEX TERMS Mobile Robot, trajectory planning, obstacle avoidance, cluttered environment, trajectory tracking intelligent hybrid control.

I. INTRODUCTION Mobile Robots are used in industries making work and life easier and more expedient. They are very important and attention-grabbing especially in areas of manufacturing, construction, and human service. Motion control is a key function of intelligent moving robots. An ability to control robot movement is thus a minimum requirement for mobile robots [1]. A mobile robot needs to be moved on the desired path. This path is called a trajectory, to be followed by the mobile robot. A motion planning, therefore, includes the planning of the desired trajectory, and then desired planned trajectory generation and tracking completes the intelligent control of lower level of a mobile robot. When a path is planned in
an uncluttered environment for mobile robots, the kinematic constraints method is used to plan robot position using any algorithm [2] generating series of connected nodes from start to goal point. However, safe path planning in a cluttered environment is more important in robot motion control. A cluttered path is one with obstacles. The obstacles need to be detected and avoided. A suitable algorithm design for path planning in a simple and complex cluttered environment is, therefore, an important step in motion planning.

Currently, researchers are working on algorithm design for mobile robot trajectory planning [3]–[7]. In finding the secure route, therefore, trajectory planning needs to be researched for mobile robots traveling from one point to the other. Reducing processing time and energy consumed in trajectory generation and tracking becomes ultimately a prime requirement. This leads to the computation of the trajectory with the smallest length.

The goal of trajectory plan and implementation is to schedule and compute an adequate route with no obstacles in known cluttered surroundings. The route is from beginning to ending of path optimized under a certain norm [8]. Depending on the nature of the setting, there are certain characteristics of the trajectory planning of moving robots. These characteristics are: (i) if the setting is static or dynamic (ii) global or local. In a static setting the obstacles do not move other than the mobile robot and in a dynamic setting both obstacles and the robot moves. While local and global trajectory plan is controlled by an algorithm. It is decided if the surrounding information is already known. In a global trajectory plan, the prior information of the setting is known while the local plan robots have no prior information of the surroundings. In the later case the robot acquires the surrounding information by using physical sensors. The algorithm, on processing acquired data, proceeds to avoid obstacles moving to the destination point [9].

Many trajectory planning methods are available in the literature to determine the route of a moving robot with obstacle collision avoidance [10]–[14]. These methods are classified as classic and reactive methods. The classical methods include for example artificial potential field (APF), probabilistic roadmap method (PRM), and rapidly-exploring random trees (RRT) [15]–[17]. Each method owns its advantages and disadvantages. A common drawback of classical methods is the failure to respond to uncertainty in the environment. This makes it less preferred for real-time applications [18]. A major issue common in these methods is the trapping of the path at local optima. These methods, therefore, provide non-optimal solutions [19].

The reactive methods such as genetic algorithm, fuzzy logic, neural network, particle swarm optimization, and ant colony optimization have immense ability to handle uncertainty present in the environment due to obstacles [20]–[22]. Some shortcomings, however, in reactive path planning methods include low efficiency and high computation time. In complicated environments, a genetic algorithm (GA) method becomes slower with high computation cost [23], [24]. Therefore, an effective algorithm is required for the computation of a reasonable route in a dense area with good performance and short computation time.

Time optimal algorithm [25] and hybrid algorithm [26]–[29] suggested in literature have short computation time but have limited applications to robots with wheels or series configuration. The other main issue is the tracking of a planned trajectory. The optimal trajectories assessed using the objective functions are converted into actuator instructions by the tracking controller. Trajectory tracking is generally defined as the process concerned with the way to determine the speed of the robot and steering settings at each instant of time, to make the robot follow a trajectory.

A trajectory is a set of points that represents the positional coordinates of a certain route. Often when dealing with a trajectory tracking problem, one also has to implement a trajectory recording unit that can store all the coordinates of the desired trajectory. Then a human operator has the possibility to steer the robot manually along some tracks but the trajectory recording unit is needed to save the information about the trajectory. The unplanned positional deviation from the trajectory is also handled by the trajectory tracking algorithm. Such deviations from the trajectory can be caused by odometric errors or by new obstacles which come on the way to the planned.

Motivated by the above observations, this study proposes the development of an effective algorithm for intelligent trajectory planning and generation for mobile robots in a cluttered static environment by using the simulated annealing method. The proposed approach with an optimal solution of trajectory is hybridized with a PID trajectory tracking control as a reference path. Preliminary information on the simulated annealing is given in section II. While in next section model of the cluttered environment and trajectory in mathematical form is presented. Section III describes objective function, cooling plan and algorithm for optimization whereas PID controller implementation is given in section IV. Section V is about the experimental procedures, results and discussions followed by the conclusion in section VI.

II. OPTIMAL TRAJECTORY PLANNING

The method in this study for optimal path planning combines features of simulated annealing (SA) for its advantage of parameter optimization and non-trapping in local minima of a solution. Preliminary knowledge of SA, the algorithm, and analogy of SA with physical annealing is established in the following sub-sections.

A. SIMULATED ANNEALING (SA)

SA, an optimization method, discovers the best solution from all the possible solutions. Annealing is a method where the material is liquefied from its solid-state increasing the temperature. Then solid-state is restored by slightly lowering the temperature of the material. This cooling process allows the particles to obtain a complete crystalline structure when solidifies.
In simulated annealing, an energy function of the crystal structure of the resulting solid material is considered equivalent to the global minimum. Temperature is decreased gradually allowing each particle to settle itself in a perfect way [23]. The process of gradual cooling of liquid metal is used to search for a minimum of function making it cost minimization problem. Boltzmann’s notion of probability distribution is proposed for the temperature to gradually cool the liquefied metal. The Boltzmann probability distribution for thermal equilibrium at temperature, the energy $E$ is distributed probabilistically as (1).

$$P(E) = e^{-E/kT} \quad (1)$$

where $P(E)$ is the probability of attaining energy level; $E$ is the energy level and $k$ is the Boltzmann’s constant. Equation (1) depicts that at high temperatures the probability of system, at any energy state, is almost identical. However, at lower temperatures, the system is less likely to be in a high energy state.

**B. SA ALGORITHM**

The proposed SA algorithm for optimal path planning is as follows:

(i) Set an initial, preferably high, temperature. Set iteration number equal to unity i.e. ($i=1$).

(ii) Set off a random point in the neighborhood of the recent point. Estimate difference in values of function as portrayed in (2).

$$\Delta E = \Delta f = f_{i+1} - f_i \equiv f(X_{i+1}) - f(X_i) \quad (2)$$

Such that

- $\Delta f$ = Difference in values of function
- $f(X_{i+1})$ = New value of function
- $f(X_i)$ = Initial value of function

The solution is accepted if the new value of function $f(X_{i+1})$ is less than the initial value of function $f(X_i)$. Otherwise, adopt a solution with probability estimated from (3):

$$P(E) = e^{-\Delta E/kT} > r \quad (3)$$

"$e$" is a change in function value, "$T$" is current temperature, "$r$" is a random number between 0 and 1.

(iii) If a new solution or point is rejected, the process of generating random points, function value estimation, application of metropolis criterion to adopt a new solution continues.

(iv) If a new point is accepted at a temperature, the temperature is decreased by a predefined fraction value ($0 > \alpha < 1$) and repeat step (ii) to (iii).

The solution is believed to be best and converged if the temperature or change in function value ($\Delta f$) is too small. A decrease in temperature is identified as the cooling plan. The slight temperature drop depicts a good chance of discovering the right valley before attempting to reach the lowest point in the valley [24].

**C. ANALOGY OF PHYSICAL AND SIMULATED ANNEALING**

An analogy is developed between the simulated annealing parameters and physical annealing parameters following [30] as listed in Table 1.

| Physical Annealing | Simulated Annealing |
|--------------------|---------------------|
| System state       | Feasible solution   |
| Energy             | Cost                |
| Change of state    | Neighbor solutions  |
| Temperature        | Control parameter   |
| Frozen state       | Optimal solution    |

**III. OPTIMAL TRAJECTORY GENERATION**

This section, in order to generate the optimal trajectory, presents how the surrounding environment of the robot is modelled. Also, trajectory representation and objective function for optimization are modelled. Moreover, the assumptions, structure, cooling plan and pseudo code of the proposed algorithm are presented mathematically before their simulation results.

**A. ENVIRONMENT MODELLING**

We have deemed a 2D environment where robot can move in a cluttered environment where obstacles are fixed. The start point is ($x_s, y_s$) and the goal point is ($x_g, y_g$). The mobile robot is expected to avoid obstacles and find a collision-free path to the goal point. In the environment, all obstacles are considered as of circular shape. The black circles shown in the Figure 1 represent the obstacles. As in real settings robots have some dimensions. Therefore, to model, the environment sizes of all obstacles are inflamed by fixed values as shown in Fig.1.

![FIGURE 1. Obstacles inflamed by a dimensional value shown as white circle. The difference in black and white circle’s diameters represents the dimension of a robot.](image)

The obstacle inflammation is carried out for mobile robots to move in the surrounding without collision. The size of the robot is added to the obstacle by increasing the size of the obstacle and the mobile robot is considered a point. By doing this, we have transformed the problem of robots staying away from obstacles to a point of moving avoiding obstacles. This reduces the severity of the problem as well.
The obstacles are of many shapes. We assumed here, for simplicity, obstacles of the shape of a circle, shown in Figure 2. The radius of the circle depends upon the size of the obstacle. The bigger the size of the obstacle, the radius of the circle will be increased according to the size of the obstacles. Some of the cases are described in Figure 2 to show how we can consider the different shaped obstacles as a circle.

**FIGURE 2.** Circles modelled for different polygons shapes of obstacles.

As the robot dimensions cannot be ignored the diameter of the circle increases according to the size of the robot. For example, if we have a square robot then we look at the center point of the robot chassis and half of the width of the robotic chassis is added to the circle distance to avoid robot collision with obstacles.

**B. TRAJECTORY REPRESENTATION**

The probable trajectory of a robot is indicated by a series of points connecting the first point \((x_s, y_s)\) with the goal point \((x_g, y_g)\). This possible trajectory is represented as (4):

\[
X = \{(x_0, y_0), (x_1, y_1), \ldots, (x_i, y_i), (x_g, y_g)\} \tag{4}
\]

**C. OBJECTIVE FUNCTION**

An objective of the proposed study is to determine the shortest and most probable route of the robot between two locations. The algorithm starts with discovering the shortest distance between the destination point and the newly engendered points. As shown in Fig. 3 shorter length represents a better solution. The objective function, assessing the length of the route, used in the proposed study is given in (5):

\[
f(x, y) = \sqrt{(x_g - x)^2 + (y_g - y)^2} \tag{5}
\]

where,

- \((x_g, y_g) = \text{Destination point coordinates.}\)
- \((x_i, y_i) = \text{Random generated point coordinates}\)

**D. ASSUMPTIONS**

Before developing the algorithm, the following assumptions were made:

- The robot surrounding has stationary obstacles only.
- All obstacles are considered circular.
- The best route selected for the robot consists of a series of best points from the beginning of the point to the goal point

**E. ALGORITHM STRUCTURE**

Generate grids in every iteration by drawing perpendicular lines on x and y axis as shown in the Figure 4.

**FIGURE 4.** Grids produced (at different iterations) drawing perpendicular lines on x and y axis.

The main structure of the SA algorithm is made up of random points produced in grids of different sizes. A distance of each random point, produced in the grid, from the destination point is calculated. The point with the shortest distance from the destination point is chosen as the best point. Then the random points in that grid are refreshed using SA. Re-select the point with the shortest distance. Compare two best points using (6):

\[
\Delta f = f(i + 1) - f(i) = f(X_{i+1}) - f(X_i) \tag{6}
\]

A negative value of \(\Delta f\) means the new point is better than the previous point. This is because the distance of new point to the goal point is less than the previous point. In this way, the algorithm automatically accepts that point. The positive value of \(\Delta f\) means the distance of a new point to the goal point is greater. And the new point appears to be worse in comparison to the previous point. But to avoid local minimum we accept the best point based on the probability function as given in (7)

\[
P[f(i + 1)] = e^{-\Delta f/T} \tag{7}
\]

As from the above equation, it is clear that the probability of accepting the worse point depends on the value of \(T\). At the start of the proposed process, the accepting probability of worse move is high because of the high value of \(T\). So for accepting the worse move we generate a random number between 0 and 1 and compare the random number generated with the probability. If the probability is greater than the random number generated, then we accept the worse solution and the chances of accepting is high because the probability at the start of the process is high else the above process is...
repeated again. When a point is selected for a certain $T$ value and by a cooling plan, given in (8), the $T$ value decreases then the process of generating new random points and selection of the best point are made to that particular $T$ value. In this research the value of $\alpha = 0.95$ in order to slowly decrease the value of $T$. The value of $T$ is 50. As from equation (7) it is clear that the chances of receiving negative movements are reduced by decreasing $T$. At the end of the process we have the best solution or the best point which is near to the goal point in that grid. This whole process is completed in one grid. In Fig. 4 the selected best point is shown in one grid. The goal point is set to be (1.9,1.9). Fig. 5 shows how a best point is selected in a number of random points generated on a single grid near the goal point. The red ‘∗’ is the best point.

$$T(k + 1) = \alpha T(k)$$ (8)

Now we increase the grid size towards the goal point and the above process is repeated. The best point i.e. point with the least distance to goal point is selected for each grid using the SA algorithm. This process of increasing the size of the grid and selecting the best point continues until it reaches the goal point as shown in Fig. 6.

The random points produced in the generated grids are checked for obstacle avoidance finding whether they lie inside the obstacle area. If the generated point lies within the obstacle area then that point is removed. This is because the generated point is unachievable and cannot be selected as shown in Fig. 7. Therefore, points outside the obstacle area are selected for testing and the best point is selected from these points. In this way, the robot can be kept away from obstacles and no collision occurs.

The quality of the solution is assessed by trajectory length. A solution with a shorter robot trajectory length, between new and goal points, is a better solution.

**F. COOLING PLAN**

The cooling plan is the most important part of the SA algorithm. It is a function that will determine how the system temperature is adjusted. Many cooling plans are used to make changes in temperature for simulated annealing [31]. These include geometric cooling, logarithmic cooling, adaptive cooling, and reheating being a function of cost. In this study, a geometric cooling plan is used. The geometric method is the most widely used in simulated annealing and serves as a basis for comparison with other broad methods. The geometric cooling method is simple as it does not take into account the state of the system [32]. In geometric cooling, temperature is
updated using expression described in (9):

\[ T_{k+1} = \alpha T_k \]  

(9)

The cooling speed parameter \( \alpha \) is considered to be a fixed value. This is less than but close to one. The typical rate of slow cooling \( \alpha \) is guessed between 0.9 and 0.99 in the search process.

**G. PSEUDO-CODE OF THE ALGORITHM**

The pseudo-code for the algorithm proposed is given in Table 2.

**TABLE 2.** Pseudo-code of the algorithm.

| Step | Action |
|------|--------|
| 1.   | Set \( T_{in} \) an initial temperature |
| 2.   | Generate initial solution |
| 3.   | Estimate fitness function |
| 4.   | for \( i=1 \) to \( n \) |
| 5.   | Generate random number |
| 6.   | Estimate fitness function |
| 7.   | Select the best function |
| 8.   | Compare best with initial function |
| 9.   | if new function < initial function |
| 10.  | Accept new solution |
| 11.  | else |
| 12.  | Accept solution with probability |
| 13.  | end |
| 14.  | Save best solution |
| 15.  | Decrease \( T \) as per cooling plan |
| 16.  | end for |

**IV. TRAJECTORY TRACKING CONTROL**

Developing intelligently controlled mobile robots, which have the ability to behave autonomously in transporting from one place to another, involves at least two activities. First trajectory planning is an important task to avoid collision of the robot with the obstacles. The second important task for the robot is trajectory tracking. In trajectory tracking the computer is assumed to have the predefined planned trajectory as given in the previous section. Normally the computer takes full control of the robot and follows the predefined trajectory considering all the parameters which change position and create errors along the way. A controller is used which makes the robot to return to its original track.

Trajectory tracking is generally defined as the process concerned with how to determine the speed of the robot and steering settings at each instant of time in order to make the robot to follow a trajectory. A trajectory is a set of points that represents the positional coordinates of a certain route. Often, when dealing with a trajectory tracking problem, one also has to implement a trajectory recording unit that can store all the coordinates of the desired path. Then a human operator has the possibility to steer the robot manually along some track but the trajectory recording unit is needed to save the information about the trajectory. The unplanned positional deviation from the trajectory is also handled by the trajectory tracking algorithm. Such deviations from the trajectory can be caused by odometric errors or by new obstacles which come on the way of the trajectory, and those obstacles must be avoided. There are many different types of trajectory tracking algorithms [33]–[36] available today but the industry prefers and also to start with simple a PID control algorithm is selected for this research.

**A. PID CONTROL ALGORITHM**

The PID controller is a form of feedback controller used in more than 95% of the control loops in the industry. The reason of selecting PID controller is partly due to their robust performance in a wide range of operating conditions and partly due to their functional simplicity, which allows the users to operate them in a simple, straightforward manner.

Let \( u(t) \) be the control signal which is sent to the system, the measured output is \( y(t) \), the desired output is \( r(t) \), and the tracking error is \( e(t) = r(t) - y(t) \). PID control law written in the general form is given in (10).

\[ u(t) = K_pe(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \]  

(10)

By adjusting the three parameters \( K_p, K_i \) and \( K_d \) the desired closed-loop dynamics is obtained. The PID controller takes a measured value from a process and then compares the measured value with a reference point value. The difference in the measured value and reference value, the error signal, is then used for the adjustment of input to the system as per control law (10).

**B. ROBOT MODEL**

The state of the robot is expressed by Newton’s second law of motion. Input \( u(t) \) is an acceleration; to find the input in terms of force, we multiply \( u(t) \) with \( m \), the mass of the body as given in (11) [23].

\[ F = ma \]  

(11.a)

\[ a = \frac{F}{m} \]  

(11.b)

\[ \ddot{x} = \frac{F}{m} \]  

(12)

Such that \( x = \) Position; \( \dot{x} = \) Velocity; \( \ddot{x} = \) Acceleration

Let \( u \) be the input and \( u \) is defined by (13)

\[ \ddot{x} = \frac{F}{m} = u \]  

(13)

Now writing the state space

\[ x = x_1 \]  

(14)

\[ \dot{x}_1 = \dot{x} = x_2 \]  

(15)

\[ \ddot{x}_1 = \ddot{x} = \dot{x}_2 \]  

(16)

Such that

\[ \dot{x}_1 = x_2 \]  

\[ \dot{x}_2 = u \]
Writing in matrix form we get
\[
\dot{x} = Ax + Bu
\]
\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix} 
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
1
\end{bmatrix} u
\]  
(17)
(18)

In the state vector differential equation, the matrices are in (19)
\[
A = 
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix} 
\quad \text{and} \quad B = 
\begin{bmatrix}
0 \\
1
\end{bmatrix}
\]  
(19)

C. CONTROL PARAMETER TUNNING

A proportional control gain ($K_p$) reduces the rise time and the steady-state error. An integral control gain ($K_i$) eliminates the steady-state error but makes the transient response worse. However, derivative control gain ($K_d$) increases the stability of the system, reducing the overshoot, and improving the transient response. The PID controller, therefore, is designed to tune three parameters $K_p$, $K_i$ and $K_d$ to stabilize the system minimizing overshoot, rise time, and settling time to 5%, 1 sec, and 5 sec respectively.

The result shows that by setting the value of $K_p = 4$, $K_i = 2$ and $K_d = 10$ the controller tracks the trajectory for the robot with set control parameter values of rise time, overshoot and settling time. The rise time is 0.17(sec). The settling time is less than 3(sec) and the overshoot is less than 4 %. The closed-loop system model diagram used for simulation is given in Figure 8.

D. REFERENCE TRAJECTORY

In this research, the reference path is the trajectory computed by simulated annealing in different environments. The trajectory is a series of points from start to goal point in $x$, $y$ coordinates and is the position of the robot in $x$, $y$ coordinates. Our aim is to track the position of the trajectory points for the robot.

V. RESULTS AND DISCUSSION

The results are simulated for generation and tracking of planned trajectory in a various cluttered environments in following sub-sections to verify the proposed algorithms.

VOLUME 9, 2021

A. SIMULATION RESULTS FOR TRAJECTORY PLANNING

In the simulation, the trajectory planning of a moving robot is performed in several situational circumstances with diversified obstacles in different locations for a range of goal points. The results attained, using the SA algorithm in various tests, are either optimal or close to optimal. Dark circles represent obstacles in the surroundings while line circles represent the size of the obstacles plus the dimensions of the robot. This addition is to avoid collisions as the robot considered in this study is assumed to be a point. The increased size of an obstacle depends on the dimensions of the robot needed to perform the different tasks. The axes represent the position of the robot in a two-dimensional plane.

Test 1: In Fig. 9, results of a simple test in a surrounding having 5 obstacles are presented. Filled black circles are obstacles in the robot’s surrounding. Blue stars (∗) in a sequence represent the best points available for the robot trajectory. Fig. 5 shows the optimal trajectory computed for the robot to move from the beginning to the target without colliding. The test parameters and their respective results are tabulated in table-3 for a view in a glance.

Test 2: In test 2 a complex area with more obstacles is considered. Fig. 10 shows that optimal trajectory with no collision is computed for the environment with same sized obstacles having the same goal point as in test-1. However.
number of obstacles is increased. Fig. 9 shows the optimal computed trajectory for the robot to move from the beginning to the target without colliding as well.

Test 3: In test 3 the robot surrounding with more number of obstacles i.e. nine is presented. All obstacles are again of the same size. Results shown in Fig. 11 reveal that the algorithm works equally well for more complex environments in computing an optimal trajectory without colliding.

Test 4: In test 4 the robot motion in an environment with six obstacles of different sizes is presented. As the size of obstacles may differ in realistic situations. Results given in Fig. 12 show that the proposed algorithm is able to compute the shortest trajectory in presence of multiple sizes of the obstacles.

Test 5: The test 5 is the same as tests 1 & 2 but with more than double the number of obstacles to reach a closer target point that is evident from table-3. The simulation results of this test are shown in Fig. 13. The results verify the capability of the SA algorithm to deal with as many obstacles on the way of a robot as possible and generating optimal trajectories without colliding.

Test 6: The test 6 is the same as test 5 but with a different goal point. Results in Fig. 14 show that the change of goal point in a highly complexly cluttered surrounding does not affect the efficiency of the proposed algorithm in finding an optimal collision-free path.

Test 7: In this test complexity of the environment is increased for 14 obstacles by selecting obstacles of variable sizes as shown in Fig. 15. This setting is simulated to see how well the proposed algorithm works when the size of all the obstacles in the surrounding of a robot is dissimilar. Fig.15 shows that the algorithm works well in a very difficult environment with a large number of obstacles of different sizes.

B. SIMULATION RESULTS FOR TRAJECTORY TRACKING

In the simulation, tracking of an optimal computed path by a mobile robot is performed in several situations. The later includes the different number of obstacle, changes in obstacle location, and variation in destination points. The trajectory is computed by the SA algorithm in different test conditions. The trajectory consists of a series of best points computed on the robot navigation path starting from the initial to the target point in x-y plane. This also describes the locations and
coordinates of the robot position in x-y plane. The objective of the control design is to track the optimally computed trajectory points for the robot. The results obtained using the PID controller in four selected environments show that the trajectory is tracked, using the PID controller, with the minimum error from the desired values. The desired control parameters were set as: the overshoot $\leq 5\%$; Rise time should be less than 1(sec) and; the settling time should be less than 3(sec). In the results given in the following subsections, the ‘black’ trajectory is the reference trajectory and the ‘green’ trajectory is the actual trajectory.

1) TRAJECTORY TRACKING FOR TEST 2
Fig. 16 shows that the trajectory computed by using simulated annealing for Test 2 is tracked by using the PID controller. For this test the value of $K_p = 4$, $K_i = 2$ and $K_D = 10$. The controller tracks the trajectory for the robot with acceptable values of rise time, overshoot, and settling time. The rise time is 0.17s. The settling time is less than 3s and the overshoot is less than 4%. The maximum error is 0.02 cm approximately.

2) TRAJECTORY TRACKING FOR TEST 4
Fig. 17 shows that the trajectory computed by using SA for test 4 is tracked by using the PID controller. For this test, we used the same values of parameters as used for test 2 i.e. $K_p = 4$, $K_i = 2$ and $K_D = 10$. The controller tracks the trajectory for the robot with acceptable values of rise time, overshoot, and settling time. The rise time is 0.17s. The settling time is less than 3s and the overshoot is less than 4%. The maximum error is recorded as 0.02 cm approximately.

3) TRAJECTORY TRACKING FOR TEST 5
Fig. 18 shows that the trajectory computed by using SA for test 5 is tracked by using the PID controller. For this test we used the same values of parameters as used for the trajectory tracking of tests 2 and 4. The results show that the controller tracks the trajectory for the robot with acceptable values of rise time, overshoot and settling time. The rise time is 0.17s. The settling time is less than 3s and the overshoot is less than 4%. The maximum error is recorded as 0.02 cm approximately.
4) TRACKING RESULT FOR TEST 7

Fig. 19 shows that the trajectory computed by using simulated annealing for test 7 is tracked by using the PID controller. For this test, we used the same values of parameters as used for earlier tests i.e. \( K_p = 4, K_i = 2 \) and \( K_d = 10 \). The trajectory is followed with a rise time of 0.17 s. The settling time is less than 3 s and the overshoot is less than 4%. The maximum error is equal to 0.02 cm approximately.

So from the above results, it is clear that the designed PID control algorithm can track different trajectories for a robot moving in different environments.

C. EXPERIMENTAL RESULTS

In order to validate the applicability of the proposed hybrid intelligent scheme, experiments are conducted in above mentioned seven test environments. The architecture of a wheeled robot used for experiments is shown in figure 20(a). Six proximity sensors are positioned at the periphery of the robot base. They are positioned at equal distance. The proximity sensors detect the obstacles while the robot is in motion.

Four-wheel encoders are installed to measure and give feedback of the distance covered by the robot. The algorithm for trajectory planning and tracking is implemented using Arduino® microcontroller, controlling the motion of four drive wheels through motors.

The experimental setup for test-1 performed in an environment of 5 obstacles is shown in figure 20(b). The rest of the experiments (test # 2-7) are conducted in the same setup but with a different number and size of obstacles as described in section V-B. The resulting length of the path in each case is plotted to compare simulation and experimental results as shown in figure 21. A difference in experimental results may be due to sensor measurement error and the sensor positions. These challenges raise a motivation for future work.

VI. CONCLUSION

In this paper, an intelligent hybrid control algorithm to enhance the usability of mobile robots in environments...
cluttered with static obstacles is presented. This hybridized algorithm served two purposes. Firstly, it avoids obstacles as well as generates an intelligent trajectory. Secondly, a complete algorithm generates as well as tracks the generated intelligent trajectory. The hybrid algorithm combined SA and PID for trajectory planning and tracking for a mobile robot.

In trajectory planning, SA used is designed to optimize the trajectory length between the initial and the final point of the robot path. This proposed SA algorithm not only plan but computes a trajectory avoiding all possible collisions of the mobile robot. The algorithm is capable of generating optimal trajectories for cluttered environments with different locations, number and sizes of obstacles. A sufficient trajectory is worked out by joining a series of best points, each best point belongs to a grid created between starting and destination points. The simulation and real-time results of the optimal trajectory estimation using SA verify that the SA algorithm is effective in finding a possible trajectory with a short distance from the beginning to the final destination point in cluttered surroundings without being caught in local minima. This trajectory planning & generation is further hybridized with a trajectory tracking algorithm. The tracking problem is solved by using PID controller. The results demonstrate that the proposed hybrid control algorithm intelligently computes the trajectory by SA algorithm tracked by PID controller within an acceptable values of ≤ 5%, ≤ 1s, and ≤ 3s for overshoot, rise time and settling time respectively. Whereas, the maximum error is minimized to 0.2 mm. Through the collision-free trajectory, computed using the SA-based approach, the hybrid intelligent scheme is verified for usability. The future work includes a comparative study with other reactive methods and extension of this work to dynamic environment.

ACKNOWLEDGMENT
(Taimoor Zahid and Zareena Kausar contributed equally to this work.)

REFERENCES
[1] J. Latombe, Robot Motion Planning. New York, NY, USA: Springer, 2012, doi: 10.1007/978-1-4615-4022-9.
[2] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” Robot. Sci. Syst., vol. 6, pp. 267–274, May 2011, doi: 10.15607/issa.v10i034.
[3] V. F. Muñoz, A. García-Cerezo, and A. Cruz, “A mobile robots trajectory planning approach under motion restrictions,” Int. J. Comput.-Aided Eng., vol. 6, no. 4, pp. 331–347, 1999, doi: 10.3233/icaa-1999-640.
[4] Y. Zi Cong and S. G. Ponnambalam, “Mobile robot path planning using ant colony optimization,” in Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, Jul. 2009, pp. 851–856, doi: 10.1109/AMIN.2009.5229903.
[5] O. Hackour, “Path planning of autonomous mobile robot,” Int. J. Syst. Appl., Eng. Develop., vol. 2, no. 4, pp. 178–190, 2008.
[6] C. Sprunk, “Planning motion trajectories for mobile robots using splines,” Dept. Comp. Sci., Albert-Ludwigs Univ. Freiburg, Freiburg im Breisgau, Germany, May 2009. [Online]. Available: http://www2.informatik.uni-freiburg.de/~lau/students/Sprunk/2008.pdf
[7] D. T. Woodard, “Graph-based path planning for mobile robots,” Ph.D. dissertation, Dec. 2006, p. 114.
[8] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, “Adaptive evolutionary planner/navigator for mobile robots,” IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 18–28, Apr. 1997.
[31] R. L. Haupt, S. E. Haupt, and A. J. Wiley, *Algorithms*, 2nd ed. New York, NY, USA: IEEE, 2004.
[32] M. de Sales Guerra Tsuzuki, T. de Castro Martins, and F. K. Takase, “Robot path planning using simulated annealing,” *IFAC Proc. Vol-umes*, vol. 39, no. 3, pp. 175–180, 2006, doi: 10.3182/20060517-3-fr-2903.00105.
[33] K. N. Faress, M. T. El Hagry, and A. El Kosy, “Trajectory tracking control for a wheeled mobile robot using fuzzy logic controller,” *WSEAS Trans. Syst.*, vol. 4, no. 7, pp. 1017–1021, 2005.
[34] R. Solea, A. Filipescu, and U. Nunes, “Sliding-mode control for trajectory-tracking of a wheeled mobile robot in presence of uncertainties,” in *Proc. 7th Asian Control Conf.*, Aug. 2009, pp. 1701–1706.
[35] D. U. Panfilov and S. B. Tkachev, “Tracking of reference trajectory for wheeled mobile robot,” *Process Control*, vol. 3, pp. 2–8, Oct. 2003.
[36] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. Hoboken, NJ, USA: Wiley, 2012.

TAIMOOR ZAHID received the B.Sc. degree in electrical engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2011, with the focus on computer engineering, and the Ph.D. degree in computer applied technology from the University of Chinese Academy of Sciences, China, in 2018. He is currently working with the College of Mechatronics and Control Engineering, Shenzhen University. His research interests include robotics, intelligent control, and energy management systems.

ZAREENA KAUSAR received the B.S. degree in mechanical engineering from UET, Lahore, Pakistan, the M.S. degree in mechanical engineering from the National University of Ireland and the second M.S. degree in mechatronics engineering from UET, and the Ph.D. degree from The University of Auckland, New Zealand. She worked as the Chair with the Department of Mechatronics Engineering, Air University, Islamabad. Her research interests include nonlinear control systems, robotics, and bio-mechatronics.

MUHAMMAD FAIZAN SHAH received the B.E. degree in mechatronics engineering from Air University, Islamabad, in 2015, and the M.S. degree in mechanical engineering from Khawaja Fareed University of Engineering and Information Technology (KFUEIT), Rahim Yar Khan, in 2019. He is currently working as a Lab Engineer with the Department of Mechanical Engineering, KFUEIT. His research interests include intelligent manufacturing systems, robotics, and control systems.

MUHAMMAD TALLAL SAEED received the B.Sc. and M.S. degrees in electrical engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2011 and 2013, respectively, and the Ph.D. degree in control theory and control engineering from Beihang University, China, in 2019. He is currently working as an Assistant Professor with the Department of Mechatronics and Biomedical Engineering, Air University, Islamabad. His research interests include modeling and simulation, robust and optimal control, and intelligent control.

JIANFEI PAN (Member, IEEE) received the Ph.D. degree from the Department of Electrical Engineering, The Hong Kong Polytechnic University, Hong Kong, in 2006. He is currently working with the College of Mechatronics and Control Engineering, Shenzhen University. His main research interests include motion control, actuator design, and robotics. 